TXB-S422

22 March 2002

TABLE OF CONTENTS

Section **Page**

[1] `$Header: d:/sears/RCS/latexit.l,v 1.15 2002-02-26 11:42:00-08 Hamilton Exp Hamilton $`
[2] Input file = "`txbs422.asm`".
[3] `$Header: d:/sears/RCS/txbs422.asm,v 1.122 2002-03-13 14:27:25-08 Hamilton Exp Hamilton $`

This page intentionally left blank

1 TXB-S422 Source code description

1.1 Global Defines

Note

Note that small "boxed" numbers, i.e. 123, refer to approximate source code line numbers.

13

If DEBUG isn't specified and real debugging is being done, then it is very unlikely that the system will boot up properly. During a normal boot up sequence this software looks at the Spectra and expects several events to happen in sequence. If the sequence of initial events is improper, then this software will hang-up, mostly waiting for a reply from the Spectra with the Spectra's address in it. As a solution to this problem, when you are in DEBUG mode it is possible to specify an address for the Spectra. This gets around the whole boot up problem. If you are not running in DEBUG mode, then each time that the software gets changed there must be a complete power cycle of the Spectra done with careful starting of the TXB-S422's software. (Not a fun activity!)

```
DEBUG
```

31

Make it so that when debugging it is possible to avoid asking the Spectra its address. This specifies what address to use.

```
DEBUG_ADDRESS   44
```

39

TEST_PROPORTIONAL is used to enable testing of proportional speed commands from an RC216 (VM96)controller when we only have an AD2083/02. (The RC216 sends different speed values than the AD2083/02 does. This forces a test for controller type to always report than an RC216 is installed when any variable speed pan command comes in.)

```
TEST_PROPORTIONAL
```

52

ALLOW_INPUT_FLIP is used to enable/disable the input data flipping logic. If defined then flipping is enabled.

```
ALLOW_INPUT_FLIP
```

61

LIST_INCLUDES when defined, makes all included files list themselves out.

```
 LIST_INCLUDES
```

68

Change the default radix from hex to decimal so that this code may be written in a civilized manner. (Either enable it here on in the project file.)

```
radix dec
```

77

errorlevel -302 is used to get rid of "Register in operand not in bank 0. Ensure that bank bits are correct." messages which gets a little old after awhile.

```
errorlevel -302
```

```
  p = processor type, c = columns/page, n = lines/page,
  st = no symbol table
  Pick whatever type of processor you wish from this list
 list    p=pic16f873,  c=132, n=58, st=off
list    p=pic16f876,  c=132, n=58, st=off

  Options for the 16F87x series of CPUs are:

  Brown out enable:              _boden_off   _boden_on
  Code Protection:               _cp_all      _cp_half
                                 _cp_off      _cp_upper_256
  Code Protect Data EEPROM:      _cpd_off     _cpd_on
  Debug enable:                  _debug_off   _debug_on
  Flash write enable:            _wrt_enable_off
                                 _wrt_enable_on
  Low voltage programing enable: _lvp_off     _lvp_on
  Oscillator type:               _hs_osc      _lp_osc
                                 _rc_osc      _xt_osc
  Power up timer enable:         _pwrte_off   _pwrte_on
  Watch dog timer enable:        _wdt_off     _wdt_on

  The only difference between "debug" and "normal" mode configuration
  settings is that in "debug" mode I do not enable code protection.
  Code protection is only enabled in released code. MicroChip does
  not recommend enabling it on EPROM type devices.
```

```
        ifdef DEBUG
         __config _boden_on&_hs_osc&_pwrte_on&_wdt_off&_cp_off&_wrt_enable_off
        endif   ifdef DEBUG

        ifndef DEBUG
          __config _boden_on&_hs_osc&_pwrte_on&_wdt_off&_cp_off&_wrt_enable_off
         __config _boden_on&_hs_osc&_pwrte_on&_wdt_off&_cp_all&_wrt_enable_off
        endif   ifndef DEBUG
```

1.2 Include files

1.2.1 dproto.inc

```
 1 ; "$Header: d:/sears/RCS/dproto.inc,v 1.23 2002-02-28 12:57:21-08 Hamilton Exp
Hamilton $"
 2 ; Copyright by Pelco, 2000, 2001, 2002
 3         nolist
 4         ifdef LIST_INCLUDES
 5          list
 6         endif ; LIST_INCLUDES
 7 ;
 8 ; To help with this long boring process, I have put together a bunch of
 9 ; macros which specify various bits in various words.  For example:
10 ;
11 ;        D_FOCUS_FAR is defined to be d_command2_M,bit7
12 ;
13 ; So all that has to be done is to see if a bit is set and then work with
14 ; it.
15 ;
16 ; All names involved in this start with a D for D protocol.
17 ; Defined names are uppercase, buffer names and values are in
18 ; lower case (usually).
19 ;
20 ; Defines to control which bits get set for coding commands in D protocol
21 #define D_ALTUSE          d_command1_M,bit7 ; Controls meaning of bits 3 and 4
22 #define D_SPARE6          d_command1_M,bit6 ; spare
23 #define D_SPARE5          d_command1_M,bit5 ; spare
24 #define D_AUTO_SCAN_ALT   d_command1_M,bit4 ; Auto scan with DALTUSE = 1
25 #define D_MANUAL_SCAN     d_command1_M,bit4 ; Manual scan with DALTUSE = 0
26 #define D_CAMERA_ON_ALT   d_command1_M,bit3 ; Camera on with DALTUSE = 1
27 #define D_CAMERA_OFF      d_command1_M,bit3 ; Camera off with DALTUSE = 0
28 #define D_IRIS_CLOSE      d_command1_M,bit2 ; Close Iris
```

```
29 #define D_IRIS_OPEN       d_command1_M,bit1 ; Open Iris
30 #define D_FOCUS_NEAR      d_command1_M,bit0 ; Focus near
31
32 #define D_FOCUS_FAR       d_command2_M,bit7 ; Focus far
33 #define D_ZOOM_OUT        d_command2_M,bit6 ; Zoom wide
34 #define D_ZOOM_IN         d_command2_M,bit5 ; Zoom tele
35 #define D_TILT_DOWN       d_command2_M,bit4 ; Down
36 #define D_TILT_UP         d_command2_M,bit3 ; Up
37 #define D_PAN_LEFT        d_command2_M,bit2 ; Left
38 #define D_PAN_RIGHT       d_command2_M,bit1 ; Right
39 #define D_EXTENDED        d_command2_M,bit0 ; 1 = extended command
40
41 #define D_TILT_MOTION     0x18              ; Tilt motion mask
42 #define D_PAN_MOTION      0x06              ; Pan motion mask
43 #define D_MOTION          0x1E              ; Any motion mask
44
45 #define D_STOP            0x00              ; All zeros stops motion
46
47 #define D_PAN_SPEED       dproto5_M         ; Pan speed
48 #define D_TILT_SPEED      dproto6_M         ; Tilt speed
49
50 #define D_SYNC            0xFF              ; Sync, 1st, byte
51
52 ; Command codes for use with extended commands in D protocol
53 #define D_CLEAR_AUXILIARY          0x0B
54 #define D_CLEAR_PERSET             0x05
55 #define D_CLEAR_SCREEN             0x17
56 #define D_END_PATTERN              0x21
57 #define D_FLIP                     0x21 ; call 33
58 #define D_GOTO_PRESET              0x07
59 #define D_HOME                     0x22 ; call 34
60 #define D_MENU                     0x5F ; set 95
61 #define D_QUERY                    0x45
62 #define D_RESET                    0x0F ; was 0x29
63 #define D_RUN_PATTERN              0x23
64 #define D_SET_AUXILIARY            0x09
65 #define D_SET_PRESET               0x03
66 #define D_START_PATTERN            0x1F
67 #define D_WRITE_CHAR               0x15
68         list
69 ; End of PROTOCOL.INC
70 ;
```

1.2.2 sdefs.inc

```
1 ; "$Header: d:/sears/RCS/sdefs.inc,v 1.17 2002-02-27 15:58:31-08 Hamilton Exp
Hamilton $"
2 ; Definitions to use with Sensormatic's RS422 protocol
3 ; Copyright by Pelco, 2001, 2002
4         nolist
5         ifdef listincludes
6          list
7         endif ; listincludes
8
9 ; Commands to the dome
10 ;
11 #define S_UNKNOWN80        0x80 ; Unknown three byte command
12 #define S_PAN_LEFT         0x81 ; Pan left (24/sec) until Pan Right or Pan
13                                 ; Stop
14 #define S_PAN_RIGHT        0x82 ; Pan right (24/sec) until Pan Left or Pan
15                                 ; Stop
16 #define S_PAN_STOP         0x83 ; Stop panning
17 #define S_TILT_UP          0x84 ; Tilt up until Tilt Down or Tilt Stop
18 #define S_TILT_DOWN        0x85 ; Tilt Down until Tilt Up or Tilt Stop
19 #define S_TILT_STOP        0x86 ; Stop tilting
20 #define S_FOCUS_NEAR       0x87 ; Focus near until Focus Far or Focus Stop
21 #define S_FOCUS_FAR        0x88 ; Focus far until Focus Near or Focus Stop
22 #define S_FOCUS_STOP       0x89 ; Stop Focus
23 #define S_ZOOM_IN          0x8A ; Zoom in until Zoom Out or Zoom Stop
24 #define S_ZOOM_OUT         0x8B ; Zoom out until Zoom In or Zoom Stop
25 #define S_ZOOM_STOP        0x8C ; Stop zoom
26 #define S_FAST             0x8D ; Increase pan and tilt speeds to 48/sec
27                                 ; until Fast Stop
28 #define S_FASTEST          0x8E ; Increase pan and tilt speeds to 96/sec
29                                 ; until Fast Stop
30 #define S_FAST_STOP        0x8F ; Stop fast/fastest speeds (back to normal
31                                 ; 24/sec)
32 #define S_IRIS_OPEN        0x90 ; Opens iris (manual iris mode/lightens
33                                 ; Iris Preference offset (auto iris mode)
34                                 ; until Iris Close or Iris Stop
35 #define S_IRIS_CLOSE       0x91 ; Closes iris (manual iris mode/darkens
36                                 ; Iris Preference offset (auto iris mode)
37                                 ; until Iris Open or Iris Stop
38 #define S_IRIS_STOP        0x92 ; Stop iris offset adjustment (also stops
39                                 ; V-Phase Adjust)
40 #define S_ALL_STOP         0x93 ; Stop all movement
```

```
41 #define S_GET_DOME_TYPE    0x94 ; Request dome type or poll
42 #define S_GET_ALARMS       0x95 ; Request status of alarm inputs
43 #define S_ACK              0x97 ; ACKnowledge sent to dome responds to
44                                 ; asynchronous commands.
45 #define S_UNKNOWN98        0x98 ; Start temp no transmit
46 #define S_UNKNOWN99        0x99 ; End temp no transmit
47 #define S_FASTER           0x9A ; Increase pan and tilt speeds to 72/sec
48                                 ; until Fast Stop
49 #define S_FASTER_STOP      0x9B ; Stop faster speeds (back to normal 24/sec)
50 #define S_DEFINE_BOUNDARY  0x9C ; Start boundary definition. This command
51                                 ; is followed by dome movement commands and
52                                 ; tour Mark Boundary commands.
53 #define S_MARK_BOUNDARY    0x9D ; Marks the current position as a boundary
54 #define S_ON_AIR           0x9E ; Set On Air status to tell the dome to
55                                 ; send the asynchronous boundary crossing
56                                 ; command
57 #define S_ON_AIR_RESET     0x9F ; Reset On Air status
58 #define S_DEFINE1          0xA0 ; Start defining Pattern 1
59 #define S_DEFINE2          0xA1 ; Start defining Pattern 2
60 #define S_DEFINE3          0xA2 ; Start defining Pattern 3
61 #define S_NEW_PATTERN      0xA3 ; Accept the new pattern as the current
62                                 ; pattern and delete the old pattern.
63 #define S_DUMP_DOME_MEMORY 0xA4 ; Dump dome memory
64 #define S_GET_POSITION     0xA5 ; Request Dome position Coordinates
65 #define S_GOTO_POSITION    0xA6 ; Goto absolute position (Multiple-byte
format)
66 #define S_MARK1            0xA8 ; Mark the current position as Target 1
67 #define S_MARK2            0xA9 ; Mark the current position as Target 2
68 #define S_MARK3            0xAA ; Mark the current position as Target 3
69 #define S_MARK4            0xAB ; Mark the current position as Target 4
70 #define S_GOTO_PAT1        0xAC ; Go to the start of pattern 1
71 #define S_GOTO_PAT2        0xAD ; Go to the start of pattern 2
72 #define S_GOTO_PAT3        0xAE ; Go to the start of pattern 3
73 #define S_GOTO_PAT4        0xAF ; Go to the start of pattern 4
74 #define S_RUN1             0xB0 ; Run Pattern 1
75 #define S_RUN2             0xB1 ; Run Pattern 2
76 #define S_RUN3             0xB2 ; Run Pattern 3
77 #define S_RUN_NEW          0xB3 ; Run a newly defined pattern to review it
78                                 ; before accepting it to replace the
79                                 ; previous pattern.
80 #define S_GOTO1            0xB4 ; Go to preset position called Target 1
81 #define S_GOTO2            0xB5 ; Go to preset position called Target 2
```

```
82 #define S_GOTO3            0xB6 ; Go to preset position called Target 3
83 #define S_GOTO4            0xB7 ; Go to preset position called Target 4
84 #define S_PATTERN_END      0xB8 ; Tells the dome to stop recording
85                                   ; (defining) a pattern
86 #define S_MARK5            0xB9 ; Mark the current position as Target 5
87 #define S_MARK6            0xBA ; Mark the current position as Target 6
88 #define S_MARK7            0xBB ; Mark the current position as Target 7
89 #define S_GOTO5            0xBC ; Go to preset position called Target 5
90 #define S_GOTO6            0xBD ; Go to preset position called Target 6
91 #define S_GOTO7            0xBE ; Go to preset position called Target 7
92 #define S_VARIABLE_SPEED   0xC0 ; Variable speed control (New command.
93                                   ; Multiple-byte format)
94 #define S_UNKNOWN_C1       0xC1 ; Unknown
95 #define S_PROP_SPEED       0xC3 ; Proportional speed pan or tilt movement
96                                   ; commands (Multiple-byte format)
97 #define S_PROP_LEFT        0x81 ; Subcommand of 0xC3, pan left
98 #define S_PROP_RIGHT       0x82 ; Subcommand of 0xC3, pan right
99 #define S_PROP_UP          0x84 ; Subcommand of 0xC3, tilt up
100 #define S_PROP_DOWN        0x85 ; Subcommand of 0xC3, tilt down
101 #define S_UNKNOWN_C4       0xC4 ; Unknown three byte command
102 #define S_RESET            0xC6 ; Run default "Apple Peel" pattern for a
103                                   ; spiral view of everything (only supported
104                                   ; by SpeedDome Ultra IV and DeltaDome, or
105                                   ; later products)
106 #define S_SOFTWARE_VERSION 0xC9 ; Get software version number from dome
107 #define S_OUTPUT0          0xE0 ; Clears all active drivers
108 #define S_OUTPUT1          0xE1 ; Set output Driver #1
109 #define S_OUTPUT2          0xE2 ; Set output Driver #2
110 #define S_OUTPUT3          0xE4 ; Set output Driver #3
111 #define S_OUTPUT4          0xE8 ; Set output Driver #4
112 #define S_TERM_PAT         0xF0 ; Stop/terminate current pattern
113
114         space 2
115 ; Messages from the dome
116
117 #define S_DOME_TYPE_IS     0x94 ; Response to Request Dome Type
118 #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
```

144

All defines which get used to "org" memory locations are located here. Note that PROGRAM_PAGE0 starts several locations AFTER the first core location where it may. This is so that if needed, an OTP type chip may be reprogrammed with some carefully written code. See the MicroChip manual "PICmicro Mid-Range MCR Family Reference Manual", DS33023A, December 1997, for information on how to do this.

```
RESET_VECTOR    0x0000    Reset vector entrance
PROGRAM_PAGE0   0x0010    Program area 0
PROGRAM_PAGE1   0x0800    Program area 1
```

159

RAM bank starting locations are defined here. There are some bad "holes" in each bank. I have listed those that I know of at the start of allocating usage for each bank.

```
RAM0            0x020     RAM Bank 0
RAM1            0x0A0     RAM Bank 1
RAM2            0x120     RAM Bank 2
RAM3            0x1A0     RAM Bank 3
```

171

Internal EEPROM defines.

This memory must be initialized before use. This is done transparently to the user by setting a flag in four consecutive bytes. If the flag is set then there is no initialization is needed, but if it is missing then the preset ID is set to `0x00`.

When EEPROM has been initialized the four EE_SETMEMx bytes will have `0xDE`ADBEEF written into them, i.e. EE_PROM0 = `0xDE` and EE_PROM3 = `0xEF`

```
EE_PRESET        0x22     Handy place in EEPROM to use for presets IDs
EE_SET_MEM0      0x40     EEPROM flag for being initialized, part 1 0xDE
EE_SET_MEM1      0x41     EEPROM flag for being initialized, part 2 0xAD
EE_SET_MEM2      0x42     EEPROM flag for being initialized, part 3 0xBE
EE_SET_MEM3      0x43     EEPROM flag for being initialized, part 4 0xEF
```

191

Pan and Tilt Speed defines. It is important to note that the "names" of the various speeds do not match what the speeds actually go at. These speeds have been verified with the VM1 (RC58) system at Sears Fresno. The RC58 keyboard is a fixed speed unit. To get more than one speed from it requires that one of three "speed enhancement" keys be depressed while a dome is in motion. The speed enhancement keys are marked "RAIL $\longrightarrow$" (fastest), "Fast" (faster) and "$\longleftarrow$ RAIL" (fast). These three keys generate the speed commands indicated in parenthesis following their names. "fastest" gives an X2 speed increase, "faster" gives an X3 speed increase, and "fast" gives an X4 speed increase. (Not exactly what any sane person would choose for names vs. speeds!) (And then the speed changes that the Spectra makes is non-linear, and has intentional "holes" for acoustic noise reduction, when compared to the values sent to it.)

```
                 value   is  should be
PAN_DEFAULT       51     26   27/sec
PAN_FASTEST       59     55   54/sec
PAN_FASTER        63     80   81/sec
PAN_FAST          64     150 108/sec

                 value  gives
TILT_DEFAULT      36     12/sec
TILT_FASTEST      48     19/sec
TILT_FASTER       57     30/sec
TILT_FAST         63     44/sec
```

221

Misc defines.
Internal D-protocol defines are first.

```
PAN_RIGHT       this_command2_M,bit1
PAN_LEFT        this_command2_M,bit2
TILT_UP         this_command2_M,bit3
TILT_DOWN       this_command2_M,bit4
ZOOM_IN         this_command2_M,bit5
ZOOM_OUT        this_command2_M,bit6
FOCUS_NEAR      this_command2_M,bit7
```

```
FOCUS_FAR       this_command1_M,bit0
IRIS_OPEN       this_command1_M,bit1
IRIS_CLOSE      this_command1_M,bit2
```

240

General use misc defines.

```
BAD_TRY             5     Number of bad inputs before flipping
BLANK            0x20     Blank character
CR               0x0D     Carriage return
 IDLE_LOOKS       216     How many times to look for an idle
                          IDLE_LOOKS must be a multiple of 9
IGNORE_ADDRESS     64     Messages to address 64 are ignored
LF               0x0A     Line feed
LONG_TIME_OUT    continuationtimer_M,bit1   End point of a long timeout
PELCO_SPEED_DOME 0xF8     Dome type to send to the controller.
PELCO_ULTRA_DOME 0xF5     Dome type to send to the controller.
                          SpeedDomes send an "E" type of response
                          until they get five poll commands and then
                          send an "F" response.
                          0xE5 = SpeedDome 2000 2-board response dome 2
                          0xF5 = SpeedDome Ultra response        dome 4
                          0xF8 = SpeedDome 2000 1-board response dome 5
                          0xE8 = SpeedDome 2000 1-board response dome 1
S_GOTO_POSITION_SIZE 13   0xA6 Command length
S_PROP_SPEED_SIZE     4   0xC3 Command length
S_PROTO_LENGTH        0   Fake length of an S protocol command
S_UNKNOWN_C1_SIZE    12   0xC1 Command length
S_VARIABLE_SPEED_SIZE 5   0xC0 Command length
```

268

IO port definations.

```
spare2          porta,bit0        2 spare spare spare
ENABLE_INPUTS   porta,bit1        3 RS-422/485 input enable
                                    Control inputs from the head-end
                                    0 enables inputs
                                    1 disables inputs
INVERT_IO       porta,bit2        4 Used to invert input/output data
                                    0 = normal input levels
                                    1 = input data is inverted
                                  Controls an input chips inversion ctl
```

```
                                    INVERT_IO Input Output Format
                                        0       1     1    normal
                                        0       0     0    normal
                                        1       1     0    inverted
                                        1       0     1    inverted
SPECTRA_OUT     porta,bit3          5 Serial data to Spectra
spare6          porta,bit4          6 spare spare spare
SPECTRA_IN      porta,bit5          7 Serial data from Spectra
notexist1       porta,bit6          - IO pin does not exist
notexist2       porta,bit7          - IO pin does not exist
```

292

It is important to always remember that the address switch bits come in "backwards", i.e. if the switch is ON then it comes in as a zero (0) and if it is OFF then it comes in as a one (1).

```
SPEEDCONTROL0   portb,bit0          21 Speed changer
SPEEDCONTROL1   portb,bit1          22 Speed changer
                                       Bits 0 and 1 give slower and
                                       slower speeds, the more that are
                                       set.
                                    Additional logic REQUIRES that these
                                    two bits be together and in positions
                                    0 and 1.
SPEEDMASK       0x03                Mask for just speed. (Use this to
                                    find the logic for implementing
                                    this option.) This speed option only
                                    affects RC216 mode speed commands.
NODELAY         portb,bit2          23 Disable or enable delays before
                                    and after sending data. Normal
                                    is to have a 250 us delay before
                                    the start of the first bit and
                                    1 ms after the last bit. These
                                    delays are to enable the RS422
                                    drivers to stabalize their
                                    outputs and to allow others to
                                    use the channel.
                                       0 enables normal before and
                                         after delays
                                       1 disables starting and ending
                                         delays
spare24         portb,bit3          24 Only unused bit of address switch
PERMIT64        portb,bit4          25 Permit address 64 to be used
```

```
DEBUG_MODE_ON   portb,bit5      26 Enables debug outputs SW1-6
DATA_OUT        portb,bit6      27 ICSP and RS-232 to debug monitor
spare28         portb,bit7      28 ICSP and spare otherwise

spare11         portc,bit0      11 spare spare spare
spare12         portc,bit1      12 spare spare spare
spare13         portc,bit2      13 spare spare spare
spare14         portc,bit3      14 spare spare spare
spare15         portc,bit4      15 spare spare spare
ENABLE_OUTPUTS  portc,bit5      16 RS-422/485 output enable
                                   Controls outputs to the head-end
                                   0 disables outputs
                                   1 enables outputs
UART_OUT        portc,bit6      17 Output of the USART
UART_IN         portc,bit7      18 Input to the USART
```

342

Mode bits.

Bit usage in the mode1_M word:

```
                   I            7 In Motion
                    N           6 No more presets
                     M          5 Detected controller type
                      C         4 Clear the Spectra's screen
                       E        3 Error detected
                        D       2 Serial debug mode is enabled
                         r      1 reserved, must be 0
                          P     0 Signal polarity status

NORMAL_POLARITY mode1_M,bit0    0 = normal polarity, 1 = reversed
RESERVED_BIT    mode1_M,bit1    MUST remain at 0
INPUT_ERROR     mode1_M,bit3    Input error flag
CLEAR_DISPLAY   mode1_M,bit4    Should the Spectra display be cleared?
                                0 = No
                                1 = Yes
RC216_MODE      mode1_M,bit5    Indicates the mode for speed cmnds
                                0 = AD2083/02, or other 8 speed src
                                1 = RC216, or other many speed device.
                                    These devices send speeds out as
                                    degree/second motion values.
NO_MORE_PRESETS mode1_M,bit6    Enable/Disable preset processing
```

```
                                    This value is controlled by timer 1
                                    and is set after getting a preset
                                    command and is cleared after waiting
                                    for three more to come in.
                                    0 = Enable processing
                                    1 = Disable processing
      IN_MOVEMENT     mode1_M,bit7  Indicates if we are processing any
                                    motion command. When we are not
                                    processing a motion command a
                                    "faster" command causes a flip.
                                    0 = Not in movement
                                    1 = Yes in movement


Bit usage in the mode2_M word:

                         s          7 spare
                          s         6 spare
                           s        5 spare
                            s       4 spare
                             s      3 spare
                              b     2 Used to indicate
                               b    1 . . when bytes come
                                b   0 . . . . in during Spectra sending

NOTE all bits are cleared by clearing the full byte
      BYTE1           mode2_M,bit0  Used in receiving while xmitting
      BYTE2           mode2_M,bit1  . . to the Spectra
      BYTE3           mode2_M,bit2  . . . . indicates that all are in
      IN_FAST         mode2_M,bit3  Indicates that we are in a "fast"
                                    type command.
                                    0 = No
                                    1 = Yes

Bit usage in the mkmode1_M byte

In the land of Sensormatic, several camera options are selected by
using several buttons simultaneously. These bits are used to control
the detecting of these special simultaneous button depressings.

                         s          7 spare
                          s         6 spare
```

```
                    s             5 spare
                     F            4 Last command was "focus x"
                      I           3 Last command was "iris open"
                       F          2 Last cmnd was "fast"
                        -         1
                         -        0

STEP1RESET      mkmode1_M,bit0    A dome reset consists of two commands
STEP2RESET      mkmode1_M,bit1    First a zoom out followed by a focus
                                  far and last by an iris open command
LAST_FAST       mkmode1_M,bit2    Set when "fast" (0x8D) is received
                                  0 = Normal command
                                  1 = Last command was a "fast" command
STEP1MENU       mkmode1_M,bit3    Set when "iris open" (0x90) is rcvd
                                  0 = Normal command
                                  1 = Last command = "iris open"
STEP2MENU       mkmode1_M,bit4    Set when a "focus x" command is
                                  received (0x87, near or 0x88, far).
                                  0 = Normal command
                                  1 = Last command = "focus x"
```

434

DBAUD_BASIC and DBAUD_REPEAT are used to control how long to delay when sending each bit, of bit-banged, data to the Spectra at 2400 baud. The original plan was to take baudperiod and multiply it by 8 (19200 = 2400 * 8) to get a delay time. However when this is done the result is greater than an eight bit number can hold (376) so a pair of values were needed. One for a major loop and one for a minor inside loop.

```
DBAUD_BASIC     47
DBAUD_REPEAT     8
```

457

The Baud Rate Generator is used to provide a clock to the on-board USART which is used for output communications with the head end. All other serial IO is bit-banged.

BAUD1200, BAUD2400, BAUD4800, BAUD9600 and BAUD19200 are used to load the baud rate generator for the onboard USART. The data for all five of these equates assumes that BRGH is 0 (for low speed operation which gives an internal divide of 64 instead of 16.) It is assumed that the xtal is cut for 11,059,200 Hz (AKA 11.0592 MHz).

Baud Rate Generator formula from page 101 is:

$$baudrate = Fosc/(64(BRG + 1))$$

Solving for BRG gives us:

$$BRG = (Fosc/(baudrate * 64)) - 1$$

or

$$BRG = ((Fosc/64)/baudrate) - 1$$
$$BRG = ((11095200/64)/baudrate) - 1$$
$$BRG = (173362.5/baudrate) - 1$$

The above calculations got me to an approximate starting value. I then fooled around with the numbers until they worked.

```
BAUD1200        143        1,200 baud
BAUD2400         68        2,400 baud
BAUD4800         34        4,800 baud
BAUD9600         17        9,600 baud
BAUD19200         8       19,200 baud
```

493

Define bit positions for bit testing.

```
bit0            0x00      Bit position '0'
bit1            0x01      Bit position '1'
bit2            0x02      Bit position '2'
bit3            0x03      Bit position '3'
bit4            0x04      Bit position '4'
bit5            0x05      Bit position '5'
bit6            0x06      Bit position '6'
bit7            0x07      Bit position '7'
```

1.3 Naming conventions

510

Naming conventions used are as follows:

Each RAM location will be have a name of some type that ends with an underscore "_" and the area of RAM that the variable is located in. There are only the following areas of memory: 0 = Last 96 locations in RAM bank 0, 1 = All 96 bytes of RAM bank 1. In case of doubt always assume that the item is globally accessible, most of the unmarked items are defined by the MicroChip assembler and I do not want to change their names.

Macros are in all uppercase. (Just like in well written C programs.)

Note that the following RAM locations are non-functional. That is why there are "holes" in the memory allocations as they represent unused peripheral options.

0x08, 0x09, 0x88, 0x89, 0x8F, 0x90, 0x95, 0x97, 0x9A, 0x9B, 0x9C, 0x9D

RAM 0 is used to store almost all variables.

The number following the name is the subroutine level that uses that RAM location. A RAM location may be used only by that level of subroutine, or the main line code. However a higher level routine may READ but NOT modify a lower level byte. There is a special suffix allowed and that is _M for items only accessed from the main line code.

There is a special convention adopted to indicate which bank of RAM is active. That convention is to always select RAM bank 0, UNLESS a different bank is needed for some reason AND when a different RAM bank is selected to indicate it in the right hand column.

1.4 RAM usage

```
cblock          RAM0      Defines the RAM bank 0 register file

badcount_M                Number of errors in input data
bytebits_1                Number of bytes to bitbang out
checksum_M                Checksum for long messages
continuationtimer_M       Used for long timeouts when setting presets
                          See also LONG_TIME_OUT's use and definition
                          which is near the "again" tag.

d_sync_M                  1 D proto sync byte
d_address_M               2 D proto address byte
d_command1_M              3 D proto command 1 (All iris, focus near)
d_command2_M              4 D proto command 2 (Focus far, all zoom,
                             . . all motion)
d_pan_speed_M             5 D proto data 1 (Pan speed)
d_tilt_speed_M            6 D proto data 2 (Tilt speed/presets etc.)
d_checksum_M              7 D proto checksum

eewritehere_0             Where to write into in EE memory
extratimeout_0            For long timeouts that are too big
hexlower_0                Debug lower half of a byte in ASCII
hexupper_0                Debug upper half of a byte in ASCII
lastchecksum_M            Most recent checksum
manycount_5               Repeating count for repeated reply bytes
messcount_4               Debug message counter/indicator thing
mkmode1_M                 Command bits relating to multiple
                          . . simultaneous key depressions
mode1_M                   System mode part 1
mode2_M                   System mode part 2
```

```
        overlong1_0             Used in communicating with the Spectra
        overlong2_0             . these are needed because 2400 baud
                                . . is real slow and we can't delay
                                . . . long enough with just 8-bits
        patternwas_M            Last pattern worked on
        save3_M                 D Protocol saving space
        save4_M                 D Protocol saving space
        save5_M                 D Protocol saving space
        save6_M                 D Protocol saving space
        saved_pan_speed_M       Saved values are updated by "fast" cmnds only
        saved_tilt_speed_M
        sentbyte_1              Byte being bit banged out
        spectraaddress_M        Address for the Spectra
        sprotolength_M          Length of this S protocol message

        sproto1_M               S protocol address,     byte 1
        sproto2_M               S protocol command,     byte 2
        sproto3_M               S protocol checksum,    byte 3
        sproto4_M               S protocol long message byte 4, motion speed
        sproto5_M               S protocol long message byte 5
        sproto6_M               S protocol long message byte 6
        sproto7_M               S protocol long message byte 7
        sproto8_M               S protocol long message byte 8
        sproto9_M               S protocol long message byte 9
        sproto10_M              S protocol long message byte 10
        sproto11_M              S protocol long message byte 11
        sproto12_M              S protocol long message byte 12
        sproto13_M              S protocol long message byte 13

        temp1_0                 Used to get around a limited instruction set
        temp2_0                 Used to get around simple instruction set
        this_command1_M         The full series of "this" values exist
        this_command2_M         . so that they will not be modified
        this_pan_speed_M        . by other output commands, such as
        this_tilt_speed_M       . anything in the preset series, etc.
        thisbit_0               Used in bitbanging in Spectra data
        thisioerror_0           Most recent serial IO error
        timeout_0               Decrementing delay value


RAM 1 is spare
        cblock          RAM1    Define RAM bank 1
```

```
The last location MUST be 0x0FF or lower
      endc                      ends RAM bank 1 definitions...

RAM 2 is spare
      cblock          RAM2      Define RAM bank 2
The last location MUST be 0x17F or lower
      endc                      ends RAM bank 2 definitions...

RAM 3 is spare
      cblock          RAM3      Define RAM bank 3
The last location MUST be 0x1FF or lower
      endc                      ends RAM bank 3 definitions...
```

640

Hardware resets, master clears, etc., start running here.

```
      org     RESET_VECTOR
      goto    init_M            Code for init_M, MUST be in code page 0
```

1.5 General coding rules

650

In writing this code several coding rules were followed (if you look carefully, you can find exceptions to these, but there shouldn't be too many).

1. All code is in instruction bank 0.

2. All RAM is in bank 0.

3. Whenever any variation to the above rules is made, then the affected bank (so far it is always a RAM or register bank) is indicated in column 80. I.e. a "3" in column 80 indicates that whatever is happening is going to happen in RAM/register bank "3".

4. All closed subroutines have their nesting level indicated in their name as a suffix. I.e. doaux_3 may be called by subroutines xx_4 ... xx_8 but not by xx_3. And more especially it may not be called by xx_0 ... xx_2 type subroutines.

5. Variables have the lowest modifying subroutine indicated by their suffix. I.e. hexlower_0 is a RAM location that gets changed by a xx_0 subroutine. Thus hexlower_0 may not be used with any assurance that it will remain undamaged by higher level subroutines such as doaux_3. There is a special class of RAM locations and those are those that have a suffix of "_M", and these are modified by the "main" line code. RAM bank 0 is used for most purposes, however the other RAM banks have various IO related registers that must be accessed from time to time and they are the ones that cause problems. This is because I didn't want to change MicroChip's names that come in from the include file to indicate the "bank of interest". So I just left them alone and hope that the "column 80" comments will be enough. Nothing in column 80, indicates that everything is happening in RAM bank 0.

1.6 "main"

687

The system starts runtime execution at "start". When the initialization code couplets, it transfers control to "start". Any required restarts, including error restarts transfer control to "again". (Had to change start to startstart so that I could find it easily.)

1.6.1 Special "set preset" note

700

The Sensormatic equipment sends requests for location (set preset commands) out three times. My logic involves incrementing a counter each time and then sending that information back to the controller. However I should only increment my counter once when getting in a series of preset sets. The solution to this is to disable processing presets, set a timer going that should last for two more position requests and to then reenable processing presets.

Preset-ignore timeouts work as follows:

1. A set preset command is detected.

2. A timer is started. The timer is based on the number of instructions executed and being only 16 bits long is not long enough to give the required timeout duration.

3. When the timeout sequence is started NO_MORE_PRESETS is set. And both parts of timer 1 are cleared.

4. Every time through the main loop, starting at "again", the state of timer 2 is checked. When the upper half of it has its most significant bit set, then the continuation timer is incremented. And timer 2 is again reset to zero.

5. When the continuation timer is large enough NO_MORE_PRESETS is cleared, the timer is turned off and the system continues on as though everything is normal.

6. An effort is made to have the NO_MORE_PRESETS flag set for $1/2 \longrightarrow 1$ seconds. The exact maximum timeout time is not too important. (I hope!)

The timer MUST be active long enough to get past the two extra preset requests and not so long as to slow down a user that is setting presets. I have decided that a timeout that is always at least 1/2 second long and is usually less than 2 1/2 seconds long is about right.

1.6.2 decodeinput_M

764

decodeinput_M This is a computed goto subroutine that is called from main and is used to decode commands in the range of `0x80` through `0xFF`.

On entry w has the command code as does sproto2_M Return is made to any of 64, or more, different places.

For several camera functions, the method of "calling" them is to depress several keyboard buttons simultaneously in a specific order. When this happens various special camera operations are activated. If the order changes, or a "stop" command (which would indicate that the button was released) comes in, then the special sequence is stopped.

The following commands get no response back:

```
0x00 --> 0x7F, 0x97, 0x98, 0x99, 0xA7, 0xBF
0xC1, 0xC2, 0xC4, 0xC5, 0xC7
```

783

The following commands get an ACK, but nothing is done inside the TXB-S422:

```
0xC8, 0xCA --> 0xDF, 0xF1 --> 0xFF
```

805

A flip command set consists of a "fast" (`0x8D`) command immediately followed by a "fastest" (`0x8E`) command. Some controllers also send out a trailing pair of "fast stop" (`0x8F`) commands but they may be ignored as other controllers don't send them out at all!

818

In testing with genuine Sensormatic domes and sim58, I have discovered that if a "fastest" command is received and there is no current motion going on, that the dome will execute a flip. So I have added in that feature too.

1.6.3 Master command decode table

877

Command	Use
`0x80`	Unknown three byte command
`0x81`	Pan Left (27^o) until Pan Right or Pan Stop
`0x82`	Pan Right (27^o) until Pan Left or Pan Stop
`0x83`	Stop panning
`0x84`	Tilt Up until Tilt Down or Tilt Stop
`0x85`	Tilt Down until Tilt Up or Tilt Stop
`0x86`	Stop tilting
`0x87`	Focus near until Focus Far or Focus Stop
`0x88`	Focus Far until Focus Near or Focus Stop
`0x89`	Stop Focus
`0x8A`	Zoom In until Zoom Out or Zoom Stop
`0x8B`	Zoom Out until Zoom In or Zoom Stop
`0x8C`	Stop zoom
`0x8D`	Increase Pan and Tilt speeds to $108/44^o$
`0x8E`	Increase Pan and Tilt speeds to $54/19^o$

Continued on the next page.

Continued from the previous page.

Command	Use
0x8F	Stop Fast/Fastest speeds (back to normal 27^o)
0x90	Opens Iris
0x91	Closes Iris
0x92	Stop Iris offset adjustment
0x93	Stop All movement
0x94	Request dome type (poll)
0x95	Request status of alarm inputs
0x96	Undetected unknown command
0x97	ACKnowledge sent to dome, don't do anything.
0x98	Start temp no transmit, no ack
0x99	End temp no transmit, no ack
0x9A	Increase Pan and Tilt speeds to $81/30^o$
0x9B	Stop Faster speeds (back to normal $27/12^o$)
0x9C	Start Boundary Definition.
0x9D	Marks the Current Position as a Boundary
0x9E	Set On Air status to tell the dome to send the
0x9F	Reset On Air status
0xA0	Start defining Pattern 1
0xA1	Start defining Pattern 2
0xA2	Start defining Pattern 3
0xA3	Accept the New Pattern
0xA4	Memory Dump
0xA5	Request Dome Position Coordinates
0xA6	Goto Absolute Position (Multiple-byte format)
0xA7	Undetected unknown command, place holder
0xA8	Mark the current position as Target 1
0xA9	Mark the current position as Target 2
0xAA	Mark the current position as Target 3
0xAB	Mark the current position as Target 4
0xAC	Goto start of Pattern 1 (now its Run Pattern 1)
0xAD	Goto start of Pattern 2 (now its Run Pattern 2)
0xAE	Goto start of Pattern 3 (now its Run Pattern 3)
0xAF	Goto start of Pattern 4 (now its Run Pattern 3)
0xB0	Run Pattern 1
0xB1	Run Pattern 2
0xB2	Run Pattern 3

Continued on the next page.

Continued from the previous page.	
Command	Use
0xB3	Run a newly defined Pattern to review it
0xB4	Go to Target 1
0xB5	Go to Target 2
0xB6	Go to Target 3
0xB7	Go to Target 4
0xB8	Tells the dome to stop recording (defining)
0xB9	Mark the current position as Target 5
0xBA	Mark the current position as Target 6
0xBB	Mark the current position as Target 7
0xBC	Go to Target 5
0xBD	Go to Target 6
0xBE	Go to Target 7
0xBF	Undetected and unknown command, place holder

1.6.3.1 AD2083/02 code translator speeds, pan logic

1014

With the AD2083/02 code translator, there are only eight variable speeds available. These have to be translated into something that is reasonable for the Spectra to use. Unfortunately the data comes out in degrees/sec (?) while Pelco uses a different system to get up to 64 different speeds. So here we have to determine which of the eight speed values are being sent and then translate these values into Spectra equivalents. If a careful examination is made of the values sent from the AD translator, it will be observed that the values are highly non-linear. However Pelco's D protocol uses a different set of non-linear steps. The next code makes the required translations.

```
Observed speeds       Decision point     Generated speeds
      0x04                  0x05              7/0x07
      0x06                  0x08             15/0x0F
      0x0A                  0x0C             23/0x17
      0x0F                  0x13             31/0x1F
      0x18                  0x1C             39/0x27
      0x21                  0x27             47/0x2F
      0x2D                  0x43             55/0x37
      0x5A and other        0x43+            64/0x40 Or turbo speed
```

1037

In the above table the column marked "Decision point" indicates at this value, all lower speeds are translated into the values shown in the "Generated speeds" column. Remember that the lower values are checked first.

panspeed_M is the pan speed to translate and is setup from sproto4_M or is forced to a default value by fixed speed commands, and is modified by the fixed speed enhanced speed buttons.

1104

stickpanin is also called from fixed speed pan commands of `0x81` and `0x82`. At this point w has the new speed value to sent to the Spectra.

1.6.3.2 VM96 variable speeds, pan

1117

It should be noted that Pelco pan speeds start changing at 8. (I.e. all values less than 8 give the same slow pan rate.) This was done for compatibility with older keyboards. There are also several speeds that get "jumped" over to avoid mechanical resonance noise in the individual Spectra its self. (And the skipped numbers change with different rev levels of the Spectra software. I don't really want to mention what happens with the Esprit.) The net result of all this is that we have less than 64 speeds available and it is impossible to predict what will happen in any given situation.

1.6.3.3 AD2083/02 code translator speeds, tilt logic

1156

this_tilt_speed_M is the tilt speed to translate and is setup from sproto4_M or is forced to a default value by fixed speed commands, and is modified by the fixed speed enhanced speed buttons.

Observed speeds	Decision point	Generated speeds
0x03	0x04	3/0x03
0x05	0x07	11/0x0B
0x09	0x0B	19/0x13
0x0D	0x10	27/0x1B
0x14	0x16	35/0x23
0x18	0x1C	43/0x2B
0x21	0x27	51/0x33
0x2D and other	0x27+	63/0x3F Or max tilt spd

1229

sticktiltin is called three ways. One way is from AD type variable speed commands, the second way is from RC216 type variable speed commands and the third way is from fixed speed (RC58 type) commands. w has the speed to use.

1.6.3.4 VM96 variable speeds, tilt

1243

It should be noted that Pelco tilt speeds start changing at 6. (I.e. all values less than 6 give the same slow tilt rate.) This was done for compatibility with older keyboards. There are also several speeds that get "jumped" over to avoid mechanical resonance noise in the individual Spectra its

self. (And the skipped numbers change with different rev levels of the Spectra software.) The net result of all this is that we have less than 64 speeds available and it is impossible to predict what will happen in any given situation/software revision.

The starting value of 6 was increased by 6 more to attempt and match Sensormatic's tilt speeds. Sensormatic has a "rapid" set of pan and tilt speeds compared to those that Pelco uses.

(This routine was changed on 26FEB02.)

1.6.3.5 VM1 fixed speed motion controls, pan and tilt

1273

Fixed speed motion controls.

1.6.3.6 VM1 fixed speed motion controls, "fast"

1291

Fixed speed, speed changing commands.

On the RC58 there are four fixed speeds. These are "normal" which is what you get when the "fingerstick" is depressed to cause motion. Then the system allows for three options to provide quicker motion. These three commands are called "fast", "faster" and "fastest". The more rapid movement speeds are obtained by using the "fingerstick" and one of the three speed increasing keys. On an RC58 these keys are marked "RAIL Right", "Fast" and "RAIL Left". "RAIL Right" gives a times two increase over the basic speed, "Fast" gives a times three increase and "RAIL left" gives a times four increase. (Note that the really old domes, the MiniDome and Cobra Dome, do not respond to the two "RAIL" speed increases but that the SpeedDome does.) Speed increasing commands apply to pan AND tilt simultaneously.

As an unexpected bonus of using the "RAIL" speed increases, the names associated with the button pushes have little relationship with the actual speed increases generated. These speeds were obtained by using sim58 and "SpeedDome 2000 Outdoor", or dome #5, which was bought new for this project.

Thus we have: (The speeds in this table were determined by using a stop watch and then rounding the results.)

Key	Command	Speed	Degrees/Sec	Name
"plain"	...	X1	27	PAN_DEFAULT
RAIL --->	fastest	X2	54	PAN_FASTEST
Fast	faster	X3	81	PAN_FASTER
RAIL <---	fast	X4	108	PAN_FAST
"plain"	...	X1	12	TILT_DEFAULT
RAIL --->	fastest	X2	19	TILT_FASTEST
Fast	faster	X3	30	TILT_FASTER
RAIL <---	fast	X4	44	TILT_FAST

1352

There are two ways to do a flip. The documented one says to send a "fast" command and to follow it with a "fastest" command. In examining what SpeedDomes and DeltaDomes actually do, if the dome is not in motion, just sending a "fastest" command will cause it to flip. So this logic handles both cases.

1539

getdometype sends the TXB-S422's type. We can be either a Speed or Ultra dome, this is most of the diffrence. More diffrences are where we have/do not have long line sizure times. (This routine was changed on 29JAN02.)

1552

Stop command processing. (All eight types.)

1.6.3.7 Variable speed control (VM96)

```
Variable Speed Command contents

Byte  Use
 1    Camera ID
 2    Op Code (0xC0)
 3    Sub-Op Code for left (0x81), right (0x82), up (0x84) or down (0x85)
 4    Speed in deg/sec
 5    Checksum
```

1643

There are several types of keyboards that generate variable speed commands. The one from American Dynamics generates a total of eight (8) different speeds, while others generate more. Thus we have to identify the matrix/keyboard type in order to determine what to do about converting the input speeds into output speeds.

If we ever get in a pan speed of `0x01`, `0x02`, `0x50` or `0x63`, the program logic will assume that it is not talking with an American Dynamics system and that it is talking to a Sensormatic system. Preliminary testing shows that the RC216 sends out the earlier speeds and that the AD2083/02 does not. This process of testing is not guaranteed to work forever but is the best that I can do to tell the difference between command sources. (Of course the RC58 system only sends fixed speed commands with modifiers.)

1.6.3.8 AUX command processing

1706

Format of an output command is that the four LSBs of the opcode indicate what to to with the AUXes. Thus we have:

The Esprit uses aux 1 to turn on and off its windshield wiper. Since neither the Spectra nor the Esprit use more than two auxes, I have redirected aux 4 to aux 1. This is so that a user using a TouchTracker can turn on/off the wiper by just hitting the B key. (The B key "toggles" the

state of aux 4.) Because of this redirecting, TXB-S422 never sends an aux 4 out and since Pelco equipment does not use aux 3 it is not sent out either. It should be noted that the TXB-S422 does not save the state associated with aux 1/4. This state is saved internally in the controller. (This routine was changed on 11FEB02.)

```
Where f = off, N = on, R = redirected to 0.

       AUXes 3 2 1 0         AUXes 3 2 1 0
       0xE0  R - f f         0xE8  R - f f
       0xE1  R - f N         0xE9  R - f N
       0xE2  R - N f         0xEA  R - N f
       0xE3  R - N N         0xEB  R - N N
       0xE4  R - f f         0xEC  R - f f
       0xE5  R - f N         0xED  R - f N
       0xE6  R - N f         0xEE  R - N f
       0xE7  R - N N         0xEF  R - N N
```

1734

To properly process this command we have to send a total of two commands. One each for each bit which gets turned on or off. In protocol D we can only specify one bit to turn on or off with each command. The following table illustrates which command matches each bit:

```
0x09, D_SET_AUXILIARY
0x0A, D_CLEAR_AUXILIARY

0x01 = Aux 1, 0xE1
0x02 = Aux 2, 0xE2
0x03 = Aux 3, 0xE4, Not sent the Spectra/Esprit only have two auxes
0x04 = Aux 1, 0xE1, Toggling aux 1, caused by the controller
```

1.6.3.9 Command processing, timing

1787

The Sensormatic protocol sends new commands almost immediately after it receives an ACK, or it retransmits the last command about 45 ms after starting a command (time information is from measurements made on an AD2083/02). Thus it is necessary to send an ACK, AFTER sending a command to the Spectra. Remember that when sending data to the Spectra we have to bit-bang the data out and thus can not receive any new data in from the head end while sending the Spectra data.

Having a 29.6 ms "dead window" makes it so that we miss many messages. However by delaying the ACK until AFTER sending a command to the Spectra, means that we almost always "get" all commands from the controller.

WARNING: If any controller retransmitts commands faster than the AD2083/02 does there may be a problem. Currently there is about 9 ms of "slack", but different controllers may retransmit faster, etc.

A special case has been detected with the RC58 at Sears, Fresno. On the first bunch of data that I acquired the switch only sent data out once if it got an acknowledge. Since the first time or two at Sears Fresno, the RC58 ALWAYS sends out each command three times. Even if it gets an acknowledge! This causes serious problems with this program's timing. (This is one of the nice "features" of using one companies (American Dynamics) pieces of equipment (AD2083/02) to debug your code with, when actually using another companies (Sensormatic's) equipment. Even though one bought the other out, there may be much older equipment in the field that we have to interface with.) To get around these problems I wrote a simulator (sim58) which runs under GWBASIC and does a very, very simple simulation of what an RC58 does. The primary thing that it does is send predictable commands quickly to a dome. Another advantage is that it is easily changed to do something different on each run.

Program logic used to be to send the ACK first and then do the rest.

1854

Note the test for BYTE3. This test is intended to allow us to receive a message in the middle of bit-banging data out to the Spectra.

1.6.3.10 Other command processing

1870

Whenever we get an On Air message say that we are in boundary #1. (Even if we don't know where we really are. Some controller might be looking for it.)

1884

Unknown commands `0x80`, `0x96`, `0x9C`, `0x9D` and `0xF0` always sends an ACK and then sends a three byte message. The contents of the message are unknown, but I have copied them here from what I saw in the field and what I saw when I sent commands to SensorMatic SpeedDomes and UltraDomes. In the field (Sears Fresno) I used an RC58 controller.

```
Command Generates  Processed by
  0x80 = 0xC4      unknown80
  0x96 = 0xC0      unknown96
  0x9C = 0xC4      boundarystart
  0x9D = 0xB0      boundarymark
  0xF0 = 0xB5      terminatepattern
```

1.7 System Subroutines

1937

System subroutines.

There are several "level"s of subroutines. The level # indicates its level. A subroutine may only call lower level routines, NEVER those at its level or above. As a convention all subroutine names end with their calling level. Subroutines that end with _M are main line extensions.

1.7.1 arrowout_3

1947

arrowout_3 Is used to stick an arrow out in debug mode. (This routine was changed on 29JAN02.)

1.7.2 bin2hex_0

1971

bin2hex_0 is used to take a binary byte in the w register and to convert it into two ASCII bytes. The upper half will be in hexupper_0 and the lower half will be in hexlower_0. (What surprising places to stuck um.) No additional core locations are used.

1.7.3 bitdelay_0

2010

bitdelay_0 is used to sit in a tight loop for one bit time. How long to wait is contained in w on entry.

1.7.4 blank_2

2023

blank_2 is used to output a blank character in debug mode.

1.7.5 blankit_3

2033

blankit_3 is used to clear the Spectra screen.

1.7.6 build_command_0

2047

build_command_0 is used to make a full command to send to the Spectra. It does this by combining various motion control bytes into one D-protocol command.

The following D-protocol bytes are built up:

```
d_command1_M
d_command2_M
d_pan_speed_M
d_tilt_speed_M
```

1.7.7 byteread_1

2073

byteread_1 is used to immediately read a byte and return with it in w. This routine is intended to sandwich reads into the middle of writing to the Spectra. Before calling BYTE1 and BYTE2 must be cleared. The received message is limited to three bytes and is stored in sproto1_M thru sproto3_M. Messages longer than 3 bytes are messed up and not saved correctly. (Had to make this a level 1 subroutine, instead of putting it into sendspectra_0 because I'm worried about not having enough levels of return stack.) A check is made of what might be the first byte to be sure that it matches "our" Spectra's address. If the address check fails, we don't advance past the first byte.

1.7.8 check4rc216_0

2134

check4rc216_0 is used to determine if this is an RC216 type matrix. This is done by examinaning speed control commands and when one is found that is unique to the RC216, then a flag is set. Once the flag, RC216_MODE is set, there is no way to reset it other than by using a full power cycle of the unit.

Unique speeds are 1, 2, 80 and 99 degree/sec.

The RC216, and possiblely others, generate many different pan speed commands, at least 30. So the problem here is to translate the non-linear Sensormatic commands to the non-linear Pelco commands.

In preliminary testing the following variable pan speeds were observed: 1 $\longrightarrow$ 20, 22, 24, 32, 48, 64, 80 and 99. (More have since been found.)

These speeds represent dome speeds in different degrees per second. There is a potential for having 99 different speeds while Pelco only has about 53 different speeds. (It must be remembered that some speeds are repeated inside the Spectra and others are blocked to avoid mechanical resonance induced noise.)

The original logic of determining the speeds to use for the Spectra when controlled by an RC216 was to take advantage of the fact that the Spectra repeats the first nine pan speed values and to then simplify the conversion process quite a lot by always dividing the input value by 2 (the input range is now 0 $\longrightarrow$ 48) and then adding eight to it giving a low Spectra speed of 0.5^o/sec and a high speed of of 41.9^o/deg per second. As this last speed is a little slow (it's NOT turbo speed) we do a special check to see if the input speed is `0x40`, or more, which we translate into a turbo pan speed of `0x40`.

In response to customer comments about the system being "sluggish" the alogrithum has been changed to take the input speed and multiply it by 1/4, 1/2, 3/4 or 1 before adding 8 to it. The rest of the logic has not changed.

```
Called with:
w = speed value to check

Changes:
temp1_0
```

```
Returns with
RC216_MODE turned on if this is an "active" speed. Is not affected
           if speed is "inactive", i.e. this routine does not turn
           off RC216_MODE.
```

1.7.9 checkrate_0

2211

checkrate_0 is used to check bits 1 and 2 on the switch and to modify the input value accordingly.

On input and return w = what to work with/return

Currently January 2002, there has been no demonstrated need for an address switch, so I have decided to use the two leist significant bits of the address switch to provide access to diffrent pan/tilt speeds for use with RC216 type controllers.

The following table holds:

```
   Bit 0 Bit 1 Speed range
     0     0   Full speed, this is slightly faster than Sensormatic pan
               and tilt speeds.

     1     0   3/4 speed, this is somewhat less than Sensormatic pan
               and tilt speeds.

     0     1   1/2 speed, this speed behavies in a "sluggish"
               manner. And is provided for improved control of the
               Spectra.

     1     1   1/4 speed, this is a quite slow speed that is provided
               for precise control of the Spectra.

Calling:
      w has the current speed value

Uses:
      temp1_0
      temp2_0

Return:
      w has a modified, or not, speed value in it
```

1.7.10 ckioerrs_0

2290

ckioerrs_0 is used to check the head end serial connection for errors and to get the data.

On return w holds the data byte. Also temp1_0 holds the most recent byte.

1.7.11 cleareememory_1

2326

cleareememory_1 is used to see if EE memory has been cleared. It is assumed that it is not cleared if the 32 bit constant `0xDEADBEEF` is not in locations EE_SET_MEM0 through EE_SET_MEM3. Anything else will result in three locations being set as follows:

```
Location     Initial value

EE_PRESET     0
```

1.7.12 crlf_2

2392

crlf_2 is used in debug mode to stick out a new-line sequence.

1.7.13 delaynohang_0

2404

delaynohang_0 is used to eliminate hangups while waiting for an IO event to finish. It is expected that this routine will be entered with RAM bank 1 selected, RAM bank 1 will be selected on return. Note that no effort is made to make this fast, after all we want some kind of delay to be done here and the longer the better.

Normally used to see if the transmiter part of the UART is sending a byte. We have to wait untill the preceeding byte is gone before sending a new one.

Timing is as follows: about 5.79 us per call (16 * .361 us)

Uses temp2_0 which should be set to zero before the first call. Is counted up to zero for the delay.

```
On return c = 0 not timed out
          c = 1 has timed out
```

1.7.14 delayspectra_0

2443

delayspectra_0 is used in receiving data from the Spectra. It delays for about one half of a bit time, at 2400 baud, so that sampling may occur in the middle of each bit period.

1.7.15 delayv_0

2459

delayv_0 is used to delay 250 Us * value in w. Each instruction step takes about .361898148 us to complete.

1.7.16 doaux_3

2478

doaux_3 is used to send an AUX command to the Spectra.

```
w = which AUX to configure
d_command2_M = A SET or CLEAR AUX command
```

1.7.17 dochecksum_0

2494

dochecksum_0 is used to sum up the contents of a transmit buffer and stick the results in the command.

1.7.18 eeread_0

2509

eeread_0 is used to read a byte from EEPROM memory.

When called w has the address to read from.

On return w has the addresses contents. (This routine has been copied from the Microchip data sheet for the PIC16F87x, DS30292C, page 43.)

1.7.19 eewrite_0

2533

eewrite_0 is used to write a byte into EEPROM memory.

On entry w has the byte to be written. eewritehere_0, has the address in it to write into.

(This routine has been copied from the Microchip data sheet for the PIC16F87x, DS30292C, page 43.)

eewrite_0 saves the currently written value in temp1_0 and does not change it.

1.7.20 endout_1

2576

endout_1 is used to complete sending a message to the controller. (This routine is new as of 25JAN02.)

1.7.21 findspectraedge_0

2616

findspectraedge_0 is used to detect when a transition occurs in the data from the Spectra at 2400 baud.

Returns with w set to the value of the transition.

```
0 = low going transition
1 = high going transition
```

2626

Note that this is an unusual routine in that it has two different exit points. For each bit, one or zero, there are different exits. Once checking for a transition, the check occurs about once every 1 us or so.

1.7.22 flipinput_0

2670

flipinput_0 is used to alternate the input flipping logic.

1.7.23 fromspectra_1

2684

fromspectra_1 is used to read three bytes from the Spectra. Since there is no UART available to do this, we have to bit-bang to get the message in. Normally this is used to get just the Spectra's address which requires reading only the second byte. It has been modified to now read in three bytes so that the alarm status may be read. In all cases the checksum is ignored.

Now bit-bang to get the answer in. Received data format is:

```
Non return to zero (NRZ)
Least significant bit comes in first
High is a one
Low is a zero
Quiescent is high
RS-422 voltage levels
One start bit which goes low
Eight data bits
One stop bit which goes high or is already high depending on the data
No parity

First byte is a sync byte of all ones.
Second byte is the Spectra's address.
Most of the rest is the software PGM number.
Last byte is the checksum of everything else.
```

1.7.24 getsensorbyte_1

2768

getsensorbyte_1 is used to get in a byte of keyboard data.

This is where the program normally sits while waiting for input from the head end. When an input comes in, then it is processed and we return here.

1.7.25 inlabel_3

2793

inlabel_3 is used to help in overwriting the Spectra's "CONFIGURATION DONE" message with our message on line 2. pe ts "TXB-S422 Rev X.XX AA" ¡message¿ 123456789 1234567890 ¡character position¿ te ps Call with w holding the character to stick into the message. Returns with the message updated for location, a new character stuck in the Spectra buffer and the buffer sent to the Spectra.

1.7.26 presetincrement_1

2815

presetincrement_1 is used to get the next preset value from EEPROM and to update the highest preset number saved in EEPROM.

On return w has the new preset value.

Range of stored presets is 0 $\longrightarrow$ 63, but range of Spectra presets is 1 $\longrightarrow$ 64. So on getting the preset it is always incremented by 1.

Note that it takes 4 $\longrightarrow$ 8 ms to complete an EEPROM write.

1.7.27 printsensorin_4

2852

printsensorin_4 is used to printout the received command, in debug mode. (This routine was changed on 29JAN02.)

1.7.28 printspectraout_4

2921

printspectraout_4 is used to printout the D protocol command, in debug mode. (This routine was changed on 29JAN02.)

1.7.29 send2hex_3

2958

send2hex_3 is used to print out a pair of converted hex digits. Enter with the value to printed out in w.

1.7.30 sendack_5

2973

sendack_5 is used to print out an arrow to mark what are ACKs. (This routine was last updated on 25JAN02.)

1.7.31 sendchecksum_6

2986

sendchecksum_6 is used to complete the checksum, send it, wait a short time and disable the transmitter. (This routine was last updated on 25JAN02.)

1.7.32 senddebugbyte_1

3003

senddebugbyte_1 is used to transmit one byte of information from the w register to the serial output port a single bit at a time. The assumed byte characteristics are: DEBUG_PERIOD baud, one start bit, two stop bits, eight data bits and no parity. (RS-232 is negative true logic with the least significant bit being shifted out first.) It is important to note that this "RS-232" does not go thru a level shifter and thus is exactly what the receiving unit receives. I know that RS-232 is not specified as having voltage levels of 0 and +5, but it does work for short distances and the only time that this is used, is in debugging. Thus it's OK to do it. (This routine was changed on 29JAN02.)

1.7.33 sendmany0_5

3050

sendmany0_5 is used to send several bytes of `0x00` to the head end. How many is in w on entry.

1.7.34 sendone_5

3066

sendone_5 is used to send the byte in w to the head end and to increment the checksum.

1.7.35 sendspectrabyte_0

3077

sendspectrabyte_0 is used to transmit one byte of information from the w register to the serial output port a single bit at a time. The assumed byte characteristics are: 2400 baud, one start bit, two stop bits, eight data bits and no parity. (RS-422 is positive true logic with the least significant bit being shifted out first.)

1.7.36 speed_save_0

3144

speed_save_0 is used to take the current pan and tilt speeds and move them to the two "save" areas.

```
Copies from this_pan_speed_M  into saved_pan_speed_M
and         this_tilt_speed_M into saved_tilt_speed_M
```

1.7.37 startout_1

3161

startout_1 is used to enable the controller output driver and then delay 250 us extra. There is always about 50 to 150 us of overhead that is always there. (And no I don't understand why it varies!) (This routine was updated on 25JAN02.)

1.7.38 threebytemessage_6

3180

threebytemessage_6 is used to send a three byte message to the controller. The address and checksum fields are calculated or supplied by this routine. (This routine was last updated on 25JAN02.)

On entry w has the value to send

1.7.39 tosensormatic_4

3211

tosensormatic_4 is used to send bytes to the Sensormatic controller. (This routine was changed on 29JAN02.)

```
w = what to send
```

1.7.40 tospectra_2

3255

tospectra_2 is used to send a D protocol command to a Spectra.

A special problem is fixed with this routine. That problem is that it takes so long to send a Spectra command in D protocol at 2400 baud (29.6 ms) that we miss commands from the Sensormatic controller. (Remember that there is only one UART on this baby and that we have to bit-bang data out to the Spectra and can't do anything else or the timing gets messed up.) The solution to this is to check the receive register after each transmitted byte and to store the data if any data comes in. If BYTE3 is set then all three bytes have been received and we can exit.

Conveniently enough the UART logic on the PIC chip is somewhat more than double buffered, i.e. there are two holding registers (in a FIFO) arrangement in addition to the input shift register. Since the data comes in at 4800 baud and we send data out to the Spectra at 2400 baud, if we check for received data after each sent byte, we should never "loose" a three byte message.

Automatically supplied fields are:

```
d_sync_M        0xFF, sync byte
d_address_M     spectraaddress_M
d_checksum_M    checksum
```

3280

Fields to be provided by the caller:

```
d_command1_M    Command 1
d_command2_M    Command 2
d_pan_speed_M   Data 1
d_tilt_speed_M  Data 2
```

1.8 System initialization

1.8.1 Step 1 of initialization

3374

Initialize IO ports, etc.

3399

In this program the standard RAM bank is bank 0. There will be some changes to RAM bank 1, but these will be rare and when this happens the is marked in the right hand column with the current RAM bank #.

Do much of RAM bank 1 setup first, then select RAM bank 0 for the rest of setting up.

```
porta is first              It is only a 6 bit port, ignore upper bits     1
                0           7 x (IO pin does not exist)                    1
                 0          6 x (IO pin does not exist)                    1
                  1         5 Serial data from Spectra                     1
                   0        4 spare                                        1
                                                                           1
                    0       3 Serial data to Spectra                       1
                     0      2 Used to invert input/output data             1
                      0     1 RS-422/485 input enable                      1
                       0    0 spare                                        1
```

```
portb is second                                                          1
                0            7 ICSP and spare otherwise                  1
                 0           6 ICSP and debug output                     1
                  1          5 Debug enable                              1
                   1         4 Permit using address 64                   1
                                                                         1
                    1        3 Address bit 3                             1
                     1       2 Address bit 2                             1
                      1      1 Address bit 1                             1
                       1     0 Address bit 0                             1


portc is last                                                            1
                1            7 Input data from the controller via the UART 1
                 0           6 Output data to the head end, via the USART  1
                  0          5 RS-422/485 output enable                  1
                   0         4 spare                                     1
                                                                         1
                    0        3 spare                                     1
                     0       2 spare                                     1
                      0      1 spare                                     1
                       0     0 spare                                     1


Set timer to instruction counter with no prescale, page 31               1
                1            7 Disable portb pull-ups                    1
                 0           6 Interrupt on falling edge                 1
                  0          5 tmr0 clock source is instruction cycle clock 1
                   0         4 pos transition                            1
                    0        3 Prescaler to Timer0                       1
                     000    012 Prescaler is set to 1:2                  1


Peripheral Interrupt Register 1                                          0
      ---       0            7 Reserved                                  0
      ---        0           6 Reserved                                  0
      RCIF       0           5 Receive Interrupt flag                    0
      TXIF        0          4 Transmit Interrupt flag                   0
      SSPIF       0          3 Synchronous Serial Port Interrupt Flag    0
      CCP1IF       0         2 Capture flag                              0
      TMR2IF        0        1 TMR2 flag                                 0
      TMR1IF         0       0 TMR1 flag                                 0
```

```
Setup timer 1                                                          0
      unused    0            7 Unimplemented                         0
      unused     0           6 Unimplemented                         0
      T1CKPS1     1          5 Timer 1 input clock prescale select bit 0
      T1CKPS0      1         4 . . prescale by 8                     0
      T1OSCEN       0        3 Oscillator on/off with off selected   0
      T1SYNC         0       2 Ignored if TMR1CS = 0                 0
      TMR1CS          0      1 Use internal/external clock, internal used 0
      TMR1ON           0     0 Start/stop timer, 1 = start, 0 = stop 0


Setup the USART, transmit first, page 99                               0
      CSRC      0            7 Synchronous mode clock select, async = nop 0
      TX9        0           6 0 = 8 bit transmit mode, 1 = 9        0
      TXEN        1          5 Transmit enable = 1, set to 1 to start xmit 0
      SYNC         0         4 Asynchronous mode = 0, sync = 1       0
      ---           0        3 Spare, unimplemented                  0
      BRGH           0       2 Low speed mode for the baud rate generator 0
      TRMT            0      1 Status of xmit reg                    0
      TX9D             0     0 9th bit xmitted, spare in 8 bit mode  0


Setup the USART, now the receive stuff, page 100                       0
      SPEN      1            7 Serial port enable                    0
      RX9        0           6 8 bit receive mode = 0, 1 = 9         0
      SREN        0          5 Sync mode receive bit, async = nop    0
      CREN         1         4 Asynchronous enable continuous receive 0
      ---           0        3 Spare                                 0
      FERR           0       2 Framing status error bit              0
      OERR            0      1 Overrun status error bit              0
      RX9D             0     0 Parity bit, not used in 8 bit mode    0
```

3550

At this point all IO is properly set up.

1.8.2 Step 2 of initialization

3557

Now get the Spectra's address. This is done by sending a query command out to the Spectra. Then waiting until the first byte comes in. The first byte is thrown away and the second byte is masked to seven active bits and is used as the Spectra's address.

3564

1. First wait until the Spectra is working.

3576

2. Now send a query command so as to get the Spectra's address.

3608

3. Display the software rev and current address switch settings.

3624

4. Identify ourselves on the second line.

3647

5. Stick out the word "Rev ".

3660

6. If bit 6 of the switch is set then say that this is debug mode. (This routine was changed on 29JAN02.)

3694

7. A rev level of BETA indicates that this is a trial version.

3712

8. A rev level of TEST indicates that this is a test version.

3728

9. Now get the block address that is loaded into the switch.

3737

10. Done labeling the Spectra.

11. Select normal mode inputs and disable comm outputs

(This routine was changed on 29JAN02.)

3759

Read in a controller message.

Moved to the end so that a computed goto will work. With this where it would most logically go, the computed goto for input command processing will be too high in memory.

1.9 Multibyte responses

3948

Multibyte responses are sent from here.

There are three "long" responses to deal with. They are:

```
software version        10 byte response
preset position         12 byte response
memory dump             104 byte response
```

1.9.1 getposition

3961

getposition is used to provide a fake value to the controller as to where the Spectra is pointing. There is a basic problem with Sensormatic vs. Pelco in the way in which presets are handled. The problem is that on the newer domes, Sensormatic sends a command to the dome that asks "where are you pointing". The dome then tells the controller where it is pointing, etc., and the controller does the remembering. When the operator wants to go to, say, preset 5, the controller has to send all pointing information to the dome. Pelco's approach requires that the dome remember where it is when it is told to "set preset 5" and then the head end just sends "goto preset 5" to the dome.

In order to fake out the Sensormatic system, we generate a fake reply message as to where we are pointing. The fake message is generated by using an old valid reply message from dome #5 and changing some of the fields so that the changed fields indicate a usable preset number.

The basic message from dome #5, a SpeedDome 2000, is:

```
0x01 0x00 0x00 0x12 0x4C 0x26 0x5D 0x28 0x37 0x31 0x37 0x57
```

3984

Translated this becomes:

```
Iris xunning    = 0x00       (Haven't yet figured out that first letter).
Zoom limit      = 0x00
Tilt Position   = 0x12 0x4C
Zoom Position   = 0x26 0x5D
Electronic Zoom = 0x28 0x37
Pan Position    = 0x31 0x37
```

3995

After the ACK the response is: (All hex fields are common to all position requests.)

Byte	Use	Contents
1	Camera ID	Camera ID
2	Iris Data	0x00
3	Zoom Limit	0x00
4,5	Tilt Position	0x12, Preset ID
6,7	Zoom Position	0x26, Preset ID
8,9	Electronic Zoom	0x28, Preset ID
10,11	Pan Position	0x31, Preset ID
12	Checksum	Checksum

4010

Values that get stuck in the reply message consist of the preset ID and various constants from a real breakout dump from dome #5. If there are any "strange" value range checks made, this should confuse them and get the controller to pass the values as OK. (This routine was last updated on 25JAN02.)

4088

Set up for a timer to run, sos I will not increment the preset ID during the next one or two requests for my position.

1.9.2 version

4100

This is the reply made by dome #5. (This routine was last updated on 25JAN02.)

```
Version Response is:

 Byte  Use                    Contents
  1    Camera ID              Camera ID
  2    Op Code                0xC9
  3    Unknown byte           0x06
4 --> 9 Software rev          0x07, 0x01, 0x00, 0x01, 0x03, 0x16
  10   Checksum               Checksum
```

1.9.3 gotoposition

4143

Input goto command is: (Only bytes 1, 2, 4 and 13 are used.)

```
 Byte  Use                    Contents

  1    Camera ID              Camera ID
  2    Op Code                0xA6
 3,4   Pan Position           0x33, Preset ID only preset ID used
 5,6   Tilt Position          0x06, Preset ID . . is in byte 4
 7,8   Zoom Position          0x26, Preset ID
 9,10  Digital Zoom           0x28, Preset ID
  11   Iris Offset            0x00
  12   Zoom Limit             0x00
  13   Checksum               Checksum
```

1.9.4 memorydump

4172

Reply to a memory dump request (`0xA4`).

The format of the reply is copied from dome #5, a SpeedDome 2000, immediately following a power up. I.e. a "clean" dome with no motion commands, or anything else, loaded into it. (This routine was last updated on 25JAN02.)

The message's content was: (The first byte is the address of the dome and the last byte is the checksum.)

```
   1    2    3    4    5    6    7    8    9   10
1-01 1-66 1-0E 1-80 1-00 1-00 1-00 1-00 1-00 1-00
  11   12   13   14   15   16   17   18   19   20
1-00 1-00 1-72 1-00 1-00 1-00 1-00 1-00 1-00 1-00
  21   22   23   24   25   26   27   28   29   30
1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
  31   32   33   34   35   36   37   38   39   40
1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
  41   42   43   44   45   46   47   48   49   50
1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
  51   52   53   54   55   56   57   58   59   60
1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-0E 1-00
  61   62   63   64   65   66   67   68   69   70
1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
  71   72   73   74   75   76   77   78   79   80
1-DF 1-FF 1-22 1-0E 1-00 1-00 1-00 1-00 1-00 1-00
  81   82   83   84   85   86   87   88   89   90
1-00 1-00 1-00 1-00 1-00 1-DF 1-FF 1-21 1-0E 1-00
  91   92   93   94   95   96   97   98   99  100
1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
 101  102  103  104
1-DF 1-FF 1-20 1-72   < 1: Dome memory dump msg length = 104>
```

1.10 Outline of operation of TXB-S422 Translator

4264

When initialized the TXB-S422 Translator, sets up some IO configuration information. It then reads the DIP switch and obtains configuration information to use and saves it away. It then sends a query command to the Spectra, waits for its reply and uses the resulting address information to determine what address the Spectra is set to. The TXB-S422 then places its ID message on the Spectra's screen and enters an endless loop looking for messages from the controller/head-end. The endless loop is contained in the getsensorbyte_1 routine. The endless loop is used to receive

commands in and translate them into commands for the Spectra. Translated commands are sent to the Spectra in D-Protocol at 2400 baud and we go back for some more.

Information about the applicable part numbers for this project is at the end of this program listing as is a change log.

1.11 Change log

4282

Various phrasing of the word "beta" are used to differentiate different test versions of this code. The following types of "beta" have been used:

```
 1       2       3       4       5       6       7       8       9
"BETA", "Beta", "BEta", "BETa", "bETA", "beTA", "beTx", "betA", "beta"

 10      11
"beTa", "bEta"

beta's not yet used:
BEtA BeTA BeTa BetA bETa bEtA
```

4297

As additional "beta test" versions of the code are needed, be sure to use a different spelling of "beta" and update the above table. The various spellings of "beta" are somewhere beteemn line numbers 3700 and 3900 of this listing. "There is a reference tag of 'r44' located there."

Prior to rev "beTx", I thought that I could remember all the diffrences between the diffrent beta-revs. Time has proved me wrong, so I'm writing down the various things that I think have been fixed. (I have also written down what I have changed and added too.)

4310

1. "beTx" is a special version of the TXBS422 code that has enabled switch position 3 to control one of two versions of pan/tilt speeds. If the switch is off then the option 1 for speeds is used, if it is on the option 2 is used. The use of two options for speed calculations is being done in an attempt to make the Spectra behave more like an UltraDome in its operator response.

   Speed option 1: Input Pan speed is multiplied by 3/4, has 8 added to it and is limit checked at 63, anything more than 63 is turned into 64 for turbo mode. Input Tilt speed is multiplied by 1/2 and has 6 added to it, it is then limit checked at 63, nothing greater than 63 is permitted for tilt speeds.

   Speed option 2: Input Pan speed has 8 added to it and is limit checked at 63, anything more than 63 is turned into 64 for turbo mode. Input Tilt speed has 6 added to it, is then limit checked at 63, nothing greater than 63 is permitted for tilt speeds.

2. "betA" changes the speeds once again and now bits 0 and 1 of the address switch are used to select the divisor values for pan and tilt speeds. 0 = none, 1 = 3/4, 2 = 1/2 and 3 = 1/4.

(I did this in response to a conversation I had with a potential customer.) Logic for adding 8 or 6 and range limit checking has not changed.

3. "beta" makes it so that when going from normal speeds, to fast speeds and then back to normal speeds, that the unit dosen't stop movment. (Actually it didn't stop, it just went verrrry sloooowly. The slow speed was caused by sending a speed command of "0" to the Spectra.)

   An additional use has been found for the switch. And that is to provide for either SpeedDome or UltraDome timing/modes of operation. The primary diffrence is with the timing of output signals. If SW-3 is off, which is the default, then all messages to the controller assert the communications line for about 250 us before sending the first bit and for about 1 ms after the last bit. This is about what a SpeedDome does and the reply byte for a dome type query request (poll) is set to `0xF8`, the SpeedDome's response. If SW-3 is on, then there are almost no assertion delays before and after sending something to the controller, which is what an UltraDome/DeltaDome does, and the reply for a dome type query request (poll) is set to `0xF5`. It should be noted that these two changes are the only diffrence between SpeedDome and UltraDome modes of operation. I.e. there are no "real" changes made to the TXB-S422's operation nor are there any additions made.

   Some time or other SW-7 will be used to terminate/unterminate the communications line from the controller. And SW-8 will be used to terminate/unterminate the communications line going to the controller. (Both of these are hardware changes that can not be monitored, or made from software.)

   Currently only SW-4 is unused.

   It should be noted that all bit positions are "live", i.e. they may be changed at any time "on the fly" and not just prior to power up time. With the Spectra it is difficult to change the switch positions with the Spectra installed in its back-box, however it is possiable to change the switch positions on an Esprit type unit. (Or if you have a special extension cable to use with a Spectra.)

4. In the "beTa" rev logic was changed to correctly interpert variable speed commands from an RC216.

   Also aux 4 becomes redircted to aux 1 for wiper motor control.

   And the "trial pattern run" command was honored. (Previously it was ignored.)

5. In rev "bEta" the tilt speed was fixed in RC216 mode. (Found a place where I used a "movf" instead of a "movwf" to save the tilt speed value.

   In the logic for identifying an RC216 type controller, speed 2 was not being decoded as a trigger.

   In an attempt to speed up tilt speeds in RC216 mode, I added an additional offset of 6 to the input value. (All speeds get incremented by 6 to get over the many slow starting tilt values. This makes it 12 to force speeds further up the decode table.)

Stopped clearing saved pan/tilt status (i.e. motion controls) on getting a "fast" type stop command. Now multiple fast/faster/fastest commands may be used.

Started to clear both d_command1_M and d_command2_M before decoding commands. Now I hope to not get both right and left pan bits on at the same time.

Fixed many little problems that turned up as I fixed other problems. (Some of these new ones were caused by the previous fixes, etc.)

Finally gave up and completly redid the motion control logic.

1.12 Part numbers

4406

The following Pelco part numbers apply to this project. Each time a software rev is made, the rev number on the IC51, PG51, BH51 and FW00 parts, must be updated.

This listing must be updated EACH TIME before releasing the code.

```
PA11-0006-00x0 Is the fully assembled board with many other things on it.
IC51-0029-0100 The programmed IC that gets stuck on the board.
IC01-1928-0876 PIC 16F876 chip that gets stuff written into.
PG51-0042-0100 All combined binary hex files. There is only one .HEX
               file on this project.
BH51-0022-0100 Single binary hex file in .HEX format
FW00-0150-0100 Source code for this project. Consists of four files:
               The main source file, TXBS422.ASM
               MicroChip's chip include file, P16C876.INC
optional       A file of Sensormatic protocol definitions, SDEFS.INC
               A file of Pelco protocol definitions, DPROTO.INC
```

4426

To operate correctly the crystal running this chip must be cut for 11.0592 MHz

```
Instruction Memory Banks

Bank    Start     End
  0     0x0000   0x07FF
  1     0x0800   0x0FFF
```

APPENDIX A

A TXBS422 Assembly Listing

```
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE  1


LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00001 ;;latex\rcsdocrev{\$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilto
                            n $}
                      00002 ;;latex\centerline{\Huge\bf TXB-S422}
                      00003 ;;latex\bigskip\bigskip
                      00004 ;;latex\centerline{\large\bf\today}
                      00005 ;;latex\bigskip\bigskip
                      00006 ;;latex\tableofcontents
                      00007 ;;latex\forcerightpage
                      00008
                      00009 ;;latex\section{\indexit{TXB-S422} Source code description}
                      00010  title "$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $"
                      00011  subtitle "Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422"
                      00012
                      00013 ;;latex\subsection{Global Defines}
                      00014 ;;ps
                      00015 ; If DEBUG isn't specified and real debugging is being done, then it is
                      00016 ; very unlikely that the system will boot up properly. During a normal
                      00017 ; boot up sequence this software looks at the Spectra and expects several
                      00018 ; events to happen in sequence. If the sequence of initial events is
                      00019 ; improper, then this software will hang-up, mostly waiting for a reply
                      00020 ; from the Spectra with the Spectra's address in it. As a solution to
                      00021 ; this problem, when you are in DEBUG mode it is possible to specify an
                      00022 ; address for the Spectra. This gets around the whole boot up problem. If
                      00023 ; you are not running in DEBUG mode, then each time that the software
                      00024 ; gets changed there must be a complete power cycle of the Spectra done
                      00025 ; with careful starting of the TXB-S422's software. (Not a fun activity!)
                      00026 ;;pe
                      00027 ;;ts
                      00028 ;#define DEBUG
                      00029 ;;te
                      00030         ifdef DEBUG
                      00031
                      00032 ;;ps
                      00033 ; Make it so that when debugging it is possible to avoid asking the
                      00034 ; Spectra its address. This specifies what address to use.
                      00035 ;;pe
                      00036 ;;ts
                      00037 #define DEBUG_ADDRESS   44
                      00038 ;;te
                      00039
                      00040 ;;ps
                      00041 ; TEST_PROPORTIONAL is used to enable testing of proportional speed
                      00042 ; commands from an RC216 (VM96)controller when we only have an AD2083/02.
                      00043 ; (The RC216 sends different speed values than the AD2083/02 does. This
                      00044 ; forces a test for controller type to always report than an RC216 is
                      00045 ; installed when any variable speed pan command comes in.)
                      00046 ;;pe
                      00047 ;;ts
                      00048 ;#define TEST_PROPORTIONAL
                      00049 ;;te
                      00050
                      00051         endif ; ifdef DEBUG
                      00052
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE  2
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00053 ;;ps
                      00054 ; ALLOW_INPUT_FLIP is used to enable/disable the input data flipping
                      00055 ; logic. If defined then flipping is enabled.
                      00056 ;;pe
                      00057 ;;ts
                      00058 ;#define ALLOW_INPUT_FLIP
                      00059 ;;te
                      00060
                      00061
```

```
                     00062 ;;ps
                     00063 ; LIST_INCLUDES when defined, makes all included files list themselves out.
                     00064 ;;pe
                     00065 ;;ts
                     00066 ;#define LIST_INCLUDES
                     00067 ;;te
                     00068
                     00069 ;;ps
                     00070 ; Change the default radix from hex to decimal so that this code may be
                     00071 ; written in a civilized manner. (Either enable it here on in the project
                     00072 ; file.)
                     00073 ;;pe
                     00074 ;;ts
                     00075 ;       radix dec
                     00076 ;;te
                     00077
                     00078 ;;ps
                     00079 ; errorlevel -302 is used to get rid of "Register in operand not in bank
                     00080 ; 0.  Ensure that bank bits are correct." messages which gets a little
                     00081 ; old after awhile.
                     00082 ;;pe
                     00083 ;;ts
                     00084         errorlevel -302
                     00085 ;;te
                     00086
                     00087 ;;ts
                     00088         ; p = processor type, c = columns/page, n = lines/page,
                     00089         ; st = no symbol table
                     00090         ; Pick whatever type of processor you wish from this list
                     00091 ;        list    p=pic16f873,  c=132, n=58, st=off
                     00092         list    p=pic16f876,  c=132, n=58, st=off
                     00093         ;
                     00094         ; Options for the 16F87x series of CPUs are:
                     00095         ;
                     00096         ; Brown out enable:              _boden_off   _boden_on
                     00097         ; Code Protection:               _cp_all      _cp_half
                     00098         ;                                _cp_off      _cp_upper_256
                     00099         ; Code Protect Data EEPROM:      _cpd_off     _cpd_on
                     00100         ; Debug enable:                  _debug_off   _debug_on
                     00101         ; Flash write enable:            _wrt_enable_off
                     00102         ;                                _wrt_enable_on
                     00103         ; Low voltage programing enable: _lvp_off     _lvp_on
                     00104         ; Oscillator type:               _hs_osc      _lp_osc
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE  3
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00105         ;                                _rc_osc      _xt_osc
                     00106         ; Power up timer enable:         _pwrte_off   _pwrte_on
                     00107         ; Watch dog timer enable:        _wdt_off     _wdt_on
                     00108         ;
                     00109         ; The only difference between "debug" and "normal" mode configuration
                     00110         ; settings is that in "debug" mode I do not enable code protection.
                     00111         ; Code protection is only enabled in released code. MicroChip does
                     00112         ; not recommend enabling it on EPROM type devices.
                     00113         ;
                     00114         ifdef DEBUG
                     00115          __config _boden_on&_hs_osc&_pwrte_on&_wdt_off&_cp_off&_wrt_enable_off
                     00116         endif ; ifdef DEBUG
                     00117
                     00118         ifndef DEBUG
                     00119 ;         __config _boden_on&_hs_osc&_pwrte_on&_wdt_off&_cp_off&_wrt_enable_off
2007   0DC2          00120          __config _boden_on&_hs_osc&_pwrte_on&_wdt_off&_cp_all&_wrt_enable_off
                     00121         endif ; ifndef DEBUG
                     00122 ;;te
                     00123
                     00001         list
                     00002 ; P16F876.INC  Standard Header File, Version 1.00    Microchip Technology, Inc.
                     00003 ; $Header: d:/sears/RCS/p16f876.inc,v 1.5 2001-12-03 13:32:29-08 Hamilton Exp Hamilton $
                     00132         list


                     00133         space 2
                     00134         include <dproto.inc>    ; Pelco D protocol related definitions
                     00001 ; "$Header: d:/sears/RCS/dproto.inc,v 1.23 2002-02-28 12:57:21-08 Hamilton Exp Hamilton $"
                     00002 ; Copyright by Pelco, 2000, 2001, 2002
                     00068         list
                     00069 ; End of PROTOCOL.INC
                     00070 ;
                     00135 ;;latex\subsection{Include files}
```

```
                     00136 ;;latex\subsubsection{\tt dproto.inc}
                     00137 ;;latex\sloppy\listing{dproto.inc}
                     00138 ;;latex\fussy\bigskip


                     00139         space 2
                     00140         include <sdefs.inc>     ; Sensormatic protocol related definitions
                     00001 ; "$Header: d:/sears/RCS/sdefs.inc,v 1.17 2002-02-27 15:58:31-08 Hamilton Exp Hamilton $"
                     00002 ; Definitions to use with Sensormatic's RS422 protocol
                     00003 ; Copyright by Pelco, 2001, 2002
                     00133         list
                     00134 ; End of SDEFS.INC
                     00135
                     00141 ;;latex\subsubsection{\tt sdefs.inc}
                     00142 ;;latex\sloppy\listing{sdefs.inc}
                     00143 ;;latex\fussy\bigskip


                     00144         space 2
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  4
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00145 ;;ps
                     00146 ; All defines which get used to "org" memory locations are located here.
                     00147 ; Note that PROGRAM_PAGE0 starts several locations AFTER the first core
                     00148 ; location where it may.  This is so that if needed, an OTP type chip may
                     00149 ; be reprogrammed with some carefully written code.  See the MicroChip
                     00150 ; manual "PICmicro Mid-Range MCR Family Reference Manual", DS33023A,
                     00151 ; December 1997, for information on how to do this.
                     00152 ;;pe
                     00153 ;;ts
                     00154 #define RESET_VECTOR    0x0000  ; Reset vector entrance
                     00155 #define PROGRAM_PAGE0   0x0010  ; Program area 0
                     00156 #define PROGRAM_PAGE1   0x0800  ; Program area 1
                     00157 ;;te
                     00158
                     00159
                     00160 ;;ps
                     00161 ; RAM bank starting locations are defined here. There are some bad
                     00162 ; "holes" in each bank. I have listed those that I know of at the start
                     00163 ; of allocating usage for each bank.
                     00164 ;;pe
                     00165 ;;ts
                     00166 #define RAM0            0x020   ; RAM Bank 0
                     00167 #define RAM1            0x0A0   ; RAM Bank 1
                     00168 #define RAM2            0x120   ; RAM Bank 2
                     00169 #define RAM3            0x1A0   ; RAM Bank 3
                     00170 ;;te
                     00171
                     00172 ;;ps
                     00173 ; Internal EEPROM defines.
                     00174 ;
                     00175 ; This memory must be initialized before use. This is done transparently
                     00176 ; to the user by setting a flag in four consecutive bytes. If the flag is set
                     00177 ; then there is no initialization is needed, but if it is missing then the
                     00178 ; preset ID is set to 0x00.
                     00179 ;
                     00180 ; When EEPROM has been initialized the four EE_SETMEMx bytes will have
                     00181 ; 0xDEADBEEF written into them, i.e. EE_PROM0 = 0xDE and EE_PROM3 = 0xEF
                     00182 ;;pe
                     00183 ;;ts
                     00184 #define EE_PRESET        0x22   ; Handy place in EEPROM to use for presets IDs
                     00185 #define EE_SET_MEM0      0x40   ; EEPROM flag for being initialized, part 1 0xDE
                     00186 #define EE_SET_MEM1      0x41   ; EEPROM flag for being initialized, part 2 0xAD
                     00187 #define EE_SET_MEM2      0x42   ; EEPROM flag for being initialized, part 3 0xBE
                     00188 #define EE_SET_MEM3      0x43   ; EEPROM flag for being initialized, part 4 0xEF
                     00189 ;;te
                     00190
                     00191
                     00192 ;;ps
                     00193 ; Pan and Tilt Speed defines. It is important to note that the "names" of
                     00194 ; the various speeds do not match what the speeds actually go at. These
                     00195 ; speeds have been verified with the VM1 (RC58) system at Sears Fresno.
                     00196 ; The RC58 keyboard is a fixed speed unit. To get more than one speed
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  5
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
                     00197 ; from it requires that one of three "speed enhancement" keys be
                     00198 ; depressed while a dome is in motion. The speed enhancement keys are
                     00199 ; marked "RAIL --->" (fastest), "Fast" (faster) and "<--- RAIL" (fast).
                     00200 ; These three keys generate the speed commands indicated in parenthesis
                     00201 ; following their names. "fastest" gives an X2 speed increase, "faster"
                     00202 ; gives an X3 speed increase, and "fast" gives an X4 speed increase. (Not
                     00203 ; exactly what any sane person would choose for names vs. speeds!) (And
                     00204 ; then the speed changes that the Spectra makes is non-linear, and has
                     00205 ; intentional "holes" for acoustic noise reduction, when compared to the
                     00206 ; values sent to it.)
                     00207 ;;pe
                     00208 ;;ts
                     00209 ;                          value   is  should be
                     00210 #define PAN_DEFAULT        51   ; 26   27/sec
                     00211 #define PAN_FASTEST       59   ; 55   54/sec
                     00212 #define PAN_FASTER        63   ; 80   81/sec
                     00213 #define PAN_FAST          64   ; 150 108/sec
                     00214
                     00215 ;                          value  gives
                     00216 #define TILT_DEFAULT      36   ; 12/sec
                     00217 #define TILT_FASTEST      48   ; 19/sec
                     00218 #define TILT_FASTER       57   ; 30/sec
                     00219 #define TILT_FAST         63   ; 44/sec
                     00220 ;;te
                     00221
                     00222 ;;ps
                     00223 ; Misc defines.
                     00224 ;
                     00225 ; Internal D-protocol defines are first.
                     00226 ;;pe
                     00227 ;;ts
                     00228 #define PAN_RIGHT      this_command2_M,bit1
                     00229 #define PAN_LEFT       this_command2_M,bit2
                     00230 #define TILT_UP        this_command2_M,bit3
                     00231 #define TILT_DOWN      this_command2_M,bit4
                     00232 #define ZOOM_IN        this_command2_M,bit5
                     00233 #define ZOOM_OUT       this_command2_M,bit6
                     00234 #define FOCUS_NEAR     this_command2_M,bit7
                     00235
                     00236 #define FOCUS_FAR      this_command1_M,bit0
                     00237 #define IRIS_OPEN      this_command1_M,bit1
                     00238 #define IRIS_CLOSE     this_command1_M,bit2
                     00239 ;;te
                     00240
                     00241 ;;ps
                     00242 ; General use misc defines.
                     00243 ;;pe
                     00244 ;;ts
                     00245 #define BAD_TRY            5   ; Number of bad inputs before flipping
                     00246 #define BLANK           0x20   ; Blank character
                     00247 #define CR              0x0D   ; Carriage return
                     00248 ;#define IDLE_LOOKS      216   ; How many times to look for an idle
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  6
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00249 ;                            ; IDLE_LOOKS must be a multiple of 9
                     00250 #define IGNORE_ADDRESS     64   ; Messages to address 64 are ignored
                     00251 #define LF              0x0A   ; Line feed
                     00252 #define LONG_TIME_OUT    continuationtimer_M,bit1 ; End point of a long timeout
                     00253 #define PELCO_SPEED_DOME 0xF8   ; Dome type to send to the controller.
                     00254 #define PELCO_ULTRA_DOME 0xF5   ; Dome type to send to the controller.
                     00255                              ; SpeedDomes send an "E" type of response
                     00256                              ; until they get five poll commands and then
                     00257                              ; send an "F" response.
                     00258                              ; 0xE5 = SpeedDome 2000 2-board response dome 2
                     00259                              ; 0xF5 = SpeedDome Ultra response        dome 4
                     00260                              ; 0xF8 = SpeedDome 2000 1-board response dome 5
                     00261                              ; 0xE8 = SpeedDome 2000 1-board response dome 1
                     00262 #define S_GOTO_POSITION_SIZE 13 ; 0xA6 Command length
                     00263 #define S_PROP_SPEED_SIZE     4 ; 0xC3 Command length
                     00264 #define S_PROTO_LENGTH        0 ; Fake length of an S protocol command
                     00265 #define S_UNKNOWN_C1_SIZE    12 ; 0xC1 Command length
                     00266 #define S_VARIABLE_SPEED_SIZE 5 ; 0xC0 Command length
                     00267 ;;te
                     00268
                     00269 ;;ps
                     00270 ; IO port definations.
                     00271 ;;pe
```

```
                    00272 ;;ts
                    00273 #define spare2          porta,bit0      ;  2 spare spare spare
                    00274 #define ENABLE_INPUTS   porta,bit1      ;  3 RS-422/485 input enable
                    00275                                           ;    Control inputs from the head-end
                    00276                                           ;    0 enables inputs
                    00277                                           ;    1 disables inputs
                    00278 #define INVERT_IO       porta,bit2      ;  4 Used to invert input/output data
                    00279                                           ;    0 = normal input levels
                    00280                                           ;    1 = input data is inverted
                    00281                                           ;  Controls an input chips inversion ctl
                    00282                                           ;  INVERT_IO Input Output Format
                    00283                                           ;      0       1     1    normal
                    00284                                           ;      0       0     0    normal
                    00285                                           ;      1       1     0    inverted
                    00286                                           ;      1       0     1    inverted
                    00287 #define SPECTRA_OUT     porta,bit3      ;  5 Serial data to Spectra
                    00288 #define spare6          porta,bit4      ;  6 spare spare spare
                    00289 #define SPECTRA_IN      porta,bit5      ;  7 Serial data from Spectra
                    00290 #define notexist1       porta,bit6      ;  - IO pin does not exist
                    00291 #define notexist2       porta,bit7      ;  - IO pin does not exist
                    00292 ;;te
                    00293 ;;ps
                    00294 ; It is important to always remember that the address switch bits come in
                    00295 ; "backwards", i.e. if the switch is ON then it comes in as a zero (0)
                    00296 ; and if it is OFF then it comes in as a one (1).
                    00297 ;;pe
                    00298 ;;ts
                    00299 #define SPEEDCONTROL0   portb,bit0      ; 21 Speed changer
                    00300 #define SPEEDCONTROL1   portb,bit1      ; 22 Speed changer
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  7
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00301                                           ;    Bits 0 and 1 give slower and
                    00302                                           ;    slower speeds, the more that are
                    00303                                           ;    set.
                    00304                                           ; Additional logic REQUIRES that these
                    00305                                           ; two bits be together and in positions
                    00306                                           ; 0 and 1.
                    00307 #define SPEEDMASK       0x03            ; Mask for just speed. (Use this to
                    00308                                           ; find the logic for implementing
                    00309                                           ; this option.) This speed option only
                    00310                                           ; affects RC216 mode speed commands.
                    00311 #define NODELAY         portb,bit2      ; 23 Disable or enable delays before
                    00312                                           ; and after sending data. Normal
                    00313                                           ; is to have a 250 us delay before
                    00314                                           ; the start of the first bit and
                    00315                                           ; 1 ms after the last bit. These
                    00316                                           ; delays are to enable the RS422
                    00317                                           ; drivers to stabalize their
                    00318                                           ; outputs and to allow others to
                    00319                                           ; use the channel.
                    00320                                           ;    0 enables normal before and
                    00321                                           ;      after delays
                    00322                                           ;    1 disables starting and ending
                    00323                                           ;      delays
                    00324 #define spare24         portb,bit3      ; 24 Only unused bit of address switch
                    00325 #define PERMIT64        portb,bit4      ; 25 Permit address 64 to be used
                    00326 #define DEBUG_MODE_ON   portb,bit5      ; 26 Enables debug outputs SW1-6
                    00327 #define DATA_OUT        portb,bit6      ; 27 ICSP and RS-232 to debug monitor
                    00328 #define spare28         portb,bit7      ; 28 ICSP and spare otherwise
                    00329
                    00330 #define spare11         portc,bit0      ; 11 spare spare spare
                    00331 #define spare12         portc,bit1      ; 12 spare spare spare
                    00332 #define spare13         portc,bit2      ; 13 spare spare spare
                    00333 #define spare14         portc,bit3      ; 14 spare spare spare
                    00334 #define spare15         portc,bit4      ; 15 spare spare spare
                    00335 #define ENABLE_OUTPUTS  portc,bit5      ; 16 RS-422/485 output enable
                    00336                                           ;    Controls outputs to the head-end
                    00337                                           ;    0 disables outputs
                    00338                                           ;    1 enables outputs
                    00339 #define UART_OUT        portc,bit6      ; 17 Output of the USART
                    00340 #define UART_IN         portc,bit7      ; 18 Input to the USART
                    00341 ;;te
                    00342
                    00343 ;;ps
                    00344 ; Mode bits.
                    00345 ;;pe
                    00346 ;;ts
                    00347 ; Bit usage in the mode1_M word:
```

```
                    00348 ;
                    00349 ;                       I             ; 7 In Motion
                    00350 ;                        N            ; 6 No more presets
                    00351 ;                         M           ; 5 Detected controller type
                    00352 ;                           C         ; 4 Clear the Spectra's screen
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  8
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00353 ;                           E         ; 3 Error detected
                    00354 ;                            D        ; 2 Serial debug mode is enabled
                    00355 ;                              r      ; 1 reserved, must be 0
                    00356 ;                               P     ; 0 Signal polarity status
                    00357 ;
                    00358 #define NORMAL_POLARITY mode1_M,bit0    ; 0 = normal polarity, 1 = reversed
                    00359 #define RESERVED_BIT    mode1_M,bit1    ; MUST remain at 0
                    00360 #define INPUT_ERROR     mode1_M,bit3    ; Input error flag
                    00361 #define CLEAR_DISPLAY   mode1_M,bit4    ; Should the Spectra display be cleared?
                    00362                                         ; 0 = No
                    00363                                         ; 1 = Yes
                    00364 #define RC216_MODE      mode1_M,bit5    ; Indicates the mode for speed cmnds
                    00365                                         ; 0 = AD2083/02, or other 8 speed src
                    00366                                         ; 1 = RC216, or other many speed device.
                    00367                                         ;     These devices send speeds out as
                    00368                                         ;     degree/second motion values.
                    00369 #define NO_MORE_PRESETS mode1_M,bit6    ; Enable/Disable preset processing
                    00370                                         ; This value is controlled by timer 1
                    00371                                         ; and is set after getting a preset
                    00372                                         ; command and is cleared after waiting
                    00373                                         ; for three more to come in.
                    00374                                         ; 0 = Enable processing
                    00375                                         ; 1 = Disable processing
                    00376 #define IN_MOVEMENT     mode1_M,bit7    ; Indicates if we are processing any
                    00377                                         ; motion command. When we are not
                    00378                                         ; processing a motion command a
                    00379                                         ; "faster" command causes a flip.
                    00380                                         ; 0 = Not in movement
                    00381                                         ; 1 = Yes in movement
                    00382
                    00383
                    00384 ; Bit usage in the mode2_M word:
                    00385 ;
                    00386 ;                       s             ; 7 spare
                    00387 ;                        s            ; 6 spare
                    00388 ;                         s           ; 5 spare
                    00389 ;                          s          ; 4 spare
                    00390 ;                           s         ; 3 spare
                    00391 ;                            b        ; 2 Used to indicate
                    00392 ;                             b       ; 1 . . when bytes come
                    00393 ;                              b      ; 0 . . . . in during Spectra sending
                    00394 ;
                    00395 ; NOTE all bits are cleared by clearing the full byte
                    00396 #define BYTE1           mode2_M,bit0    ; Used in receiving while xmitting
                    00397 #define BYTE2           mode2_M,bit1    ; . . to the Spectra
                    00398 #define BYTE3           mode2_M,bit2    ; . . . . indicates that all are in
                    00399 #define IN_FAST         mode2_M,bit3    ; Indicates that we are in a "fast"
                    00400                                         ; type command.
                    00401                                         ; 0 = No
                    00402                                         ; 1 = Yes
                    00403
                    00404 ; Bit usage in the mkmode1_M byte
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  9
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00405 ;
                    00406 ; In the land of Sensormatic, several camera options are selected by
                    00407 ; using several buttons simultaneously. These bits are used to control
                    00408 ; the detecting of these special simultaneous button depressings.
                    00409 ;
                    00410 ;                       s             ; 7 spare
                    00411 ;                        s            ; 6 spare
                    00412 ;                         s           ; 5 spare
                    00413 ;                          F          ; 4 Last command was "focus x"
                    00414 ;                           I         ; 3 Last command was "iris open"
                    00415 ;                            F        ; 2 Last cmnd was "fast"
                    00416 ;                             -       ; 1
                    00417 ;                              -      ; 0
```

```
                    00418 ;
                    00419 #define STEP1RESET      mkmode1_M,bit0  ; A dome reset consists of two commands
                    00420 #define STEP2RESET      mkmode1_M,bit1  ; First a zoom out followed by a focus
                    00421                                          ; far and last by an iris open command
                    00422 #define LAST_FAST       mkmode1_M,bit2  ; Set when "fast" (0x8D) is received
                    00423                                          ; 0 = Normal command
                    00424                                          ; 1 = Last command was a "fast" command
                    00425 #define STEP1MENU       mkmode1_M,bit3  ; Set when "iris open" (0x90) is rcvd
                    00426                                          ; 0 = Normal command
                    00427                                          ; 1 = Last command = "iris open"
                    00428 #define STEP2MENU       mkmode1_M,bit4  ; Set when a "focus x" command is
                    00429                                          ; received (0x87, near or 0x88, far).
                    00430                                          ; 0 = Normal command
                    00431                                          ; 1 = Last command = "focus x"
                    00432 ;;te
                    00433
                    00434
                    00435 ;;ps
                    00436 ; DBAUD_BASIC and DBAUD_REPEAT are used to control how long to delay when
                    00437 ; sending each bit, of bit-banged, data to the Spectra at 2400 baud. The
                    00438 ; original plan was to take baudperiod and multiply it by 8 (19200 = 2400
                    00439 ; * 8) to get a delay time.  However when this is done the result is
                    00440 ; greater than an eight bit number can hold (376) so a pair of values
                    00441 ; were needed.  One for a major loop and one for a minor inside loop.
                    00442 ;;pe
                    00443 ;;ts
                    00444 #define DBAUD_BASIC     47
                    00445 #define DBAUD_REPEAT     8
                    00446 ;;te
                    00447
                    00448
                    00449 ; Note as these get to higher speeds that more compensation is needed for
                    00450 ; what otherwise is minor overhead introduced elsewhere.
                    00451 #define DEBUG_PERIOD      4     ; Bit duration of debug data, 115200
                    00452 ;#define DEBUG_PERIOD     12     ; Bit duration of debug data, 57600
                    00453 ;#define DEBUG_PERIOD     20     ; Bit duration of debug data, 38400
                    00454 ;#define DEBUG_PERIOD     44     ; Bit duration of debug data, 19200
                    00455 ;#define DEBUG_PERIOD     90     ; Bit duration of debug data, 9600
                    00456 ;#define DEBUG_PERIOD     180    ; Bit duration of debug data, 4800
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 10
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00457
                    00458 ;;ps
                    00459 ; The Baud Rate Generator is used to provide a clock to the on-board
                    00460 ; USART which is used for output communications with the head end. All other
                    00461 ; serial IO is bit-banged.
                    00462 ;
                    00463 ; BAUD1200, BAUD2400, BAUD4800, BAUD9600 and BAUD19200 are used to load the
                    00464 ; baud rate generator for the onboard USART.  The data for all five of these
                    00465 ; equates assumes that BRGH is 0 (for low speed operation which gives an
                    00466 ; internal divide of 64 instead of 16.)  It is assumed that the xtal is
                    00467 ; cut for 11,059,200 Hz (AKA 11.0592 MHz).
                    00468 ;
                    00469 ; Baud Rate Generator formula from page 101 is:
                    00470 ;
                    00471 ;$$ baud rate = Fosc / (64 (BRG + 1))$$
                    00472 ;
                    00473 ; Solving for BRG gives us:
                    00474 ;
                    00475 ;$$ BRG = (Fosc/(baud rate * 64)) - 1$$
                    00476 ;
                    00477 ; or
                    00478 ;
                    00479 ;$$ BRG = ((Fosc/64) / baud rate) - 1$$
                    00480 ;$$ BRG = ((11095200/64) / baud rate) - 1$$
                    00481 ;$$ BRG = (173362.5 / baud rate) - 1$$
                    00482 ;
                    00483 ; The above calculations got me to an approximate starting value. I then
                    00484 ; fooled around with the numbers until they worked.
                    00485 ;;pe
                    00486 ;;ts
                    00487 #define BAUD1200        143     ;  1,200 baud
                    00488 #define BAUD2400         68     ;  2,400 baud
                    00489 #define BAUD4800         34     ;  4,800 baud
                    00490 #define BAUD9600         17     ;  9,600 baud
                    00491 #define BAUD19200         8     ; 19,200 baud
                    00492 ;;te
                    00493
```

```
                    00494 ;;ps
                    00495 ; Define bit positions for bit testing.
                    00496 ;;pe
                    00497 ;;ts
                    00498 #define bit0            0x00    ; Bit position '0'
                    00499 #define bit1            0x01    ; Bit position '1'
                    00500 #define bit2            0x02    ; Bit position '2'
                    00501 #define bit3            0x03    ; Bit position '3'
                    00502 #define bit4            0x04    ; Bit position '4'
                    00503 #define bit5            0x05    ; Bit position '5'
                    00504 #define bit6            0x06    ; Bit position '6'
                    00505 #define bit7            0x07    ; Bit position '7'
                    00506 ;;te
                    00507
                    00508
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 11
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    00509         page
                    00510 ;;latex\subsectionindex{Naming conventions}
                    00511 ;;ps
                    00512 ; Naming conventions used are as follows:
                    00513 ;
                    00514 ; Each RAM location will be have a name of some type that ends with an
                    00515 ; underscore "_" and the area of RAM that the variable is located in.
                    00516 ; There are only the following areas of memory: 0 = Last 96 locations in
                    00517 ; RAM bank 0, 1 = All 96 bytes of RAM bank 1.  In case of doubt always
                    00518 ; assume that the item is globally accessible, most of the unmarked items
                    00519 ; are defined by the MicroChip assembler and I do not want to change
                    00520 ; their names.
                    00521 ;
                    00522 ; Macros are in all uppercase. (Just like in well written C programs.)
                    00523 ;
                    00524 ; Note that the following RAM locations are non-functional. That is why
                    00525 ; there are "holes" in the memory allocations as they represent unused
                    00526 ; peripheral options.
                    00527 ;
                    00528 ; 0x08, 0x09, 0x88, 0x89, 0x8F, 0x90, 0x95, 0x97, 0x9A, 0x9B, 0x9C, 0x9D
                    00529 ;
                    00530 ; RAM 0 is used to store almost all variables.
                    00531
                    00532 ; The number following the name is the subroutine level that uses that
                    00533 ; RAM location. A RAM location may be used only by that level of
                    00534 ; subroutine, or the main line code. However a higher level routine may
                    00535 ; READ but NOT modify a lower level byte. There is a special suffix
                    00536 ; allowed and that is _M for items only accessed from the main line code.
                    00537 ;
                    00538 ; There is a special convention adopted to indicate which bank of RAM is
                    00539 ; active. That convention is to always select RAM bank 0, UNLESS a
                    00540 ; different bank is needed for some reason AND when a different RAM bank is
                    00541 ; selected to indicate it in the right hand column.
                    00542 ;;pe
                    00543
                    00544 ;;latex\subsectionindex{RAM usage}
                    00545 ;;latex\sloppy
                    00546 ;;ts
                    00547         cblock          RAM0    ; Defines the RAM bank 0 register file
                    00548
  00000020          00549         badcount_M              ; Number of errors in input data
  00000021          00550         bytebits_1              ; Number of bytes to bitbang out
  00000022          00551         checksum_M              ; Checksum for long messages
  00000023          00552         continuationtimer_M     ; Used for long timeouts when setting presets
                    00553                                 ; See also LONG_TIME_OUT's use and definition
                    00554                                 ; which is near the "again" tag.
                    00555
  00000024          00556         d_sync_M                ; 1 D proto sync byte
  00000025          00557         d_address_M             ; 2 D proto address byte
  00000026          00558         d_command1_M            ; 3 D proto command 1 (All iris, focus near)
  00000027          00559         d_command2_M            ; 4 D proto command 2 (Focus far, all zoom,
                    00560                                 ;    . . all motion)
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 12
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

  00000028          00561         d_pan_speed_M           ; 5 D proto data 1 (Pan speed)
  00000029          00562         d_tilt_speed_M          ; 6 D proto data 2 (Tilt speed/presets etc.)
  0000002A          00563         d_checksum_M            ; 7 D proto checksum
```

```
                        00564
0000002B                00565           eewritehere_0           ; Where to write into in EE memory
0000002C                00566           extratimeout_0          ; For long timeouts that are too big
0000002D                00567           hexlower_0              ; Debug lower half of a byte in ASCII
0000002E                00568           hexupper_0              ; Debug upper half of a byte in ASCII
0000002F                00569           lastchecksum_M          ; Most recent checksum
00000030                00570           manycount_5             ; Repeating count for repeated reply bytes
00000031                00571           messcount_4             ; Debug message counter/indicator thing
00000032                00572           mkmode1_M               ; Command bits relating to multiple
                        00573                                   ; . . simultaneous key depressions
00000033                00574           mode1_M                 ; System mode part 1
00000034                00575           mode2_M                 ; System mode part 2
00000035                00576           overlong1_0             ; Used in communicating with the Spectra
00000036                00577           overlong2_0             ; . these are needed because 2400 baud
                        00578                                   ; . . is real slow and we can't delay
                        00579                                   ; . . . long enough with just 8-bits
00000037                00580           patternwas_M            ; Last pattern worked on
00000038                00581           save3_M                 ; D Protocol saving space
00000039                00582           save4_M                 ; D Protocol saving space
0000003A                00583           save5_M                 ; D Protocol saving space
0000003B                00584           save6_M                 ; D Protocol saving space
0000003C                00585           saved_pan_speed_M       ; Saved values are updated by "fast" cmnds only
0000003D                00586           saved_tilt_speed_M
0000003E                00587           sentbyte_1              ; Byte being bit banged out
0000003F                00588           spectraaddress_M        ; Address for the Spectra
00000040                00589           sprotolength_M          ; Length of this S protocol message
                        00590
00000041                00591           sproto1_M               ; S protocol address,     byte 1
00000042                00592           sproto2_M               ; S protocol command,     byte 2
00000043                00593           sproto3_M               ; S protocol checksum,    byte 3
00000044                00594           sproto4_M               ; S protocol long message byte 4, motion speed
00000045                00595           sproto5_M               ; S protocol long message byte 5
00000046                00596           sproto6_M               ; S protocol long message byte 6
00000047                00597           sproto7_M               ; S protocol long message byte 7
00000048                00598           sproto8_M               ; S protocol long message byte 8
00000049                00599           sproto9_M               ; S protocol long message byte 9
0000004A                00600           sproto10_M              ; S protocol long message byte 10
0000004B                00601           sproto11_M              ; S protocol long message byte 11
0000004C                00602           sproto12_M              ; S protocol long message byte 12
0000004D                00603           sproto13_M              ; S protocol long message byte 13
                        00604
0000004E                00605           temp1_0                 ; Used to get around a limited instruction set
0000004F                00606           temp2_0                 ; Used to get around simple instruction set
00000050                00607           this_command1_M         ; The full series of "this" values exist
00000051                00608           this_command2_M         ; . so that they will not be modified
00000052                00609           this_pan_speed_M        ; . by other output commands, such as
00000053                00610           this_tilt_speed_M       ; . anything in the preset series, etc.
00000054                00611           thisbit_0               ; Used in bitbanging in Spectra data
00000055                00612           thisioerror_0           ; Most recent serial IO error
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 13
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

00000056                00613           timeout_0               ; Decrementing delay value
                        00614 ;;te
                        00615 ;;latex\fussy
                        00616 ; The last location MUST be 0x07F or lower
                        00617           endc                    ; ends RAM bank 0 definitions...
                        00618 ;;te
                        00619
                        00620
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 14
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                        00621           page
                        00622 ;;ts
                        00623 ; RAM 1 is spare
                        00624           cblock          RAM1    ; Define RAM bank 1
                        00625 ; The last location MUST be 0x0FF or lower
                        00626           endc                    ; ends RAM bank 1 definitions...
                        00627
                        00628 ; RAM 2 is spare
                        00629           cblock          RAM2    ; Define RAM bank 2
                        00630 ; The last location MUST be 0x17F or lower
                        00631           endc                    ; ends RAM bank 2 definitions...
                        00632
                        00633 ; RAM 3 is spare
```

```
                      00634         cblock          RAM3    ; Define RAM bank 3
                      00635 ; The last location MUST be 0x1FF or lower
                      00636         endc                    ; ends RAM bank 3 definitions...
                      00637 ;;te
                      00638
                      00639
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 15
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00640         page
                      00641 ;;ps
                      00642 ; Hardware resets, master clears, etc., start running here.
                      00643 ;;pe
                      00644 ;;ts
0000                  00645         org     RESET_VECTOR
0000   2CBF           00646         goto    init_M          ; Code for init_M, MUST be in code page 0
                      00647 ;;te
                      00648


                      00649         space 2
                      00650 ;;latex\subsectionindex{General coding rules}
                      00651 ;;ps
                      00652 ; In writing this code several coding rules were followed (if you look
                      00653 ; carefully, you can find exceptions to these, but there shouldn't be too
                      00654 ; many).
                      00655 ;
                      00656 ; 1. All code is in instruction bank 0.
                      00657 ;
                      00658 ; 2. All RAM is in bank 0.
                      00659 ;
                      00660 ; 3. Whenever any variation to the above rules is made, then the affected
                      00661 ;    bank (so far it is always a RAM or register bank) is indicated in
                      00662 ;    column 80. I.e. a "3" in column 80 indicates that whatever is
                      00663 ;    happening is going to happen in RAM/register bank "3".
                      00664 ;
                      00665 ; 4. All closed subroutines have their nesting level indicated in their
                      00666 ;    name as a suffix. I.e. doaux_3 may be called by subroutines xx_4 ...
                      00667 ;    xx_8 but not by xx_3. And more especially it may not be called by
                      00668 ;    xx_0 ... xx_2 type subroutines.
                      00669 ;
                      00670 ; 5. Variables have the lowest modifying subroutine indicated by their
                      00671 ;    suffix. I.e. hexlower_0 is a RAM location that gets changed by a
                      00672 ;    xx_0 subroutine. Thus hexlower_0 may not be used with any assurance
                      00673 ;    that it will remain undamaged by higher level subroutines such as
                      00674 ;    doaux_3. There is a special class of RAM locations and those are
                      00675 ;    those that have a suffix of "_M", and these are modified by the
                      00676 ;    "main" line code. RAM bank 0 is used for most purposes, however the
                      00677 ;    other RAM banks have various IO related registers that must be
                      00678 ;    accessed from time to time and they are the ones that cause
                      00679 ;    problems. This is because I didn't want to change MicroChip's names
                      00680 ;    that come in from the include file to indicate the "bank of
                      00681 ;    interest". So I just left them alone and hope that the "column 80"
                      00682 ;    comments will be enough. Nothing in column 80, indicates that
                      00683 ;    everything is happening in RAM bank 0.
                      00684 ;;pe
                      00685
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 16
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      00686         page
                      00687 ;;latex\subsectionindex{``main''}
                      00688 ;;ps
                      00689 ; The system starts runtime execution at "start". When the initialization
                      00690 ; code couplets, it transfers control to "start". Any required restarts,
                      00691 ; including error restarts transfer control to "again".
                      00692 ; (Had to change start to startstart so that I could find it easily.)
                      00693 ;;pe
0001                  00694 startstart
                      00695         ; Enable the receiver, selecting 8 or 9 bit mode is done in init_M
0001   1798           00696         bsf     rcsta,spen      ; spen = Serial Port Enable, bit 7
0002   1618           00697         bsf     rcsta,cren      ; cren = Continuous Receive Enable, bit 4
                      00698


                      00699         space 1
                      00700 ;;latex\subsubsection{Special ``set preset'' note}
```

```
                     00701 ;;ps
                     00702 ; The Sensormatic equipment sends requests for location (set preset
                     00703 ; commands) out three times. My logic involves incrementing a counter
                     00704 ; each time and then sending that information back to the controller.
                     00705 ; However I should only increment my counter once when getting in a
                     00706 ; series of preset sets. The solution to this is to disable processing
                     00707 ; presets, set a timer going that should last for two more position
                     00708 ; requests and to then reenable processing presets.
                     00709
                     00710 ; Preset-ignore timeouts work as follows:
                     00711 ;
                     00712 ; 1. A set preset command is detected.
                     00713 ;
                     00714 ; 2. A timer is started. The timer is based on the number of instructions
                     00715 ;    executed and being only 16 bits long is not long enough to give the
                     00716 ;    required timeout duration.
                     00717 ;
                     00718 ; 3. When the timeout sequence is started NO_MORE_PRESETS is set. And both
                     00719 ;    parts of timer 1 are cleared.
                     00720 ;
                     00721 ; 4. Every time through the main loop, starting at "again", the state of
                     00722 ;    timer 2 is checked. When the upper half of it has its most
                     00723 ;    significant bit set, then the continuation timer is incremented. And
                     00724 ;    timer 2 is again reset to zero.
                     00725 ;
                     00726 ; 5. When the continuation timer is large enough NO_MORE_PRESETS is
                     00727 ;    cleared, the timer is turned off and the system continues on as
                     00728 ;    though everything is normal.
                     00729 ;
                     00730 ; 6. An effort is made to have the NO_MORE_PRESETS flag set for 1/2 --> 1
                     00731 ;    seconds. The exact maximum timeout time is not too important. (I hope!)
                     00732 ;
                     00733 ;
                     00734 ; The timer MUST be active long enough to get past the two extra preset
                     00735 ; requests and not so long as to slow down a user that is setting
                     00736 ; presets. I have decided that a timeout that is always at least 1/2
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 17
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00737 ; second long and is usually less than 2 1/2 seconds long is about right.
                     00738 ;;pe
0003                 00739 again                           ; Error and general restart location
0003   1F33          00740         btfss   NO_MORE_PRESETS ; Are we in a preset timeout?
0004   280E          00741         goto    backtonormal    ; Nope, then don't miss the next command
                     00742
0005   1F8F          00743         btfss   tmr1h,bit7      ; Now wait until this gets negative
0006   280E          00744         goto    backtonormal    ; Finished "ignore out" for same presets
                     00745
0007   018E          00746         clrf    tmr1l           ; Reset these
0008   018F          00747         clrf    tmr1h           ; . . both parts
0009   0AA3          00748         incf    continuationtimer_M,f ; Increment the long timer
000A   1CA3          00749         btfss   LONG_TIME_OUT   ; Have we timed out?
000B   280E          00750         goto    backtonormal    ; Nope, still holding off on preset processing
                     00751
000C   1010          00752         bcf     t1con,tmr1on    ; Stop the timer from going
000D   1333          00753         bcf     NO_MORE_PRESETS ; Allow further preset processing
                     00754
000E                 00755 backtonormal
000E   3005          00756         movlw   BAD_TRY         ; Get error retry counter
000F   00A0          00757         movwf   badcount_M      ; Start/restart error counter
0010   3000          00758         movlw   S_PROTO_LENGTH  ; Get the length of a normal S protocol cmnd
0011   00C0          00759         movwf   sprotolength_M  ; Setup the normal message length
0012   2D53          00760         goto    getinput        ; Had to move this so that the following
                     00761                                 ; . . computed goto will work
                     00762
                     00763         space 1
                     00764 ;;latex\subsubsectionindex{decodeinput_M}
                     00765 ;;ps
                     00766 ; decodeinput_M This is a computed goto subroutine that is called from
                     00767 ; main and is used to decode commands in the range of 0x80 through 0xFF.
                     00768 ;
                     00769 ; On entry w has the command code as does sproto2_M
                     00770 ; Return is made to any of 64, or more, different places.
                     00771 ;
                     00772 ; For several camera functions, the method of "calling" them is to
                     00773 ; depress several keyboard buttons simultaneously in a specific order.
                     00774 ; When this happens various special camera operations are activated. If
                     00775 ; the order changes, or a "stop" command (which would indicate that the
```

```
                     00776 ; button was released) comes in, then the special sequence is stopped.
                     00777 ;
                     00778 ; The following commands get no response back:
                     00779 ;;pe
                     00780 ;;ts
                     00781 ;        0x00 --> 0x7F, 0x97, 0x98, 0x99, 0xA7, 0xBF
                     00782 ;        0xC1, 0xC2, 0xC4, 0xC5, 0xC7
                     00783 ;;te
                     00784 ;;ps
                     00785 ; The following commands get an ACK, but nothing is done inside the
                     00786 ; TXB-S422:
                     00787 ;;pe
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 18
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     00788 ;;ts
                     00789 ;        0xC8, 0xCA --> 0xDF, 0xF1 --> 0xFF
                     00790 ;;te
                     00791 ;
0013                 00792 decodeinput_M
0013   01A6          00793         clrf    d_command1_M    ; Don't have any left                    28FEB02
0014   01A7          00794         clrf    d_command2_M    ; . . over bits in these                 28FEB02
0015   3C7F          00795         sublw   0x7F            ; Is the command in the range of 0x00 --> 0x7F?
0016   1803          00796         btfsc   status,c        ; Is it a small one?
0017   2803          00797         goto    again           ; Too small, try for another
                     00798
0018   08B2          00799         movf    mkmode1_M,f     ; Are we in some part of a multikey command?
0019   1903          00800         btfsc   status,z        ; Check
001A   2835          00801         goto    notmultikey     ; Nope
                     00802
001B   1D32          00803         btfss   LAST_FAST       ; Was the last command a "fast"
001C   2821          00804         goto    lastnotfast     ; No
                     00805
                     00806 ;;ps
                     00807 ; A flip command set consists of a "fast" (0x8D) command
                     00808 ; immediately followed by a "fastest" (0x8E) command. Some
                     00809 ; controllers also send out a trailing pair of "fast stop"
                     00810 ; (0x8F) commands but they may be ignored as other controllers
                     00811 ; don't send them out at all!
                     00812 ;;pe
001D   3C8E          00813         sublw   S_FASTEST       ; Is this command "fastest"
001E   1D03          00814         btfss   status,z        ; Check
001F   1132          00815         bcf     LAST_FAST       ; Nope, clear flag
0020   2835          00816         goto    notmultikey     ; All processing for the command following
                     00817                                 ; . . a "fast" command
                     00818
                     00819 ;;ps
                     00820 ; In testing with genuine Sensormatic domes and sim58, I have discovered
                     00821 ; that if a "fastest" command is received and there is no current
                     00822 ; motion going on, that the dome will execute a flip. So I have added in
                     00823 ; that feature too.
                     00824 ;;pe
0021                 00825 lastnotfast
0021   1DB2          00826         btfss   STEP1MENU       ; Check for a menu starter
0022   282D          00827         goto    lastnotmenu1
                     00828
0023   11B2          00829         bcf     STEP1MENU       ; Clear caller
0024   0842          00830         movf    sproto2_M,w     ; Restore input command
0025   3C87          00831         sublw   S_FOCUS_NEAR    ; Is this a "focus near" command? (0x87)
0026   1903          00832         btfsc   status,z
0027   282B          00833         goto    isfocus
                     00834
0028   3E01          00835         addlw   1               ; Is this a "focus far" command? (0x88)
0029   1D03          00836         btfss   status,z
002A   2835          00837         goto    notmultikey
                     00838
002B                 00839 isfocus
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 19
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

002B   1632          00840         bsf     STEP2MENU
002C   2835          00841         goto    notmultikey
                     00842
002D                 00843 lastnotmenu1
002D   1C32          00844         btfss   STEP1RESET      ; Are we working on a reset sequence
002E   2835          00845         goto    notmultikey
```

```
                        00846
002F   1032             00847         bcf     STEP1RESET
0030   0842             00848         movf    sproto2_M,w
0031   3C88             00849         sublw   S_FOCUS_FAR
0032   1D03             00850         btfss   status,z
0033   2835             00851         goto    notmultikey
                        00852
0034   14B2             00853         bsf     STEP2RESET
                        00854
0035                    00855 notmultikey                     ; Normal command, not part of a multikey cmnd
0035   0842             00856         movf    sproto2_M,w     ; Restore input command
0036   01A8             00857         clrf    d_pan_speed_M   ; Clear Data 1
0037   01A9             00858         clrf    d_tilt_speed_M  ; Clear Data 2 which becomes a counter
0038   39C0             00859         andlw   0xC0            ; Is the command .ge. C0? I.e. C0 --> FF
0039   3CC0             00860         sublw   0xC0            ; Cx, Dx, Ex and Fx all have the two MSBs set
003A   1903             00861         btfsc   status,z        ; Check
003B   2880             00862         goto    cormore         ; Yep it was originally from C0 --> FF
                        00863
                        00864         ; Sneaky way to create a computed jump table, from Applications Note
                        00865         ; 514. Because of the way that the "addwf pcl,f" instruction works,
                        00866         ; remembering that this is an 8-bit computer, in the below
                        00867         ; code, this trick must end in the first 0xFF locations of an
                        00868         ; instruction page.
003C   0842             00869         movf    sproto2_M,w     ; Get original command
003D   393F             00870         andlw   0x3F            ; Dump slop, making command range 00 --> 3F
003E   00CE             00871         movwf   temp1_0         ; Save command, command range is now 00 --> 3F
                        00872                                 ; This is needed because there are only 64
                        00873                                 ; entries in this computed goto table.
003F   0782             00874         addwf   pcl,f           ; Now increment the "P" register
                        00875


                        00876         space 1
                        00877 ;;latex\subsubsectionindex{Master command decode table}
                        00878 ;;specialstart
0040   2A13             00879         goto unknown80  ; 0x80 Unknown three byte command
0041   2915             00880         goto panleft    ; 0x81 Pan Left (27) until Pan Right or Pan Stop
0042   2912             00881         goto panright   ; 0x82 Pan Right (27) until Pan Left or Pan Stop
0043   2997             00882         goto stoppan    ; 0x83 Stop panning
                        00883
0044   28D6             00884         goto tiltup     ; 0x84 Tilt Up until Tilt Down or Tilt Stop
0045   28D9             00885         goto tiltdown   ; 0x85 Tilt Down until Tilt Up or Tilt Stop
0046   299C             00886         goto stoptilt   ; 0x86 Stop tilting
0047   2951             00887         goto focusnear  ; 0x87 Focus near until Focus Far or Focus Stop
                        00888
0048   2954             00889         goto focusfar   ; 0x88 Focus Far until Focus Near or Focus Stop
0049   29AE             00890         goto stopfocus  ; 0x89 Stop Focus
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 20
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                        00891
004A   2958             00892         goto zoomin     ; 0x8A Zoom In until Zoom Out or Zoom Stop
004B   295B             00893         goto zoomout    ; 0x8B Zoom Out until Zoom In or Zoom Stop
                        00894
004C   29B1             00895         goto stopzoom   ; 0x8C Stop zoom
004D   291A             00896         goto fast       ; 0x8D Increase Pan and Tilt speeds to 108/44
004E   292B             00897         goto fastest    ; 0x8E Increase Pan and Tilt speeds to 54/19
004F   29A7             00898         goto stopfast   ; 0x8F Stop Fast/Fastest speeds (back to normal 27)
                        00899
0050   293D             00900         goto irisopen   ; 0x90 Opens Iris
0051   2947             00901         goto irisclose  ; 0x91 Closes Iris
0052   29B4             00902         goto stopiris   ; 0x92 Stop Iris offset adjustment
0053   29B7             00903         goto stopall    ; 0x93 Stop All movement
                        00904
0054   2993             00905         goto getdometype; 0x94 Request dome type (poll)
0055   2A22             00906         goto getalarms  ; 0x95 Request status of alarm inputs
0056   2A16             00907         goto unknown96  ; 0x96 Undetected unknown command
0057   2803             00908         goto again      ; 0x97 ACKnowledge sent to dome, don't do anything.
                        00909                         ;      I.e. don't ack an ack.
                        00910
0058   2803             00911         goto again      ; 0x98 Start temp no transmit, no ack
0059   2803             00912         goto again      ; 0x99 End temp no transmit, no ack
005A   2923             00913         goto faster     ; 0x9A Increase Pan and Tilt speeds to 81/30
005B   29A7             00914         goto stopfaster ; 0x9B Stop Faster speeds (back to normal 27/12)
                        00915
005C   2A19             00916         goto boundarystart ; 0x9C Start Boundary Definition.
005D   2A1C             00917         goto boundarymark ; 0x9D Marks the Current Position as a Boundary
005E   2A10             00918         goto onair      ; 0x9E Set On Air status to tell the dome to send the
                        00919                         ;      current Boundary Position
005F   2A12             00920         goto onairreset ; 0x9F Reset On Air status
```

```
0060   2968           00921        goto pattern1   ; 0xA0 Start defining Pattern 1
0061   2967           00922        goto pattern2   ; 0xA1 Start defining Pattern 2
0062   2966           00923        goto pattern3   ; 0xA2 Start defining Pattern 3
0063   2A12           00924        goto patternaccept; 0xA3 Accept the New Pattern
                      00925
0064   2E36           00926        goto memorydump ; 0xA4 Memory Dump
0065   2DD5           00927        goto getposition; 0xA5 Request Dome Position Coordinates
0066   2E2F           00928        goto gotoposition ; 0xA6 Goto Absolute Position (Multiple-byte format)
0067   29BE           00929        goto unknown    ; 0xA7 Undetected unknown command, place holder
                      00930
0068   297E           00931        goto mark1      ; 0xA8 Mark the current position as Target 1
0069   297D           00932        goto mark2      ; 0xA9 Mark the current position as Target 2
006A   297C           00933        goto mark3      ; 0xAA Mark the current position as Target 3
006B   297B           00934        goto mark4      ; 0xAB Mark the current position as Target 4
                      00935
006C   2972           00936        goto run1       ; 0xAC Goto start of Pattern 1 (now its Run Pattern 1)
006D   2971           00937        goto run2       ; 0xAD Goto start of Pattern 2 (now its Run Pattern 2)
006E   2970           00938        goto run3       ; 0xAE Goto start of Pattern 3 (now its Run Pattern 3)
006F   2970           00939        goto run3       ; 0xAF Goto start of Pattern 4 (now its Run Pattern 3)
                      00940
0070   2972           00941        goto run1       ; 0xB0 Run Pattern 1
0071   2971           00942        goto run2       ; 0xB1 Run Pattern 2
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 21
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0072   2970           00943        goto run3       ; 0xB2 Run Pattern 3
0073   2974           00944        goto runreview  ; 0xB3 Run a newly defined Pattern to review it
                      00945
0074   2987           00946        goto target1    ; 0xB4 Go to Target 1
0075   2986           00947        goto target2    ; 0xB5 Go to Target 2
0076   2985           00948        goto target3    ; 0xB6 Go to Target 3
0077   2984           00949        goto target4    ; 0xB7 Go to Target 4
                      00950
0078   296C           00951        goto patternend ; 0xB8 Tells the dome to stop recording (defining)
0079   297A           00952        goto mark5      ; 0xB9 Mark the current position as Target 5
007A   2979           00953        goto mark6      ; 0xBA Mark the current position as Target 6
007B   2978           00954        goto mark7      ; 0xBB Mark the current position as Target 7
                      00955
007C   2983           00956        goto target5    ; 0xBC Go to Target 5
007D   2982           00957        goto target6    ; 0xBD Go to Target 6
007E   2981           00958        goto target7    ; 0xBE Go to Target 7
007F   29BE           00959        goto unknown    ; 0xBF Undetected and unknown command, place holder
                      00960 ;;specialend
                      00961
                      00962
                      00963 ; Command is large, which is an underpopulated area (i.e. "sparse")
0080                  00964 cormore
0080   0842           00965        movf sproto2_M,w; Get original command C0 --> FF
0081   3CC0           00966        sublw S_VARIABLE_SPEED ; Well what was it?
0082   1903           00967        btfsc status,z  ; Check
0083   29C0           00968        goto variablespeed ; 0xC0 Variable speed control (Multiple-byte format)
                      00969
0084   0842           00970        movf sproto2_M,w; Get original command C0 --> FF
0085   3CC1           00971        sublw S_UNKNOWN_C1
0086   1903           00972        btfsc status,z  ; Check
0087   29BE           00973        goto unknownc1  ; 0xC1 Unknown C1
                      00974
0088   0842           00975        movf sproto2_M,w; Get original command C0 --> FF
0089   3CC3           00976        sublw S_PROP_SPEED
008A   1903           00977        btfsc status,z  ; Check
008B   29BE           00978        goto propspeed  ; 0xC3 Proportional speed pan or tilt movement
                      00979
008C   0842           00980        movf sproto2_M,w; Get original command C0 --> FF
008D   3CC6           00981        sublw S_RESET
008E   1903           00982        btfsc status,z  ; Check
008F   298A           00983        goto reset      ; 0xC6 Run default "Apple Peel" pattern for a spiral
                      00984                        ; In out case we just reset the dome.
                      00985
0090   0842           00986        movf sproto2_M,w; Get original command C0 --> FF
0091   3CC9           00987        sublw S_SOFTWARE_VERSION
0092   1903           00988        btfsc status,z  ; Check
0093   2E18           00989        goto version    ; 0xC9 Get software version number from dome
                      00990
                      00991        ; Decoding for the range of 0xE0 --> 0xEF (S_OUTPUT0 --> S_OUTPUT0+0x0F)
0094   0842           00992        movf sproto2_M,w; Get original command C0 --> FF
0095   3CE0           00993        sublw   S_OUTPUT0 ; Subtract out the 0xE0
0096   1803           00994        btfsc   status,c; Is it smaller than 0xE0?
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 22
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
```

```
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0097   29BE           00995         goto    unknown ; Yep, it's smaller
                      00996
0098   0842           00997         movf sproto2_M,w; Getting it again
0099   3CEF           00998         sublw S_OUTPUT0+0x0F; Is this upper end?
009A   1803           00999         btfsc status,c  ; Nope, but it is within low and high
009B   29DE           01000         goto output     ; 0xE0 Set output drivers range = 0xE0 --> 0xEF
                      01001
009C   0842           01002         movf sproto2_M,w; Get original command C0 --> FF
009D   3CF0           01003         sublw S_TERM_PAT
009E   1903           01004         btfsc status,z  ; Check
009F   2A1F           01005         goto terminatepattern ; 0xF0 Terminate current pattern
                      01006
00A0   29BE           01007         goto unknown    ; Woops it's an unknown command
                      01008
                      01009
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 23
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01010         page
                      01011 ; Command processors
                      01012

                      01013         space 1
                      01014 ;;latex\subsubsubsectionindex{AD2083/02 code translator speeds, pan logic}
                      01015 ;;ps
                      01016 ; With the AD2083/02 code translator, there are only eight variable
                      01017 ; speeds available. These have to be translated into something that is
                      01018 ; reasonable for the Spectra to use. Unfortunately the data comes out in
                      01019 ; degrees/sec (?) while Pelco uses a different system to get up to 64
                      01020 ; different speeds. So here we have to determine which of the eight speed
                      01021 ; values are being sent and then translate these values into Spectra
                      01022 ; equivalents. If a careful examination is made of the values sent from
                      01023 ; the AD translator, it will be observed that the values are highly
                      01024 ; non-linear. However Pelco's D protocol uses a different set of
                      01025 ; non-linear steps. The next code makes the required translations.
                      01026 ;;pe
                      01027 ;;ts
                      01028 ; Observed speeds       Decision point     Generated speeds
                      01029 ;       0x04                    0x05              7/0x07
                      01030 ;       0x06                    0x08             15/0x0F
                      01031 ;       0x0A                    0x0C             23/0x17
                      01032 ;       0x0F                    0x13             31/0x1F
                      01033 ;       0x18                    0x1C             39/0x27
                      01034 ;       0x21                    0x27             47/0x2F
                      01035 ;       0x2D                    0x43             55/0x37
                      01036 ;       0x5A and other          0x43+            64/0x40 Or turbo speed
                      01037 ;;te
                      01038 ;;ps
                      01039 ; In the above table the column marked "Decision point" indicates at this
                      01040 ; value, all lower speeds are translated into the values shown in the
                      01041 ; "Generated speeds" column. Remember that the lower values are checked
                      01042 ; first.
                      01043 ;
                      01044 ; panspeed_M is the pan speed to translate and is setup from sproto4_M
                      01045 ; or is forced to a default value by fixed speed commands, and is
                      01046 ; modified by the fixed speed enhanced speed buttons.
                      01047 ;;pe
00A1                  01048 sendpancommand
00A1   1AB3           01049         btfsc   RC216_MODE      ; 0xC0 Is this an RC216 switch
00A2   28CD           01050         goto    do216pan        ; Yep
                      01051
00A3   0852           01052         movf    this_pan_speed_M,w; Get pan speed
00A4   3C05           01053         sublw   0x05            ; Slowest speed
00A5   1803           01054         btfsc   status,c
00A6   28C7           01055         goto    pspeed1
                      01056
00A7   3E03           01057         addlw   0x08-0x05
00A8   1803           01058         btfsc   status,c
00A9   28C5           01059         goto    pspeed2
                      01060
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 24
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
                        01061         addlw   0x0C-0x08
00AB   1803             01062         btfsc   status,c
00AC   28C3             01063         goto    pspeed3
                        01064
00AD   3E07             01065         addlw   0x13-0x0C
00AE   1803             01066         btfsc   status,c
00AF   28C1             01067         goto    pspeed4
                        01068
00B0   3E09             01069         addlw   0x1C-0x13
00B1   1803             01070         btfsc   status,c
00B2   28BF             01071         goto    pspeed5
                        01072
00B3   3E0B             01073         addlw   0x27-0x1C
00B4   1803             01074         btfsc   status,c
00B5   28BD             01075         goto    pspeed6
                        01076
00B6   3E1C             01077         addlw   0x43-0x27
00B7   1803             01078         btfsc   status,c
00B8   28BB             01079         goto    pspeed7
                        01080
                        01081         ; What would be "pspeed8" just falls thru the end of the above decodes
00B9   3040             01082 pspeed8 movlw    64             ; Turbo speed is 64
00BA   28C9             01083         goto     stickpanin64
                        01084
00BB   3037             01085 pspeed7 movlw    55
00BC   28C8             01086         goto     stickpanin
                        01087
00BD   302F             01088 pspeed6 movlw    47
00BE   28C8             01089         goto     stickpanin
                        01090
00BF   3027             01091 pspeed5 movlw    39
00C0   28C8             01092         goto     stickpanin
                        01093
00C1   301F             01094 pspeed4 movlw    31
00C2   28C8             01095         goto     stickpanin
                        01096
00C3   3017             01097 pspeed3 movlw    23
00C4   28C8             01098         goto     stickpanin
                        01099
00C5   300F             01100 pspeed2 movlw    15
00C6   28C8             01101         goto     stickpanin
                        01102
00C7   3007             01103 pspeed1 movlw    7
                        01104
                        01105 ;;ps
                        01106 ; stickpanin is also called from fixed speed pan commands of 0x81 and
                        01107 ; 0x82. At this point w has the new speed value to sent to the Spectra.
                        01108 ;;pe
00C8                    01109 stickpanin
00C8   12B3             01110         bcf     RC216_MODE      ; Assume that it is not an RC216 controller
00C9                    01111 stickpanin64
00C9   00D2             01112         movwf   this_pan_speed_M
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 25
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

00CA   17B3             01113         bsf     IN_MOVEMENT     ; Say that we are moving
00CB   2257             01114         call    build_command_0 ; Build up the command
00CC   29F5             01115         goto    sendcommand     ; Send it
                        01116
                        01117 ;;latex\subsubsubsectionindex{VM96 variable speeds, pan}
                        01118 ;;ps
                        01119 ; It should be noted that Pelco pan speeds start changing at 8.
                        01120 ; (I.e. all values less than 8 give the same slow pan rate.) This was
                        01121 ; done for compatibility with older keyboards. There are also several
                        01122 ; speeds that get "jumped" over to avoid mechanical resonance noise in
                        01123 ; the individual Spectra its self. (And the skipped numbers change with
                        01124 ; different rev levels of the Spectra software. I don't really want to
                        01125 ; mention what happens with the Esprit.) The net result of all this is
                        01126 ; that we have less than 64 speeds available and it is impossible to
                        01127 ; predict what will happen in any given situation.
                        01128 ;;pe
00CD                    01129 do216pan
00CD   0852             01130         movf    this_pan_speed_M,w; Get input pan speed
00CE   228D             01131         call    checkrate_0     ; Check the speed control bits
00CF   3E08             01132         addlw   8               ; Now get over the repeated start values
00D0   00CE             01133         movwf   temp1_0         ; Save start value
00D1   3C40             01134         sublw   0x40            ; Turbo, or more, speed? (In "beta" was 0x63.)
00D2   1C03             01135         btfss   status,c        ; Yep
```

```
00D3   28B9           01136         goto    pspeed8         ; Turbos away
                      01137
00D4   084E           01138         movf    temp1_0,w       ; Restore input + 8 value for use
00D5   28C9           01139         goto    stickpanin64    ; And send it.
                      01140
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 26
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01141         page
00D6                  01142 tiltup                          ; 0x84 Tilt Up until Tilt Down or Tilt Stop
00D6   15D1           01143         bsf     TILT_UP
00D7   1251           01144         bcf     TILT_DOWN
00D8   28DB           01145         goto    fixed_tilt
                      01146
00D9                  01147 tiltdown                        ; 0x85 Tilt Down until Tilt Up or Tilt Stop
00D9   1651           01148         bsf     TILT_DOWN
00DA   11D1           01149         bcf     TILT_UP
00DB                  01150 fixed_tilt
00DB   11B4           01151         bcf     IN_FAST         ; Say that we are not in a "fast" type command
00DC   3024           01152         movlw   TILT_DEFAULT    ; Get the default speed for tilt
00DD   2905           01153         goto    sticktiltin     ; And send it to the Spectra
                      01154
                      01155         space 1
                      01156 ;;latex\subsubsubsectionindex{AD2083/02 code translator speeds, tilt logic}
                      01157 ;;ps
                      01158 ; this_tilt_speed_M is the tilt speed to translate and is setup from
                      01159 ; sproto4_M or is forced to a default value by fixed speed commands, and is
                      01160 ; modified by the fixed speed enhanced speed buttons.
                      01161 ;;pe
                      01162 ;;ts
                      01163 ; Observed speeds       Decision point     Generated speeds
                      01164 ;       0x03                    0x04              3/0x03
                      01165 ;       0x05                    0x07             11/0x0B
                      01166 ;       0x09                    0x0B             19/0x13
                      01167 ;       0x0D                    0x10             27/0x1B
                      01168 ;       0x14                    0x16             35/0x23
                      01169 ;       0x18                    0x1C             43/0x2B
                      01170 ;       0x21                    0x27             51/0x33
                      01171 ;       0x2D and other          0x27+            63/0x3F Or max tilt spd
                      01172 ;;te
                      01173 ;
00DE                  01174 sendtiltcommand
00DE   1AB3           01175         btfsc   RC216_MODE      ; 0xC0 Is this an RC216 switch
00DF   2909           01176         goto    do216tilt       ; Yep
                      01177
00E0   0853           01178         movf    this_tilt_speed_M,w; Get tilt speed
00E1   3C04           01179         sublw   0x04            ; Slowest speed
00E2   1803           01180         btfsc   status,c
00E3   2904           01181         goto    tspeed1
                      01182
00E4   3E03           01183         addlw   0x07-0x04
00E5   1803           01184         btfsc   status,c
00E6   2902           01185         goto    tspeed2
                      01186
00E7   3E04           01187         addlw   0x0B-0x07
00E8   1803           01188         btfsc   status,c
00E9   2900           01189         goto    tspeed3
                      01190
00EA   3E05           01191         addlw   0x10-0x0B
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 27
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

00EB   1803           01192         btfsc   status,c
00EC   28FE           01193         goto    tspeed4
                      01194
00ED   3E06           01195         addlw   0x16-0x10
00EE   1803           01196         btfsc   status,c
00EF   28FC           01197         goto    tspeed5
                      01198
00F0   3E06           01199         addlw   0x1C-0x16
00F1   1803           01200         btfsc   status,c
00F2   28FA           01201         goto    tspeed6
                      01202
00F3   3E0B           01203         addlw   0x27-0x1C
00F4   1803           01204         btfsc   status,c
```

```
00F5   28F8           01205         goto    tspeed7
                      01206
                      01207         ; What would be "tspeed8" just falls thru the end of the decodes
00F6   303F           01208 tspeed8 movlw   63              ; There is no turbo in tilt
00F7   2905           01209         goto    sticktiltin
                      01210
00F8   3033           01211 tspeed7 movlw   51
00F9   2905           01212         goto    sticktiltin
                      01213
00FA   302B           01214 tspeed6 movlw   43
00FB   2905           01215         goto    sticktiltin
                      01216
00FC   3023           01217 tspeed5 movlw   35
00FD   2905           01218         goto    sticktiltin
                      01219
00FE   301B           01220 tspeed4 movlw   27
00FF   2905           01221         goto    sticktiltin
                      01222
0100   3013           01223 tspeed3 movlw   19
0101   2905           01224         goto    sticktiltin
                      01225
0102   300B           01226 tspeed2 movlw   11
0103   2905           01227         goto    sticktiltin
                      01228
0104   3003           01229 tspeed1 movlw    3
                      01230 ;;ps
                      01231 ; sticktiltin is called three ways. One way is from AD type variable
                      01232 ; speed commands, the second way is from RC216 type variable speed
                      01233 ; commands and the third way is from fixed speed (RC58 type) commands.
                      01234 ; w has the speed to use.
                      01235 ;;pe
0105                  01236 sticktiltin
0105   00D3           01237         movwf   this_tilt_speed_M; Stick tilt speed into the command
0106   17B3           01238         bsf     IN_MOVEMENT     ; Say that we are moving
0107   2257           01239         call    build_command_0 ; Build up the command
0108   29F5           01240         goto    sendcommand     ; Send it
                      01241
                      01242         space 1
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 28
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01243 ;;latex\subsubsubsectionindex{VM96 variable speeds, tilt}
                      01244 ;;ps
                      01245 ; It should be noted that Pelco tilt speeds start changing at 6. (I.e.
                      01246 ; all values less than 6 give the same slow tilt rate.) This was done for
                      01247 ; compatibility with older keyboards. There are also several speeds that
                      01248 ; get "jumped" over to avoid mechanical resonance noise in the individual
                      01249 ; Spectra its self. (And the skipped numbers change with different rev
                      01250 ; levels of the Spectra software.) The net result of all this is that we
                      01251 ; have less than 64 speeds available and it is impossible to predict what
                      01252 ; will happen in any given situation/software revision.
                      01253 ;
                      01254 ; The starting value of 6 was increased by 6 more to attempt and match
                      01255 ; Sensormatic's tilt speeds. Sensormatic has a "rapid" set of pan and tilt
                      01256 ; speeds compared to those that Pelco uses.
                      01257 ;
                      01258 ; (This routine was changed on 26FEB02.)
                      01259 ;;pe
0109                  01260 do216tilt
0109   0853           01261         movf    this_tilt_speed_M,w; Get input speed
010A   228D           01262         call    checkrate_0     ; Check the speed control bits
010B   3E0C           01263         addlw   6+6             ; Get over Pelco's starting repeats
010C   00CE           01264         movwf   temp1_0         ; Save start value
010D   3C3F           01265         sublw   0x3F            ; Is this a turbo speed, or faster, command?
010E   1C03           01266         btfss   status,c        ; Yep, and there's no such thing as turbo-tilt
010F   28F6           01267         goto    tspeed8         ; Limit out at the fastest tilt speed
                      01268
0110   084E           01269         movf    temp1_0,w       ; Restore input +6 value for use
0111   2905           01270         goto    sticktiltin     ; And send it.
                      01271
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 29
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01272         page
                      01273 ;;latex\subsubsubsectionindex{VM1 fixed speed motion controls, pan and tilt}
```

```
                         01274 ;;ps
                         01275 ;        Fixed speed motion controls.
                         01276 ;;pe
0112                     01277 panright                        ; 0x81 Pan right (27) until Pan Left or Stop
0112   14D1              01278         bsf     PAN_RIGHT
0113   1151              01279         bcf     PAN_LEFT
0114   2917              01280         goto    fixed_pan
                         01281
0115                     01282 panleft                         ; 0x82 Pan left (27) until Pan Right or Stop
0115   1551              01283         bsf     PAN_LEFT
0116   10D1              01284         bcf     PAN_RIGHT
0117                     01285 fixed_pan
0117   11B4              01286         bcf     IN_FAST         ; Say that we are not in a "fast" type command
0118   3033              01287         movlw   PAN_DEFAULT     ; Get the default speed for pan
0119   28C8              01288         goto    stickpanin      ; And send it to the Spectra
                         01289
                         01290         space 1
                         01291 ;;latex\subsubsubsectionindex{VM1 fixed speed motion controls, ``fast''}
                         01292 ;;ps
                         01293 ; Fixed speed, speed changing commands.
                         01294 ;
                         01295 ; On the RC58 there are four fixed speeds. These are "normal" which is
                         01296 ; what you get when the "fingerstick" is depressed to cause motion.
                         01297 ; Then the system allows for three options to provide quicker motion.
                         01298 ; These three commands are called "fast", "faster" and "fastest".
                         01299 ; The more rapid movement speeds are obtained by using the
                         01300 ; "fingerstick" and one of the three speed increasing keys. On an RC58
                         01301 ; these keys are marked "RAIL Right", "Fast" and "RAIL Left".
                         01302 ; "RAIL Right" gives a times two increase over the basic speed,
                         01303 ; "Fast" gives a times three increase and "RAIL left" gives a times
                         01304 ; four increase. (Note that the really old domes, the MiniDome and Cobra
                         01305 ; Dome, do not respond to the two "RAIL" speed increases but that the
                         01306 ; SpeedDome does.) Speed increasing commands apply to pan AND tilt
                         01307 ; simultaneously.
                         01308 ;
                         01309 ; As an unexpected bonus of using the "RAIL" speed increases, the names
                         01310 ; associated with the button pushes have little relationship with the
                         01311 ; actual speed increases generated. These speeds were obtained by using
                         01312 ; sim58 and "SpeedDome 2000 Outdoor", or dome #5, which was bought new
                         01313 ; for this project.
                         01314 ;
                         01315 ; Thus we have: (The speeds in this table were determined by using a stop
                         01316 ; watch and then rounding the results.)
                         01317 ;;pe
                         01318 ;;ts
                         01319 ;              Key          Command    Speed    Degrees/Sec   Name
                         01320 ;              "plain"      ...         X1         27         PAN_DEFAULT
                         01321 ;              RAIL --->    fastest     X2         54         PAN_FASTEST
                         01322 ;              Fast         faster      X3         81         PAN_FASTER
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 30
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                         01323 ;              RAIL <---    fast        X4        108         PAN_FAST
                         01324 ;
                         01325 ;              "plain"      ...         X1         12         TILT_DEFAULT
                         01326 ;              RAIL --->    fastest     X2         19         TILT_FASTEST
                         01327 ;              Fast         faster      X3         30         TILT_FASTER
                         01328 ;              RAIL <---    fast        X4         44         TILT_FAST
                         01329 ;;te
011A                     01330 fast                            ; 0x8D Increase pan and tilt speeds
011A   1532              01331         bsf     LAST_FAST       ; Remember what command type this was
011B   1FB3              01332         btfss   IN_MOVEMENT     ; Are we moving?
011C   29BE              01333         goto    ackdonecommand  ; Nope, but ack it anyway
                         01334
011D   2451              01335         call    speed_save_0    ; Save the original values
011E   3040              01336         movlw   PAN_FAST        ; Set pan to 150/sec
011F   00D2              01337         movwf   this_pan_speed_M
0120   303F              01338         movlw   TILT_FAST       ; Set tilt to 44/sec
0121   00D3              01339         movwf   this_tilt_speed_M
0122   293A              01340         goto    resendmotion    ; Start/continue motion at new rate
                         01341
0123                     01342 faster                          ; 0x9A Increase pan and tilt speeds
0123   1FB3              01343         btfss   IN_MOVEMENT     ; Are we moving?
0124   29BE              01344         goto    ackdonecommand  ; Nope, be sure to ack it though
                         01345
0125   2451              01346         call    speed_save_0    ; Save the original values
0126   303F              01347         movlw   PAN_FASTER
0127   00D2              01348         movwf   this_pan_speed_M
```

```
0128   3039           01349         movlw   TILT_FASTER
0129   00D3           01350         movwf   this_tilt_speed_M
012A   293A           01351         goto    resendmotion    ; Start/continue motion at new rate
                      01352
                      01353 ;;ps
                      01354 ; There are two ways to do a flip. The documented one says to send a
                      01355 ; "fast" command and to follow it with a "fastest" command. In examining
                      01356 ; what SpeedDomes and DeltaDomes actually do, if the dome is not in
                      01357 ; motion, just sending a "fastest" command will cause it to flip. So this
                      01358 ; logic handles both cases.
                      01359 ;;pe
012B                  01360 fastest                         ; 0x8E Increase pan and tilt speeds
012B   1D32           01361         btfss   LAST_FAST       ; Is this going to be a "flip"?
012C   2933           01362         goto    notaflip
                      01363
012D   13B3           01364 doflip  bcf     IN_MOVEMENT     ; Clear motion status
012E   1132           01365         bcf     LAST_FAST       ; Get ready to do a flip
012F   3021           01366         movlw   D_FLIP          ; Get a flip command
0130   00A9           01367         movwf   d_tilt_speed_M  ; Stick it in as a preset
0131   3007           01368         movlw   D_GOTO_PRESET   ; Send a go to preset command
0132   29F4           01369         goto    set_d_command2  ; Do it
                      01370
0133                  01371 notaflip
0133   1FB3           01372         btfss   IN_MOVEMENT     ; Are we in motion?
0134   292D           01373         goto    doflip          ; Nope, do flip
                      01374
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 31
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0135   2451           01375         call    speed_save_0    ; Save the current speeds
0136   303B           01376         movlw   PAN_FASTEST     ; Not a flip just a "fastese"
0137   00D2           01377         movwf   this_pan_speed_M
0138   3030           01378         movlw   TILT_FASTEST
0139   00D3           01379         movwf   this_tilt_speed_M
013A                  01380 resendmotion
013A   15B4           01381         bsf     IN_FAST         ; Say we have a "fast" type command here
013B   2257           01382         call    build_command_0
013C   29F5           01383         goto    sendcommand      ; Send it
                      01384
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 32
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE


                      01385         page
                      01386 ; Iris processing.
013D                  01387 irisopen                        ; 0x90 Opens iris
013D   1CB2           01388         btfss   STEP2RESET      ; Last step of a reset sequence?
013E   2943           01389         goto    normalirisopen
                      01390
013F   10B2           01391         bcf     STEP2RESET      ; Clear caller
0140   300F           01392         movlw   D_RESET
0141   00A7           01393         movwf   d_command2_M
0142   2956           01394         goto    sendnonmotion
                      01395
0143                  01396 normalirisopen
0143   15B2           01397         bsf     STEP1MENU       ; Used in entering menu mode
0144   1150           01398         bcf     IRIS_CLOSE      ; Don't do both
0145   14D0           01399         bsf     IRIS_OPEN
0146   2956           01400         goto    sendnonmotion
                      01401
0147                  01402 irisclose                       ; 0x91 Closes iris
0147   1CB2           01403         btfss   STEP2RESET      ; Last step of a home sequence?
0148   294E           01404         goto    normalirisclose
                      01405
0149   10B2           01406         bcf     STEP2RESET      ; Clear caller
014A   3022           01407         movlw   D_HOME
014B   00A9           01408         movwf   d_tilt_speed_M
014C   3007           01409         movlw   D_GOTO_PRESET
014D   29F4           01410         goto    set_d_command2
                      01411
014E                  01412 normalirisclose
014E   10D0           01413         bcf     IRIS_OPEN       ; Don't bo both
014F   1550           01414         bsf     IRIS_CLOSE
0150   2956           01415         goto    sendnonmotion
                      01416
```

```
                     01417         space 2
                     01418 ; Focus processing.
0151                 01419 focusnear                       ; 0x87 Focus near until Focus Far or Focus Stop
0151   1050          01420         bcf     FOCUS_FAR       ; Don't do both
0152   17D1          01421         bsf     FOCUS_NEAR
0153   2956          01422         goto    sendnonmotion
                     01423
0154                 01424 focusfar                        ; 0x88 Focus far until Focus Near or Focus Stop
0154   13D1          01425         bcf     FOCUS_NEAR      ; Don't do both
0155   1450          01426         bsf     FOCUS_FAR
0156                 01427 sendnonmotion
                     01428         ; Changed the status bits for focus, zoom or iris. Now get any active
                     01429         ; motion data and stick it into the message too.
0156   2257          01430         call    build_command_0
0157   29F5          01431         goto    sendcommand     ; And send it on its way
                     01432


MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 33
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     01433         space 2
                     01434 ; Zoom processing.
0158                 01435 zoomin                          ; 0x8A Zoom in until Zoom Out or Zoom Stop
0158   1351          01436         bcf     ZOOM_OUT        ; Don't do both
0159   16D1          01437         bsf     ZOOM_IN
015A   2956          01438         goto    sendnonmotion
                     01439
015B                 01440 zoomout                         ; 0x8B Zoom out until Zoom In or Zoom Stop
015B   1E32          01441         btfss   STEP2MENU       ; And check for any "focus" command
015C   2962          01442         goto    normalzoomout
                     01443
015D   1232          01444         bcf     STEP2MENU       ; Clear 2nd, 3rd of the partial caller
015E   305F          01445         movlw   D_MENU          ; Get a menu start command
015F   00A9          01446         movwf   d_tilt_speed_M  ; Stick it in as a preset
0160   3003          01447         movlw   D_SET_PRESET    ; Send a set preset command
0161   29F4          01448         goto    set_d_command2  ; Do it
                     01449
0162                 01450 normalzoomout
0162   1432          01451         bsf     STEP1RESET      ; This also might be for a dome reset
0163   12D1          01452         bcf     ZOOM_IN         ; Don't do both
0164   1751          01453         bsf     ZOOM_OUT
0165   2956          01454         goto    sendnonmotion
                     01455
                     01456
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 34
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     01457         page
                     01458 ; Pattern processing.
                     01459 ;
                     01460 ; Note d_tilt_speed_M always starts at zero
                     01461 ;
0166                 01462 pattern3                        ; 0xA2 Start defining Pattern 3
0166   0AA9          01463         incf    d_tilt_speed_M,f
0167                 01464 pattern2                        ; 0xA1 Start defining Pattern 2
0167   0AA9          01465         incf    d_tilt_speed_M,f
0168                 01466 pattern1                        ; 0xA0 Start defining Pattern 1
                     01467 ;       incf    d_tilt_speed_M  ; Increment the type
0168   0829          01468         movf    d_tilt_speed_M,w; Get the pattern number
0169   00B7          01469         movwf   patternwas_M    ; Save this pattern number
016A   301F          01470         movlw   D_START_PATTERN
016B   29F4          01471         goto    set_d_command2
                     01472
016C                 01473 patternend                      ; 0xB8 Recording (defining) a pattern
016C   0837          01474         movf    patternwas_M,w  ; Get the last pattern number
016D   00A9          01475         movwf   d_tilt_speed_M  ; Stick it into the command
016E   3021          01476         movlw   D_END_PATTERN
016F   29F4          01477         goto    set_d_command2
                     01478


                     01479         space 2
                     01480 ; Pattern execuition
                     01481 ;
                     01482 ; Note d_tilt_speed_M always starts at zero
```

```
0170   0AA9           01483 run3    incf    d_tilt_speed_M,f ; 0xB2 Run Pattern 3
0171   0AA9           01484 run2    incf    d_tilt_speed_M,f ; 0xB1 Run Pattern 2
0172                  01485 run1   ;incf    d_tilt_speed_M,f ; 0xB0 Run Pattern 1
0172   3023           01486         movlw   D_RUN_PATTERN
0173   29F4           01487         goto    set_d_command2
                      01488
                      01489         space 2
                      01490 ; Run the most recently defined pattern when requested
                      01491 ; (New command decoding 11FEB02. In older code this command was ignored.)
                      01492 ;
0174                  01493 runreview                       ; 0xB3 Run a newly defined pattern to review it
0174   0837           01494         movf    patternwas_M,w  ; Get the last pattern number
0175   00A9           01495         movwf   d_tilt_speed_M  ; Stick it into the command
0176   3023           01496         movlw   D_RUN_PATTERN
0177   29F4           01497         goto    set_d_command2
                      01498
                      01499
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 35
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01500         page
                      01501 ; Preset processing for fixed speed systems such as RC58.
                      01502 ;
                      01503 ; Note d_tilt_speed_M always starts at zero
0178   0AA9           01504 mark7   incf    d_tilt_speed_M,f ; 0xBB Mark the current position as Target 7
0179   0AA9           01505 mark6   incf    d_tilt_speed_M,f ; 0xBA Mark the current position as Target 6
017A   0AA9           01506 mark5   incf    d_tilt_speed_M,f ; 0xB9 Mark the current position as Target 5
017B   0AA9           01507 mark4   incf    d_tilt_speed_M,f ; 0xAB Mark the current position as Target 4
017C   0AA9           01508 mark3   incf    d_tilt_speed_M,f ; 0xAA Mark the current position as Target 3
017D   0AA9           01509 mark2   incf    d_tilt_speed_M,f ; 0xA9 Mark the current position as Target 2
017E   0AA9           01510 mark1   incf    d_tilt_speed_M,f ; 0xA8 Mark the current position as Target 1
017F   3003           01511         movlw   D_SET_PRESET
0180   29F4           01512         goto    set_d_command2
                      01513
                      01514         space 2
                      01515 ; Note d_tilt_speed_M always starts at zero
0181   0AA9           01516 target7 incf    d_tilt_speed_M,f ; 0xBE Go to Target 7
0182   0AA9           01517 target6 incf    d_tilt_speed_M,f ; 0xBD Go to Target 6
0183   0AA9           01518 target5 incf    d_tilt_speed_M,f ; 0xBC Go to Target 5
0184   0AA9           01519 target4 incf    d_tilt_speed_M,f ; 0xB7 Go to Target 4
0185   0AA9           01520 target3 incf    d_tilt_speed_M,f ; 0xB6 Go to Target 3
0186   0AA9           01521 target2 incf    d_tilt_speed_M,f ; 0xB5 Go to Target 2
0187   0AA9           01522 target1 incf    d_tilt_speed_M,f ; 0xB4 Go to Target 1
0188   3007           01523         movlw   D_GOTO_PRESET
0189   29F4           01524         goto    set_d_command2
                      01525
018A                  01526 reset                           ; 0xC6 Run default "Apple Peel" pattern, or
                      01527                                 ; for out use we just reset the dome.
                      01528 ;        call    default_d_values_1; Clear all my status too
018A   01D0           01529         clrf    this_command1_M ; Dump the status stuff
018B   01D1           01530         clrf    this_command2_M
018C   300F           01531         movlw   D_RESET         ; Do a full dome reset
018D   00A7           01532         movwf   d_command2_M    ; Stick command type in
018E   2483           01533         call    tospectra_2     ; Send it
018F   23DA           01534         call    printspectraout_4; Debugging
0190   30C8           01535         movlw   200             ; Initialize delay timer
0191   22FC           01536         call    delayv_0        ; Gives about 50 ms for the dome to
                      01537                                 ; . . start getting its act together
0192   2CBF           01538         goto    init_M          ; Now restart ourselves.
                      01539
                      01540 ;;ps
                      01541 ; getdometype sends the TXB-S422's type.
                      01542 ; We can be either a Speed or Ultra dome, this is most of the diffrence.
                      01543 ; More diffrences are where we have/do not have long line sizure times.
                      01544 ; (This routine was changed on 29JAN02.)
                      01545 ;;pe
0193                  01546 getdometype                     ; 0x94 Request dome type (poll)
0193   30F5           01547         movlw   PELCO_ULTRA_DOME; Get a Ultra dome type to send
0194   1D06           01548         btfss   NODELAY         ; Is this for an Ultra or Speed type dome
0195   30F8           01549         movlw   PELCO_SPEED_DOME; Get a Speed dome type to send
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 36
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
0196   2A23           01550         goto    sendthree
                      01551
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 37
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01552         page
                      01553 ;;ps
                      01554 ; Stop command processing. (All eight types.)
                      01555 ;;pe
0197                  01556 stoppan                         ; 0x83 Stop Panning
0197   10D1           01557         bcf    PAN_RIGHT
0198   1151           01558         bcf    PAN_LEFT
0199   3033           01559         movlw  PAN_DEFAULT      ; Get the default pan speed
019A   00D2           01560         movwf  this_pan_speed_M ; Set the speed in
019B   29A0           01561         goto   stoppantilt      ; . . stop it too
                      01562
019C                  01563 stoptilt                        ; 0x86 Stop Tilting
019C   11D1           01564         bcf    TILT_UP
019D   1251           01565         bcf    TILT_DOWN
019E   3024           01566         movlw  TILT_DEFAULT     ; Get the default tilt speed
019F   00D3           01567         movwf  this_tilt_speed_M; Set the tilt speed in
01A0                  01568 stoppantilt
01A0   0851           01569         movf   this_command2_M,w; Get current motion causing byte
                      01570                                 ; Remember that the other axis might be moving
01A1   391E           01571         andlw  D_MOTION         ; Dump non-motion bits
01A2   17B3           01572         bsf    IN_MOVEMENT      ; Reenable the movement flag
01A3   1903           01573         btfsc  status,z         ; Is any type of motion going on?
01A4                  01574 stopmovement
01A4   13B3           01575         bcf    IN_MOVEMENT      ; Nope, say that motion has stopped
01A5                  01576 minorstop                       ; Used in stopping focus, iris and zoom
01A5   1132           01577         bcf    LAST_FAST        ; Stop a possiable flip too
01A6   2956           01578         goto   sendnonmotion    ; Send it
                      01579
01A7                  01580 stopfast                        ; 0x8F Stop fast/fastest speeds
01A7                  01581 stopfaster                      ; 0x9B Stop faster speeds
01A7   083C           01582         movf    saved_pan_speed_M,w; Get the most recent pan speed
01A8   00D2           01583         movwf   this_pan_speed_M; Set the speed in
01A9   083D           01584         movf    saved_tilt_speed_M,w; Get the most recent tilt speed
01AA   00D2           01585         movwf   this_pan_speed_M; Set the tilt speed in
01AB   1132           01586         bcf    LAST_FAST        ; Stop a possiable flip too
01AC   2257           01587         call    build_command_0
01AD   29F5           01588         goto    sendcommand     ; Stop it
                      01589
01AE                  01590 stopfocus                       ; 0x89 Stop Focus
01AE   13D1           01591         bcf    FOCUS_NEAR
01AF   1050           01592         bcf    FOCUS_FAR
01B0   29A5           01593         goto   minorstop
                      01594
01B1                  01595 stopzoom                        ; 0x8C Stop Zoom
01B1   12D1           01596         bcf    ZOOM_IN
01B2   1351           01597         bcf    ZOOM_OUT
01B3   29A5           01598         goto   minorstop
                      01599
01B4                  01600 stopiris                        ; 0x92 Stop Iris offset adjustment
01B4   10D0           01601         bcf    IRIS_OPEN
01B5   1150           01602         bcf    IRIS_CLOSE
01B6   29A5           01603         goto   minorstop
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 38
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01604
01B7                  01605 stopall                         ; 0x93 Stop all movement
01B7   01D0           01606         clrf   this_command1_M
01B8   01D1           01607         clrf   this_command2_M
01B9   3033           01608         movlw  PAN_DEFAULT      ; Get the default pan speed
01BA   00D2           01609         movwf  this_pan_speed_M ; Set the speed in
01BB   3024           01610         movlw  TILT_DEFAULT     ; Get the default tilt speed
01BC   00D3           01611         movwf  this_tilt_speed_M; Set the tilt speed in
01BD   29A4           01612         goto   stopmovement
                      01613
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 39
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
                     01614         page
                     01615 ; Misc commands that have not been completed and those that never will be
01BE                 01616 unknownc1                     ; 0xC1 Unknown message
                     01617                               ; 12 bytes in, xx bytes back
01BE                 01618 propspeed                     ; 0xC3 Proportional speed pan or tilt movement
                     01619                               ; 4 bytes in, 3 bytes back (ACK)
                     01620                               ; Don't yet know what axis it is for
                     01621 ; These commands get ignored. Processing is the same as for unknowns.
01BE                 01622 unknown                       ; Processing for all unknown commands.
                     01623                               ; So far the list contains:
                     01624                               ; 0xA7, 0xBF,
                     01625                               ; 0xC2, 0xC4, 0xC5, 0xC7, 0xC8, and
                     01626                               ; all commands 0xCA --> 0xDF, 0xF1 --> 0xFF
01BE                 01627 ackdonecommand                ; Ack it and then finnish up
01BE   23FC          01628         call    sendack_5     ; Ack it
01BF                 01629 donecommand                   ; Most commands end up here
01BF   2803          01630         goto    again         ; Look for more
                     01631
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 40
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     01632         page
                     01633 ;;latex\subsubsubsectionindex{Variable speed control (VM96)}
                     01634 ;;ts
                     01635 ; Variable Speed Command contents
                     01636 ;
                     01637 ; Byte  Use
                     01638 ;  1    Camera ID
                     01639 ;  2    Op Code (0xC0)
                     01640 ;  3    Sub-Op Code for left (0x81), right (0x82), up (0x84) or down (0x85)
                     01641 ;  4    Speed in deg/sec
                     01642 ;  5    Checksum
                     01643 ;;te
                     01644 ;;ps
                     01645 ; There are several types of keyboards that generate variable speed
                     01646 ; commands. The one from American Dynamics generates a total of eight (8)
                     01647 ; different speeds, while others generate more. Thus we have to identify
                     01648 ; the matrix/keyboard type in order to determine what to do about
                     01649 ; converting the input speeds into output speeds.
                     01650 ;
                     01651 ; If we ever get in a pan speed of 0x01, 0x02, 0x50 or 0x63, the program
                     01652 ; logic will assume that it is not talking with an American Dynamics
                     01653 ; system and that it is talking to a Sensormatic system. Preliminary
                     01654 ; testing shows that the RC216 sends out the earlier speeds and that the
                     01655 ; AD2083/02 does not. This process of testing is not guaranteed to work
                     01656 ; forever but is the best that I can do to tell the difference between
                     01657 ; command sources. (Of course the RC58 system only sends fixed speed
                     01658 ; commands with modifiers.)
                     01659 ;;pe
01C0                 01660 variablespeed                 ; 0xC0 Variable speed control
                     01661                               ; 5 bytes in, 1 byte back (ACK)
01C0   0843          01662         movf    sproto3_M,w   ; Get direction to move in
01C1   3C81          01663         sublw   S_PROP_LEFT
01C2   1903          01664         btfsc   status,z
01C3   29D9          01665         goto    propleft
                     01666
01C4   0843          01667         movf    sproto3_M,w   ; Get direction to move in
01C5   3C82          01668         sublw   S_PROP_RIGHT
01C6   1903          01669         btfsc   status,z
01C7   29D7          01670         goto    propright
                     01671
01C8   0843          01672         movf    sproto3_M,w   ; Get direction to move in
01C9   3C84          01673         sublw   S_PROP_UP
01CA   1903          01674         btfsc   status,z
01CB   29D5          01675         goto    propup
                     01676
01CC   0843          01677         movf    sproto3_M,w   ; Get direction to move in
01CD   3C85          01678         sublw   S_PROP_DOWN
01CE   1D03          01679         btfss   status,z
01CF   2A0B          01680         goto    alldone       ; Not valid, quit
                     01681
01D0   1651          01682         bsf     TILT_DOWN
01D1                 01683 proptilt
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 41
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
01D1   0844           01684         movf    sproto4_M,w     ; Get tilt speed
01D2   00D3           01685         movwf   this_tilt_speed_M
01D3   227E           01686         call    check4rc216_0
01D4   28DE           01687         goto    sendtiltcommand
                      01688
01D5                  01689 propup
01D5   15D1           01690         bsf     TILT_UP
01D6   29D1           01691         goto    proptilt
                      01692
01D7                  01693 propright
01D7   14D1           01694         bsf     PAN_RIGHT
01D8   29DA           01695         goto    proppan
                      01696
01D9                  01697 propleft
01D9   1551           01698         bsf     PAN_LEFT
01DA                  01699 proppan
01DA   0844           01700         movf    sproto4_M,w      ; Get pan speed
01DB   00D2           01701         movwf   this_pan_speed_M
01DC   227E           01702         call    check4rc216_0
01DD   28A1           01703         goto    sendpancommand
                      01704
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 42
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      01705         page
                      01706 ;;latex\subsubsubsectionindex{AUX command processing}
                      01707 ;;ps
                      01708 ; Format of an output command is that the four LSBs of the opcode
                      01709 ; indicate what to to with the AUXes. Thus we have:
                      01710 ;
                      01711 ; The Esprit uses aux 1 to turn on and off its windshield wiper. Since
                      01712 ; neither the Spectra nor the Esprit use more than two auxes, I have
                      01713 ; redirected aux 4 to aux 1. This is so that a user using a TouchTracker
                      01714 ; can turn on/off the wiper by just hitting the B key. (The B key "toggles"
                      01715 ; the state of aux 4.) Because of this redirecting, TXB-S422 never sends
                      01716 ; an aux 4 out and since Pelco equipment does not use aux 3 it is not
                      01717 ; sent out either. It should be noted that the TXB-S422 does not save the
                      01718 ; state associated with aux 1/4. This state is saved internally in the
                      01719 ; controller. (This routine was changed on 11FEB02.)
                      01720 ;;pe
                      01721 ;;ts
                      01722 ; Where f = off, N = on, R = redirected to 0.
                      01723 ;
                      01724 ;        AUXes 3 2 1 0         AUXes 3 2 1 0
                      01725 ;        0xE0  R - f f         0xE8  R - f f
                      01726 ;        0xE1  R - f N         0xE9  R - f N
                      01727 ;        0xE2  R - N f         0xEA  R - N f
                      01728 ;        0xE3  R - N N         0xEB  R - N N
                      01729 ;        0xE4  R - f f         0xEC  R - f f
                      01730 ;        0xE5  R - f N         0xED  R - f N
                      01731 ;        0xE6  R - N f         0xEE  R - N f
                      01732 ;        0xE7  R - N N         0xEF  R - N N
                      01733 ;;te
01DE                  01734 output                         ; 0xE0 Set/clear output drivers
                      01735 ;;ps
                      01736 ; To properly process this command we have to send a total of two
                      01737 ; commands. One each for each bit which gets turned on or off. In
                      01738 ; protocol D we can only specify one bit to turn on or off with each
                      01739 ; command. The following table illustrates which command matches each
                      01740 ; bit:
                      01741 ;;pe
                      01742 ;;ts
                      01743 ; 0x09, D_SET_AUXILIARY
                      01744 ; 0x0A, D_CLEAR_AUXILIARY
                      01745 ;
                      01746 ; 0x01 = Aux 1, 0xE1
                      01747 ; 0x02 = Aux 2, 0xE2
                      01748 ; 0x03 = Aux 3, 0xE4, Not sent the Spectra/Esprit only have two auxes
                      01749 ; 0x04 = Aux 1, 0xE1, Toggling aux 1, caused by the controller
                      01750 ;;te
01DE   3009           01751         movlw   D_SET_AUXILIARY ; Turn on the AUX
01DF   1C42           01752         btfss   sproto2_M,bit0  ; Or turn it off
01E0   300B           01753         movlw   D_CLEAR_AUXILIARY; Turn off the AUX
01E1   00A7           01754         movwf   d_command2_M    ; Stick type in message
01E2   3001           01755         movlw   1               ; This is for AUX 1
01E3   2304           01756         call    doaux_3         ; Go and do it
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 43
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
```

```
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

01E4   23FC           01757         call    sendack_5       ; Send an ACK
01E5   23DA           01758         call    printspectraout_4; Show that command for debugging
                      01759
01E6   3009           01760         movlw   D_SET_AUXILIARY ; Turn on the AUX
01E7   1CC2           01761         btfss   sproto2_M,bit1  ; Or turn it off
01E8   300B           01762         movlw   D_CLEAR_AUXILIARY; Turn off the AUX
01E9   00A7           01763         movwf   d_command2_M    ; Stick type in message
01EA   3002           01764         movlw   2               ; This is for AUX 2
01EB   2304           01765         call    doaux_3         ; Go and do it
01EC   23DA           01766         call    printspectraout_4; Show that command for debugging
                      01767
                      01768 ; Logic for aux 3. Unfortunately we don't have an aux 3, so I commented it out
                      01769 ;        movlw   D_SET_AUXILIARY ; Turn on the AUX
                      01770 ;        btfss   sproto2_M,bit2  ; Or turn it off
                      01771 ;        movlw   D_CLEAR_AUXILIARY; Turn off the AUX
                      01772 ;        movwf   d_command2_M    ; Stick type in message
                      01773 ;        movlw   3               ; This is for AUX 3
                      01774 ;        call    doaux_3         ; Go and do it
                      01775 ;        call    printspectraout_4; Show that command for debugging
                      01776
01ED   3009           01777         movlw   D_SET_AUXILIARY ; Turn on the AUX
01EE   1DC2           01778         btfss   sproto2_M,bit3  ; Or turn it off
01EF   300B           01779         movlw   D_CLEAR_AUXILIARY ; Turn off the AUX
01F0   00A7           01780         movwf   d_command2_M    ; Stick type in message
                      01781 ;        movlw   4               ; This is for AUX 1
01F1   3001           01782         movlw   1               ; This is for AUX 1
01F2   2304           01783         call    doaux_3         ; Go and do it
01F3   29BF           01784         goto    donecommand     ; An we are done
                      01785
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 44
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE


                      01786         page
                      01787 ;;latex\subsubsubsectionindex{Command processing, timing}
                      01788 ;;ps
                      01789 ; The Sensormatic protocol sends new commands almost immediately
                      01790 ; after it receives an ACK, or it retransmits the last command
                      01791 ; about 45 ms after starting a command (time information is from
                      01792 ; measurements made on an AD2083/02). Thus it is necessary to
                      01793 ; send an ACK, AFTER sending a command to the Spectra. Remember
                      01794 ; that when sending data to the Spectra we have to bit-bang the
                      01795 ; data out and thus can not receive any new data in from the head
                      01796 ; end while sending the Spectra data.
                      01797 ;
                      01798 ; Having a 29.6 ms "dead window" makes it so that we miss many
                      01799 ; messages. However by delaying the ACK until AFTER sending a
                      01800 ; command to the Spectra, means that we almost always "get" all
                      01801 ; commands from the controller.
                      01802 ;
                      01803 ; WARNING: If any controller retransmitts commands faster than
                      01804 ; the AD2083/02 does there may be a problem. Currently there is
                      01805 ; about 9 ms of "slack", but different controllers may retransmit
                      01806 ; faster, etc.
                      01807 ;
                      01808 ; A special case has been detected with the RC58 at Sears,
                      01809 ; Fresno. On the first bunch of data that I acquired the switch
                      01810 ; only sent data out once if it got an acknowledge. Since the
                      01811 ; first time or two at Sears Fresno, the RC58 ALWAYS sends out
                      01812 ; each command three times. Even if it gets an acknowledge! This
                      01813 ; causes serious problems with this program's timing.  (This is
                      01814 ; one of the nice "features" of using one companies (American
                      01815 ; Dynamics) pieces of equipment (AD2083/02) to debug your code
                      01816 ; with, when actually using another companies (Sensormatic's)
                      01817 ; equipment. Even though one bought the other out, there may be
                      01818 ; much older equipment in the field that we have to interface
                      01819 ; with.) To get around these problems I wrote a simulator (sim58)
                      01820 ; which runs under GWBASIC and does a very, very simple
                      01821 ; simulation of what an RC58 does. The primary thing that it does
                      01822 ; is send predictable commands quickly to a dome. Another
                      01823 ; advantage is that it is easily changed to do something different
                      01824 ; on each run.
                      01825 ;
                      01826 ; Program logic used to be to send the ACK first and then do the
                      01827 ; rest.
                      01828 ;;pe
```

```
01F4                    01829 set_d_command2                  ; Allows word 4 to be changed
01F4   00A7             01830         movwf   d_command2_M    ; Stick command type in
01F5                    01831 sendcommand                     ; D protocol word 4 is already loaded
01F5   1134             01832         bcf     BYTE3           ; Clear the "extra" message flag
01F6   1E33             01833         btfss   CLEAR_DISPLAY   ; Should we clear the Spectra screen?
01F7   2A0A             01834         goto    iscleared       ; Nope, is cleared
                        01835
01F8   1233             01836         bcf     CLEAR_DISPLAY   ; Clear the caller
01F9   0826             01837         movf    d_command1_M,w  ; Save the current command
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 45
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

01FA   00B8             01838         movwf   save3_M
01FB   0827             01839         movf    d_command2_M,w
01FC   00B9             01840         movwf   save4_M
01FD   0828             01841         movf    d_pan_speed_M,w
01FE   00BA             01842         movwf   save5_M
01FF   0829             01843         movf    d_tilt_speed_M,w
0200   00BB             01844         movwf   save6_M
0201   2250             01845         call    blankit_3       ; Clear the screen
0202   0838             01846         movf    save3_M,w       ; Restore the old command
0203   00A6             01847         movwf   d_command1_M
0204   0839             01848         movf    save4_M,w
0205   00A7             01849         movwf   d_command2_M
0206   083A             01850         movf    save5_M,w
0207   00A8             01851         movwf   d_pan_speed_M
0208   083B             01852         movf    save6_M,w
0209   00A9             01853         movwf   d_tilt_speed_M
                        01854
                        01855 ;;ps
                        01856 ; Note the test for BYTE3. This test is intended to allow us to receive a
                        01857 ; message in the middle of bit-banging data out to the Spectra.
                        01858 ;;pe
020A                    01859 iscleared
020A   2483             01860         call    tospectra_2     ; Now send command out to the Spectra
020B                    01861 alldone
020B   23DA             01862         call    printspectraout_4; Show that command for debugging
020C   1D34             01863         btfss   BYTE3           ; Has a message come in while sending to Spectra
020D   29BE             01864         goto    ackdonecommand  ; Finish up
                        01865
020E   1134             01866         bcf     BYTE3           ; Clear the "extra" message flag
020F   2813             01867         goto    decodeinput_M   ; Decode the new message
                        01868
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 46
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                        01869         page
                        01870 ;;latex\subsubsubsectionindex{Other command processing}
                        01871 ;;ps
                        01872 ; Whenever we get an On Air message say that we are in boundary #1. (Even
                        01873 ; if we don't know where we really are. Some controller might be looking
                        01874 ; for it.)
                        01875 ;;pe
0210                    01876 onair                           ; 0x9E Set On Air status
0210   30B0             01877         movlw  S_BOUNDARY0CROSSED; Boundary crossing #1
0211   2A23             01878         goto   sendthree        ; Send it and quit
                        01879
                        01880 ; sendack is used to send an acknowledge to the controller
0212                    01881 onairreset                      ; 0x9F Reset On Air status
0212                    01882 patternaccept                   ; 0xA3 Accept the new pattern
0212   29BE             01883 sendack goto    ackdonecommand  ; Finish up
                        01884
                        01885 ;;ps
                        01886 ; Unknown commands 0x80, 0x96, 0x9C, 0x9D and 0xF0 always sends an ACK
                        01887 ; and then sends a three byte message. The contents of the message are
                        01888 ; unknown, but I have copied them here from what I saw in the field and
                        01889 ; what I saw when I sent commands to SensorMatic SpeedDomes and
                        01890 ; UltraDomes. In the field (Sears Fresno) I used an RC58 controller.
                        01891 ;;pe
                        01892 ;;ts
                        01893 ;   Command Generates  Processed by
                        01894 ;     0x80 = 0xC4      unknown80
                        01895 ;     0x96 = 0xC0      unknown96
                        01896 ;     0x9C = 0xC4      boundarystart
                        01897 ;     0x9D = 0xB0      boundarymark
                        01898 ;     0xF0 = 0xB5      terminatepattern
```

```
                     01899 ;;te
0213                 01900 unknown80
0213  23FC           01901         call    sendack_5       ; Ack that unknown
0214  30C4           01902         movlw   0xC4            ; . . Get an unknown response
0215  2A23           01903         goto    sendthree       ; . . . . Send three byte unknown response
                     01904
0216                 01905 unknown96
0216  23FC           01906         call    sendack_5       ; Ack and
0217  30C0           01907         movlw   0xC0            ; . . respond
0218  2A23           01908         goto    sendthree       ; . . . . do it
                     01909
0219                 01910 boundarystart                  ; 0x9C Start boundary definition.
0219  23FC           01911         call    sendack_5       ; Ack and
021A  30C4           01912         movlw   0xC4            ; . . respond
021B  2A23           01913         goto    sendthree       ; . . . . do it
                     01914
021C                 01915 boundarymark                   ; 0x9D Marks the current position as a boundary
021C  23FC           01916         call    sendack_5       ; Ack and
021D  30B0           01917         movlw   0xB0            ; . . respond
021E  2A23           01918         goto    sendthree       ; . . . . do it
                     01919
021F                 01920 terminatepattern               ; 0xF0 Terminate the current pattern
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 47
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

021F  23FC           01921         call    sendack_5       ; Ack and
0220  30B5           01922         movlw   0xB5            ; . . respond
0221  2A23           01923         goto    sendthree       ; Should send an ack and a pattern end command
                     01924                                 ; . . to the controller, but I don't know what
                     01925                                 ; . . to send to the Spectra.
                     01926
                     01927
                     01928 ; getalarms is used to send the status of this unit's alarms
0222                 01929 getalarms                      ; 0x95 Request status of alarm inputs
0222  3000           01930         movlw   0x00            ; Temp fake value, an UltraDome reports 0x0F
                     01931                                 ; . . While a SpeedDome reports 0x00
0223                 01932 sendthree
0223  245C           01933         call    threebytemessage_6 ; Send it
0224  29BF           01934         goto    donecommand
                     01935
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 48
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     01936         page
                     01937 ;;latex\subsectionindex{System Subroutines}
                     01938 ;;ps
                     01939 ; System subroutines.
                     01940 ;
                     01941 ; There are several "level"s of subroutines.  The level # indicates its
                     01942 ; level.  A subroutine may only call lower level routines, NEVER those at
                     01943 ; its level or above. As a convention all subroutine names end with their
                     01944 ; calling level.  Subroutines that end with _M are main line extensions.
                     01945 ;;pe
                     01946
                     01947 ;;latex\subsubsectionindex{arrowout_3}
                     01948 ;;ps
                     01949 ; arrowout_3 Is used to stick an arrow out in debug mode.
                     01950 ; (This routine was changed on 29JAN02.)
                     01951 ;;pe
0225                 01952 arrowout_3
0225  2456           01953         call    startout_1      ; Every time we stick out an arrow we are
                     01954                                 ; beginning to xmit to the controller. So
                     01955                                 ; I put this here even though it has no good
                     01956                                 ; reason to be here.
0226  1A86           01957         btfsc   DEBUG_MODE_ON   ; Do we echo this character?
0227  2A2F           01958         goto    noarrow
                     01959
0228  302D           01960         movlw   "-"     ; Clearly indicate that this is an ACK
0229  2406           01961         call    senddebugbyte_1
022A  302D           01962         movlw   "-"
022B  2406           01963         call    senddebugbyte_1
022C  303E           01964         movlw   ">"
022D  2406           01965         call    senddebugbyte_1
022E  224D           01966         call    blank_2
022F                 01967 noarrow
022F  0008           01968         return
```

```
                      01969
                      01970         space 1
                      01971 ;;latex\subsubsectionindex{bin2hex_0}
                      01972 ;;ps
                      01973 ; bin2hex_0 is used to take a binary byte in the w register and to convert
                      01974 ; it into two ASCII bytes.  The upper half will be in hexupper_0 and the
                      01975 ; lower half will be in hexlower_0. (What surprising places to stuck um.)
                      01976 ; No additional core locations are used.
                      01977 ;;pe
0230                  01978 bin2hex_0
0230  00AD            01979         movwf   hexlower_0      ; Initialize lower half
0231  00AE            01980         movwf   hexupper_0      ; Initialize upper half
0232  0EAE            01981         swapf   hexupper_0,f    ; Flip around
0233  300F            01982         movlw   0x0F            ; Get single nibble mask
0234  05AE            01983         andwf   hexupper_0,f    ; Dump slop
0235  05AD            01984         andwf   hexlower_0,f    ; More slop dumping
0236  082E            01985         movf    hexupper_0,w    ; Get upper byte first
0237  3830            01986         iorlw   "0"             ; Do 0 --> 9
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 49
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0238  00AE            01987         movwf   hexupper_0      ; Save it
0239  303A            01988         movlw   ":"             ; Get breaking point between letters/numbers
023A  022E            01989         subwf   hexupper_0,w    ; Is it .gt. 9
023B  1C03            01990         btfss   status,c        ; Did we need a carry cuz its too big
023C  2A3F            01991         goto    gotfirst        ; Nope, are done
                      01992
023D  3007            01993         movlw   "A"-":"         ; Get ASCII start for letters
023E  07AE            01994         addwf   hexupper_0,f    ; Convert upper nibble
023F                  01995 gotfirst
023F  082D            01996         movf    hexlower_0,w    ; Do lower nibble
0240  3830            01997         iorlw   "0"             ; Do 0 --> 9
0241  00AD            01998         movwf   hexlower_0      ; Save it
0242  303A            01999         movlw   ":"             ; Get breaking point between letters/numbers
0243  022D            02000         subwf   hexlower_0,w    ; Is it .gt. 9
0244  1C03            02001         btfss   status,c
0245  2A48            02002         goto    donconvert
                      02003
0246  3007            02004         movlw   "A"-":"         ; Get ASCII start for letters
0247  07AD            02005         addwf   hexlower_0,f
0248                  02006 donconvert
0248  3400            02007         retlw   0               ; All done converting from 1 binary to 2 ASCII
                      02008


                      02009         space 1
                      02010 ;;latex\subsubsectionindex{bitdelay_0}
                      02011 ;;ps
                      02012 ; bitdelay_0 is used to sit in a tight loop for one bit time. How long to
                      02013 ; wait is contained in w on entry.
                      02014 ;;pe
0249                  02015 bitdelay_0
0249  00D6            02016         movwf   timeout_0       ; Init counter for delay
024A  0BD6            02017 bitwait decfsz  timeout_0,f     ; Done delaying?
024B  2A4A            02018         goto    bitwait         ; Nope
                      02019
024C  3400            02020         retlw   0               ; Return, its time
                      02021


                      02022         space 1
                      02023 ;;latex\subsubsectionindex{blank_2}
                      02024 ;;ps
                      02025 ; blank_2 is used to output a blank character in debug mode.
                      02026 ;;pe
024D                  02027 blank_2
024D  3020            02028         movlw   BLANK           ; Blank
024E  2406            02029         call    senddebugbyte_1 ; Stick out a nice spacer
024F  3400            02030         retlw   0
                      02031


                      02032         space 1
                      02033 ;;latex\subsubsectionindex{blankit_3}
                      02034 ;;ps
                      02035 ; blankit_3 is used to clear the Spectra screen.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 50
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
2371
2372                        02036 ;;pe
2373  0250                  02037 blankit_3
2374  0250  01A6           02038         clrf    d_command1_M
2375  0251  3017           02039         movlw   D_CLEAR_SCREEN          ; Command to clear the screen
2376  0252  00A7           02040         movwf   d_command2_M
2377  0253  01A8           02041         clrf    d_pan_speed_M
2378  0254  01A9           02042         clrf    d_tilt_speed_M
2379  0255  2483           02043         call    tospectra_2
2380  0256  0008           02044         return
2381                        02045
2382
2383                        02046         space 1
2384                        02047 ;;latex\subsubsection{build_command_0}
2385                        02048 ;;ps
2386                        02049 ; build_command_0 is used to make a full command to send to the Spectra.
2387                        02050 ; It does this by combining various motion control bytes into one
2388                        02051 ; D-protocol command.
2389                        02052 ;
2390                        02053 ; The following D-protocol bytes are built up:
2391                        02054 ;;pe
2392                        02055 ;;ts
2393                        02056 ;       d_command1_M
2394                        02057 ;       d_command2_M
2395                        02058 ;       d_pan_speed_M
2396                        02059 ;       d_tilt_speed_M
2397                        02060 ;;te
2398  0257                  02061 build_command_0
2399  0257  0850           02062         movf    this_command1_M,w       ; Note that all four important
2400  0258  00A6           02063         movwf   d_command1_M            ; . D-protocol bytes
2401  0259  0851           02064         movf    this_command2_M,w       ; . are "mirrored".
2402  025A  00A7           02065         movwf   d_command2_M            ; . This is done so that they
2403  025B  0852           02066         movf    this_pan_speed_M,w      ; . will not be damaged by
2404  025C  00A8           02067         movwf   d_pan_speed_M           ; . "preset", and similar
2405  025D  0853           02068         movf    this_tilt_speed_M,w     ; . commands.
2406  025E  00A9           02069         movwf   d_tilt_speed_M
2407  025F  0008           02070         return
2408                        02071
2409
2410                        02072         space 1
2411                        02073 ;;latex\subsubsectionindex{byteread_1}
2412                        02074 ;;ps
2413                        02075 ; byteread_1 is used to immediately read a byte and return with it in w.
2414                        02076 ; This routine is intended to sandwich reads into the middle of writing
2415                        02077 ; to the Spectra. Before calling BYTE1 and BYTE2 must be cleared. The
2416                        02078 ; received message is limited to three bytes and is stored in sproto1_M
2417                        02079 ; thru sproto3_M. Messages longer than 3 bytes are messed up and not
2418                        02080 ; saved correctly. (Had to make this a level 1 subroutine, instead of
2419                        02081 ; putting it into sendspectra_0 because I'm worried about not having
2420                        02082 ; enough levels of return stack.)  A check is made of what might be the
2421                        02083 ; first byte to be sure that it matches "our" Spectra's address. If the
2422                        02084 ; address check fails, we don't advance past the first byte.
2423                        02085 ;;pe
2424  MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 51
2425  $Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
2426  Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
2427  LOC  OBJECT CODE     LINE SOURCE TEXT
2428    VALUE
2429
2430  0260                  02086 byteread_1
2431  0260  1E8C           02087         btfss   pir1,rcif       ; Got anything in?
2432  0261  2A7D           02088         goto    nothinghere     ; Nope
2433                        02089
2434  0262                  02090 readsomemore
2435  0262  22A7           02091         call    ckioerrs_0      ; Get byte and see if it's OK
2436  0263  19B3           02092         btfsc   INPUT_ERROR     ; Had an I/O error?
2437  0264  2A7D           02093         goto    nothinghere     ; Yep
2438                        02094
2439  0265  1C34           02095         btfss   BYTE1           ; First successful read?
2440  0266  2A6D           02096         goto    try2            ; Nope, already have one
2441                        02097
2442  0267  00C1           02098         movwf   sproto1_M       ; Yep, get first byte
2443  0268  023F           02099         subwf   spectraaddress_M,w; Does it match our Spectra's address
2444  0269  1D03           02100         btfss   status,z        ; Check
2445  026A  2A71           02101         goto    ck4more         ; Nope
2446                        02102
2447  026B  1434           02103         bsf     BYTE1           ; Set first byte flag
2448  026C  2A71           02104         goto    ck4more         ; Might exit
2449                        02105
2450  026D                  02106 try2
2451  026D  1CB4           02107         btfss   BYTE2           ; Is this the second byte?
2452  026E  2A74           02108         goto    store3          ; Nope, third or more
```

```
                        02109
026F   00C2             02110         movwf   sproto2_M       ; Save byte 2
0270   14B4             02111         bsf     BYTE2           ; Say all the rest go to the same saving location
0271                    02112 ck4more
0271   1A8C             02113         btfsc   pir1,rcif       ; Got anything in?
0272   2A62             02114         goto    readsomemore    ; Nope
                        02115
0273   2A7D             02116         goto    nothinghere     ; Finished reading this in
                        02117
0274                    02118 store3
0274   00C3             02119         movwf   sproto3_M       ; Past byte 2, they all get stored here
0275   1534             02120         bsf     BYTE3           ; Say we got it mostly all in
0276   0741             02121         addwf   sproto1_M,w     ; This was byte three, now check the checksum
0277   0742             02122         addwf   sproto2_M,w     ; All good checksums overflow and leave 0 in w
0278   1903             02123         btfsc   status,z        ; Check the checksum
0279   2A7D             02124         goto    nothinghere     ; Its good, finush it all
                        02125
027A   1034             02126         bcf     BYTE1           ; It's bad, start over
027B   10B4             02127         bcf     BYTE2           ; . . get all the status
027C   1134             02128         bcf     BYTE3           ; . . . . and flag bits
027D                    02129 nothinghere                     ; Either nothing new here yet, or it's an error
027D   0008             02130         return                  ; Back to the pore user
                        02131
                        02132


                        02133         space 1
                        02134 ;;latex\subsubsection{check4rc216_0}
                        02135 ;;ps
                        02136 ; check4rc216_0 is used to determine if this is an RC216 type matrix.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 52
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                        02137 ; This is done by examinaning speed control commands and when one is
                        02138 ; found that is unique to the RC216, then a flag is set. Once the flag,
                        02139 ; RC216_MODE is set, there is no way to reset it other than by using a
                        02140 ; full power cycle of the unit.
                        02141 ;
                        02142 ; Unique speeds are 1, 2, 80 and 99 degree/sec.
                        02143 ;
                        02144 ; The RC216, and possiblely others, generate many different pan speed
                        02145 ; commands, at least 30. So the problem here is to translate the
                        02146 ; non-linear Sensormatic commands to the non-linear Pelco commands.
                        02147 ;
                        02148 ; In preliminary testing the following variable pan speeds were observed:
                        02149 ; 1 --> 20, 22, 24, 32, 48, 64, 80 and 99. (More have since been found.)
                        02150 ;
                        02151 ; These speeds represent dome speeds in different degrees per second.
                        02152 ; There is a potential for having 99 different speeds while Pelco only has
                        02153 ; about 53 different speeds. (It must be remembered that some speeds are
                        02154 ; repeated inside the Spectra and others are blocked to avoid mechanical
                        02155 ; resonance induced noise.)
                        02156 ;
                        02157 ; The original logic of determining the speeds to use for the Spectra
                        02158 ; when controlled by an RC216 was to take advantage of the fact that the
                        02159 ; Spectra repeats the first nine pan speed values and to then simplify
                        02160 ; the conversion process quite a lot by always dividing the input value
                        02161 ; by 2 (the input range is now 0 --> 48) and then adding eight to it
                        02162 ; giving a low Spectra speed of 0.5/sec and a high speed of  of
                        02163 ; 41.9/deg per second. As this last speed is a little slow (it's NOT
                        02164 ; turbo speed) we do a special check to see if the input speed is 0x40,
                        02165 ; or more, which we translate into a turbo pan speed of 0x40.
                        02166 ;
                        02167 ; In response to customer comments about the system being "sluggish" the
                        02168 ; alogrithum has been changed to take the input speed and multiply it by
                        02169 ; 1/4, 1/2, 3/4 or 1 before adding 8 to it. The rest of the logic has
                        02170 ; not changed.
                        02171 ;;pe
                        02172 ;;ts
                        02173 ; Called with:
                        02174 ; w = speed value to check
                        02175 ;
                        02176 ; Changes:
                        02177 ; temp1_0
                        02178 ;
                        02179 ; Returns with;
                        02180 ; RC216_MODE turned on if this is an "active" speed. Is not affected
                        02181 ;            if speed is "inactive", i.e. this routine does not turn
                        02182 ;            off RC216_MODE.
                        02183 ;;te
```

```
027E                    02184 check4rc216_0
027E   00CE              02185         movwf   temp1_0         ; Save it
                         02186         ifdef   TEST_PROPORTIONAL
                         02187           goto  setrc216        ; Force RC216 mode for testing
                         02188         endif ; ifdef TEST_PROPORTIONAL
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 53
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

027F   034E              02189         decf    temp1_0,w       ; See if this is pan speed 1
0280   1903              02190         btfsc   status,z        ; Speed 1 is unique to the RC216
0281   2A8B              02191         goto    setrc216
                         02192
0282   3C01              02193         sublw   1               ; Is this a pan speed 2
0283   1903              02194         btfsc   status,z        ; Check
0284   2A8B              02195         goto    setrc216        ; Yep
                         02196
0285   3E4E              02197         addlw   0x50-2          ; How about 0x50
0286   1903              02198         btfsc   status,z
0287   2A8B              02199         goto    setrc216
                         02200
0288   3E13              02201         addlw   0x63-0x50       ; Or even 0x63
0289   1903              02202         btfsc   status,z
028A   2A8C              02203         goto    notrc216speed
                         02204
028B                    02205 setrc216
028B   16B3              02206         bsf     RC216_MODE      ; This is a variable speed RC216 type command
028C                    02207 notrc216speed
028C   0008              02208         return
                         02209

                         02210         space 1
                         02211 ;;latex\subsubsectionindex{checkrate_0}
                         02212 ;;ps
                         02213 ; checkrate_0 is used to check bits 1 and 2 on the switch and to modify
                         02214 ; the input value accordingly.
                         02215 ;
                         02216 ; On input and return w = what to work with/return
                         02217 ;
                         02218 ; Currently January 2002, there has been no demonstrated need for an
                         02219 ; address switch, so I have decided to use the two leist significant bits
                         02220 ; of the address switch to provide access to diffrent pan/tilt speeds for
                         02221 ; use with RC216 type controllers.
                         02222 ;
                         02223 ; The following table holds:
                         02224 ;;pe
                         02225 ;;ts
                         02226 ;     Bit 0 Bit 1 Speed range
                         02227 ;       0     0   Full speed, this is slightly faster than Sensormatic pan
                         02228 ;                 and tilt speeds.
                         02229 ;
                         02230 ;       1     0   3/4 speed, this is somewhat less than Sensormatic pan
                         02231 ;                 and tilt speeds.
                         02232 ;
                         02233 ;       0     1   1/2 speed, this speed behavies in a "sluggish"
                         02234 ;                 manner. And is provided for improved control of the
                         02235 ;                 Spectra.
                         02236 ;
                         02237 ;       1     1   1/4 speed, this is a quite slow speed that is provided
                         02238 ;                 for precise control of the Spectra.
                         02239 ;
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 54
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                         02240 ; Calling:
                         02241 ;       w has the current speed value
                         02242 ;
                         02243 ; Uses:
                         02244 ;       temp1_0
                         02245 ;       temp2_0
                         02246 ;
                         02247 ; Return:
                         02248 ;       w has a modified, or not, speed value in it
                         02249 ;
                         02250 ;;te
028D                    02251 checkrate_0
028D   00CE              02252         movwf   temp1_0         ; Save
```

```
028E   1C86           02253         btfss   SPEEDCONTROL1   ; Slow range? Switch open/off = 1 = high
028F   2A93           02254         goto    slowhalf        ; Yep, bits come in backward
                       02255
0290   1806           02256         btfsc   SPEEDCONTROL0   ; Speed highest or next to highest?
0291   2AA6           02257         goto    returnvalue     ; Highest, w still has calling value
                       02258
0292   2A9A           02259         goto    threequartervalue; Do it just a little slow.
                       02260
0293                   02261 slowhalf
0293   1806           02262         btfsc   SPEEDCONTROL0   ; Speed lowest or next to lowest?
0294   2AA4           02263         goto    onehalfvalue    ; Try for soso-slow
                       02264
0295   1003           02265         bcf     status,c        ; Time for slug slow, 1/4 speed
0296   0CCE           02266         rrf     temp1_0,f       ; Divide by 2
0297   1003           02267         bcf     status,c
0298   0C4E           02268         rrf     temp1_0,w       ; Get 1/4th input value
0299   2AA6           02269         goto    returnvalue     ; Return to caller
                       02270
029A                   02271 threequartervalue              ; Use 3/4 speed values
029A   084E           02272         movf    temp1_0,w       ; Get original value
029B   00CF           02273         movwf   temp2_0         ; Save it
029C   1003           02274         bcf     status,c        ; Don't let carry get in the way
029D   0CCF           02275         rrf     temp2_0,f       ; Divide input speed by 2
029E   1003           02276         bcf     status,c        ; No carries messing it up
029F   0CCF           02277         rrf     temp2_0,f       ; Divide by 2 again to get 1/4 of the input
02A0   1003           02278         bcf     status,c        ; Carry again, just messed me up
02A1   0C4E           02279         rrf     temp1_0,w       ; Divide saved input by 2 to get 1/2 of input
02A2   074F           02280         addwf   temp2_0,w       ; Get input times 3/4 into w (1/2 + 1/4 = 3/4)
02A3   2AA6           02281         goto    returnvalue     ; Go on back to the caller
                       02282
02A4                   02283 onehalfvalue
02A4   1003           02284         bcf     status,c        ; No supprises
02A5   0C4E           02285         rrf     temp1_0,w       ; Divide saved input by 2
02A6                   02286 returnvalue
02A6   0008           02287         return                  ; And all done
                       02288
                       02289         space 1
                       02290 ;;latex\subsubsectionindex{ckioerrs_0}
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 55
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                       02291 ;;ps
                       02292 ; ckioerrs_0 is used to check the head end serial connection for errors
                       02293 ; and to get the data.
                       02294 ;
                       02295 ; On return w holds the data byte. Also temp1_0 holds the most recent
                       02296 ; byte.
                       02297 ;;pe
02A7                   02298 ckioerrs_0
02A7   11B3           02299         bcf     INPUT_ERROR     ; Clear the input error flag
02A8   1C98           02300         btfss   rcsta,oerr      ; Test for overrun error
02A9   2AB1           02301         goto    nooverrun       ; It might be good, no errors yet
                       02302
02AA   15B3           02303         bsf     INPUT_ERROR     ; Say that we had an error
02AB   0818           02304         movf    rcsta,w         ; Get status
02AC   00D5           02305         movwf   thisioerror_0   ; Save it
02AD   081A           02306         movf    rcreg,w         ; Flush the FIFO
02AE   081A           02307         movf    rcreg,w         ; . . both parts
02AF   1218           02308         bcf     rcsta,cren      ; Clear the
02B0   1618           02309         bsf     rcsta,cren      ; . . overrun error
02B1                   02310 nooverrun
02B1   1D18           02311         btfss   rcsta,ferr      ; Check on framing error
02B2   2ABA           02312         goto    frameok         ; Nice, no framing error
                       02313
02B3   15B3           02314         bsf     INPUT_ERROR     ; Say that we had an error
02B4   0818           02315         movf    rcsta,w         ; Get status
02B5   00D5           02316         movwf   thisioerror_0   ; Save it
02B6   081A           02317         movf    rcreg,w         ; Flush the FIFO
02B7   081A           02318         movf    rcreg,w         ; . . both parts
02B8   1218           02319         bcf     rcsta,cren      ; Clear the
02B9   1618           02320         bsf     rcsta,cren      ; . . framing error
02BA   081A           02321 frameok movf    rcreg,w         ; Get data into w
02BB   00CE           02322         movwf   temp1_0         ; Save the data byte too
02BC   0008           02323         return                  ; Don't messup w or c
                       02324
                       02325         space 1
                       02326 ;;latex\subsubsectionindex{cleareememory_1}
```

```
                     02327 ;;ps
                     02328 ; cleareememory_1 is used to see if EE memory has been cleared. It is
                     02329 ; assumed that it is not cleared if the 32 bit constant 0xDEADBEEF is
                     02330 ; not in locations EE_SET_MEM0 through EE_SET_MEM3. Anything else will result
                     02331 ; in three locations being set as follows:
                     02332 ;;pe
                     02333 ;;ts
                     02334 ; Location     Initial value
                     02335 ;
                     02336 ; EE_PRESET     0
                     02337 ;;te
02BD                 02338 cleareememory_1
02BD   3040          02339         movlw   EE_SET_MEM0     ; Get the first address to read from
02BE   2310          02340         call    eeread_0        ; Read an EE byte
02BF   3CDE          02341         sublw   0xDE            ; Is this the first ID byte
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 56
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

02C0   1D03          02342         btfss   status,z        ; Check
02C1   2AD2          02343         goto    clearee         ; Not initialized yet, do it now
                     02344
02C2   3041          02345         movlw   EE_SET_MEM1
02C3   2310          02346         call    eeread_0        ; Read an EE byte
02C4   3CAD          02347         sublw   0xAD            ; Is this the second ID byte
02C5   1D03          02348         btfss   status,z        ; Check
02C6   2AD2          02349         goto    clearee         ; Not initialized yet, do it now
                     02350
02C7   3042          02351         movlw   EE_SET_MEM2
02C8   2310          02352         call    eeread_0        ; Read an EE byte
02C9   3CBE          02353         sublw   0xBE            ; Is this the third ID byte
02CA   1D03          02354         btfss   status,z        ; Check
02CB   2AD2          02355         goto    clearee         ; Not initialized yet, do it now
                     02356
02CC   3043          02357         movlw   EE_SET_MEM3
02CD   2310          02358         call    eeread_0        ; Read an EE byte
02CE   3CEF          02359         sublw   0xEF            ; Is this the last ID byte
02CF   1903          02360         btfsc   status,z        ; Check
02D0   2AD2          02361         goto    clearee         ; Not initialized yet, do it now
                     02362
02D1                 02363 doneclearing
                     02364         ; Initialize fast copies of EE memory values
02D1   0008          02365         return                  ; All initialized, return
                     02366
02D2                 02367 clearee                         ; Not yet cleared/initialized
02D2   3040          02368         movlw   EE_SET_MEM0     ; Get first location
02D3   00AB          02369         movwf   eewritehere_0   ; Set up call
02D4   30DE          02370         movlw   0xDE            ; Get ID byte
02D5   231B          02371         call    eewrite_0       ; Write it in
02D6   3041          02372         movlw   EE_SET_MEM1
02D7   00AB          02373         movwf   eewritehere_0
02D8   30AD          02374         movlw   0xAD            ; 0xAD
02D9   231B          02375         call    eewrite_0
02DA   3042          02376         movlw   EE_SET_MEM2
02DB   00AB          02377         movwf   eewritehere_0
02DC   30BE          02378         movlw   0xBE            ; 0xBE
02DD   231B          02379         call    eewrite_0
02DE   3043          02380         movlw   EE_SET_MEM3
02DF   00AB          02381         movwf   eewritehere_0
02E0   30EF          02382         movlw   0xEF            ; 0xEF
02E1   231B          02383         call    eewrite_0
                     02384         ; Set the flags, now stick the data
02E2   3022          02385         movlw   EE_PRESET        ; First the preset ID #
02E3   00AB          02386         movwf   eewritehere_0
02E4   3000          02387         movlw   0x00
02E5   231B          02388         call    eewrite_0
02E6   2AD1          02389         goto    doneclearing
                     02390


                     02391         space 1
                     02392 ;;latex\subsubsectionindex{crlf_2}
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 57
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE


                     02393 ;;ps
                     02394 ; crlf_2 is used in debug mode to stick out a new-line sequence.
                     02395 ;;pe
```

```
02E7                02396 crlf_2
02E7   300D         02397         movlw   CR              ; carriage return
02E8   2406         02398         call    senddebugbyte_1
02E9   300A         02399         movlw   LF              ; line feed
02EA   2406         02400         call    senddebugbyte_1
02EB   3400         02401         retlw   0
                    02402
                    02403         space 1
                    02404 ;;latex\subsubsectionindex{delaynohang_0}
                    02405 ;;ps
                    02406 ; delaynohang_0 is used to eliminate hangups while waiting for an IO event
                    02407 ; to finish. It is expected that this routine will be entered with RAM
                    02408 ; bank 1 selected, RAM bank 1 will be selected on return. Note that no
                    02409 ; effort is made to make this fast, after all we want some kind of delay
                    02410 ; to be done here and the longer the better.
                    02411 ;
                    02412 ; Normally used to see if the transmiter part of the UART is sending a
                    02413 ; byte. We have to wait untill the preceeding byte is gone before sending
                    02414 ; a new one.
                    02415 ;
                    02416 ; Timing is as follows: about 5.79 us per call (16 * .361 us)
                    02417 ;
                    02418 ; Uses temp2_0 which should be set to zero before the first call.
                    02419 ; Is counted up to zero for the delay.
                    02420 ;;pe
                    02421 ;;ts
                    02422 ; On return c = 0 not timed out
                    02423 ;           c = 1 has timed out
                    02424 ;;te
02EC                02425 delaynohang_0                   ;                                       1
02EC   1283         02426         bcf     status,rp0      ; 1 RAM bank 0                          0
02ED   2AEE         02427         goto    $+1             ; 2 Cycle delay that                    0
02EE   2AEF         02428         goto    $+1             ; 2 . . does nothing slowly             0
02EF   2AF0         02429         goto    $+1             ; 2 . . . . . in less space than a nop  0
02F0   0FCF         02430         incfsz  temp2_0,f       ; 1 Decrement time out counter          0
02F1   2AF5         02431         goto    clearcandexit   ; 2 All done?                           0
                    02432
02F2   1403         02433         bsf     status,c        ; 1 Time out all done, set time out flag  0
02F3                02434 exitnohang                      ; - When exiting, must reselect the RAM bank  0
02F3   1683         02435         bsf     status,rp0      ; 1 RAM bank 1                          1
02F4   0008         02436         return                  ; 2 Return with RAM bank 1 selected     1
                    02437
02F5                02438 clearcandexit                   ; - Not timed out yet                   0
02F5   1003         02439         bcf     status,c        ; 1 Clear timeout flag                  0
02F6   2AF3         02440         goto    exitnohang      ; 2 Exit, with RAM bank selecting       0
                    02441

                    02442         space 1
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 58
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    02443 ;;latex\subsubsectionindex{delayspectra_0}
                    02444 ;;ps
                    02445 ; delayspectra_0 is used in receiving data from the Spectra.  It delays
                    02446 ; for about one half of a bit time, at 2400 baud, so that sampling may
                    02447 ; occur in the middle of each bit period.
                    02448 ;;pe
02F7                02449 delayspectra_0
02F7   30C0         02450         movlw   0xC0            ; Gives 208 us timeout
02F8   00B5         02451         movwf   overlong1_0     ; Load the down counter's lower half
02F9                02452 spectrawait
02F9   0BB5         02453         decfsz  overlong1_0,f   ; Decrement timer
02FA   2AF9         02454         goto    spectrawait     ; Not zero, check data line again
                    02455
02FB   3400         02456         retlw   0               ; Go on back to the caller
                    02457

                    02458         space 1
                    02459 ;;latex\subsubsectionindex{delayv_0}
                    02460 ;;ps
                    02461 ; delayv_0 is used to delay 250 Us * value in w.
                    02462 ; Each instruction step takes about .361898148 us to complete.
                    02463 ;;pe
02FC                02464 delayv_0
02FC   00CF         02465         movwf   temp2_0         ; Stick it in
02FD   30E6         02466         movlw   230             ; Minor loop counter, max value
02FE   00CE         02467 d2      movwf   temp1_0         ; Load the minor one too
02FF   0BCE         02468 d1      decfsz  temp1_0,f       ; With 230 loaded this step takes about 250 us
```

```
                      02469                             ; (Actually it's 249.5659722 us.)
0300   2AFF           02470         goto    d1          ; Keep on delaying
                      02471
0301   0BCF           02472         decfsz  temp2_0,f   ; 250 us * w = xx ms
0302   2AFE           02473         goto    d2          ; Back to decrementing
                      02474
0303   0008           02475         return              ; Done, return
                      02476
                      02477         space 1
                      02478 ;;latex\subsubsectionindex{doaux_3}
                      02479 ;;ps
                      02480 ; doaux_3 is used to send an AUX command to the Spectra.
                      02481 ;;pe
                      02482 ;;ts
                      02483 ; w = which AUX to configure
                      02484 ; d_command2_M = A SET or CLEAR AUX command
                      02485 ;;te
0304                  02486 doaux_3
0304   01A6           02487         clrf    d_command1_M    ; Zero out first command byte
0305   01A8           02488         clrf    d_pan_speed_M   ; Zero out first data byte
0306   00A9           02489         movwf   d_tilt_speed_M  ; Stick the AUX ID in
0307   2483           02490         call    tospectra_2     ; Send it on out
0308   0008           02491         return                  ; An we are done
                      02492
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 59
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
                      02493         space 1
                      02494 ;;latex\subsubsectionindex{dochecksum_0}
                      02495 ;;ps
                      02496 ; dochecksum_0 is used to sum up the contents of a transmit buffer and
                      02497 ; stick the results in the command.
                      02498 ;;pe
0309                  02499 dochecksum_0
0309   0825           02500         movf    d_address_M,w   ; Do checksum, don't include first byte
030A   0726           02501         addwf   d_command1_M,w  ; Add in the first user supplied byte
030B   0727           02502         addwf   d_command2_M,w  ; . and the second
030C   0728           02503         addwf   d_pan_speed_M,w ; . . the third
030D   0729           02504         addwf   d_tilt_speed_M,w; . . . and lastly the fourth
030E   00AA           02505         movwf   d_checksum_M    ; Stick the checksum in
030F   3400           02506         retlw   0               ; And return
                      02507
                      02508         space 1
                      02509 ;;latex\subsubsectionindex{eeread_0}
                      02510 ;;ps
                      02511 ; eeread_0 is used to read a byte from EEPROM memory.
                      02512 ;
                      02513 ; When called w has the address to read from.
                      02514 ;
                      02515 ; On return w has the addresses contents.
                      02516 ; (This routine has been copied from the Microchip data sheet for the
                      02517 ; PIC16F87x, DS30292C, page 43.)
                      02518 ;;pe
0310                  02519 eeread_0
0310   1703           02520         bsf     status,rp1      ; Ram bank 2                            2
0311   008D           02521         movwf   eeadr           ; Set up pointer                        2
0312   018F           02522         clrf    eeadrh          ; Zero out upper address too            2
0313   1683           02523         bsf     status,rp0      ; Ram bank 3                            3
0314   138C           02524         bcf     eecon1,eepgd    ; Point to data memory                  3
0315   140C           02525         bsf     eecon1,rd       ; Start read operation                  3
0316   1283           02526         bcf     status,rp0      ; Ram bank 2                            2
0317   080C           02527         movf    eedata,w        ; Get data                              2
0318   1303           02528         bcf     status,rp1      ; Ram bank 0                            0
0319   00CE           02529         movwf   temp1_0         ; Save recently read value              0
031A   0008           02530         return
                      02531
                      02532         space 1
                      02533 ;;latex\subsubsectionindex{eewrite_0}
                      02534 ;;ps
                      02535 ; eewrite_0 is used to write a byte into EEPROM memory.
                      02536 ;
                      02537 ; On entry w has the byte to be written.
                      02538 ; eewritehere_0, has the address in it to write into.
                      02539 ;
                      02540 ; (This routine has been copied from the Microchip data sheet for the
                      02541 ; PIC16F87x, DS30292C, page 43.)
```

```
                     02542 ;
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE 60
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     02543 ; eewrite_0 saves the currently written value in temp1_0 and does not
                     02544 ; change it.
                     02545 ;;pe
031B                 02546 eewrite_0
031B   00CE          02547         movwf   temp1_0         ; Save what gets written                    0
031C   082B          02548         movf    eewritehere_0,w ; Get where to write into                   0
031D   1703          02549         bsf     status,rp1      ; RAM bank 2                                2
031E   1683          02550         bsf     status,rp0      ; RAM bank 3                                3
031F                 02551 eeprombusy                      ;                                           3
031F   188C          02552         btfsc   eecon1,wr       ; Is the EEPROM busy?                       3
0320   2B1F          02553         goto    eeprombusy      ; Yep                                       3
                     02554                                 ;                                           3
0321   1283          02555         bcf     status,rp0      ; RAM bank 2                                2
0322   008D          02556         movwf   eeadr           ; Stick in pointer                          2
0323   018F          02557         clrf    eeadrh          ; Zero out upper address too                2
0324   1303          02558         bcf     status,rp1      ; RAM bank 0                                0
0325   084E          02559         movf    temp1_0,w       ; Get what gets written                     0
0326   1703          02560         bsf     status,rp1      ; RAM bank 2                                2
0327   008C          02561         movwf   eedata          ; Stick into holding register               2
0328   1683          02562         bsf     status,rp0      ; RAM bank 3                                3
0329   138C          02563         bcf     eecon1,eepgd    ; Point to data memory                      3
032A   150C          02564         bsf     eecon1,wren     ; Enable writing                            3
032B   3055          02565         movlw   0x55            ; First magic constant                      3
032C   008D          02566         movwf   eecon2          ; Send it out                               3
032D   30AA          02567         movlw   0xAA            ; Second magic constant                     3
032E   008D          02568         movwf   eecon2          ; Send it out too                           3
032F   148C          02569         bsf     eecon1,wr       ; Start write operation                     3
0330   110C          02570         bcf     eecon1,wren     ; Disable further writes                    3
0331   1283          02571         bcf     status,rp0      ; RAM bank 2                                2
0332   1303          02572         bcf     status,rp1      ; RAM bank 0                                0
0333   0008          02573         return
                     02574
                     02575         space 1
                     02576 ;;latex\subsubsectionindex{endout_1}
                     02577 ;;ps
                     02578 ; endout_1 is used to complete sending a message to the controller.
                     02579 ; (This routine is new as of 25JAN02.)
                     02580 ;;pe
0334                 02581 endout_1
0334   1D06          02582         btfss   NODELAY         ; Ending Delay? (Input bits are backward.)
0335   2B39          02583         goto    nodelay2        ; Nope
                     02584
0336   3005          02585         movlw   5               ; Gives a 1 ms delay
0337   22FC          02586         call    delayv_0        ; Just like Sensormatic does
0338   2B43          02587         goto    nomorewaiting   ; The 1 ms delay is always enough
                     02588
0339                 02589 nodelay2
                     02590         ; Besure to have the last bit go before turning off
                     02591         ; the communications line.
0339   01CF          02592         clrf    temp2_0         ; Initialize timer counter                  0
033A                 02593 waitforlastbit
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE 61
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

033A   1683          02594         bsf     status,rp0      ;  1 RAM bank 1                             1
033B   22EC          02595         call    delaynohang_0   ; 16 Delays 5.78 us each call               1
033C   1803          02596         btfsc   status,c        ;  1 Time out = 1, 0 = no timeout           1
033D   2B40          02597         goto    stopwaiting     ;  2 Took too long, quit this stuff         1
                     02598
033E   1C98          02599         btfss   txsta,trmt      ;  1 Is the transmitter busy?, bank 1       1
033F   2B3A          02600         goto    waitforlastbit  ;  2 Yep, keep timing out                   1
                     02601                                 ; 23 Total loop time is 8.31 us             1
                     02602
0340                 02603 stopwaiting                     ; 8.31 us * 256 = 2.13 ms, which is enough  1
0340   1283          02604         bcf     status,rp0      ; Back to the home bank                     0
0341   3001          02605         movlw   1               ; Now delay for 250 us
0342   22FC          02606         call    delayv_0        ; . to let the last bit out
                     02607                                 ; . . when txsta,trmt goes high, this means
                     02608                                 ; . . that the shift register is EMPTY, not
                     02609                                 ; . . that the last bit has been sent.
0343                 02610 nomorewaiting
```

```
0343   1287           02611         bcf     ENABLE_OUTPUTS  ; Turn off the output drivers
0344   0008           02612         return
                      02613
                      02614
                      02615         space 1
                      02616 ;;latex\subsubsectionindex{findspectraedge_0}
                      02617 ;;ps
                      02618 ; findspectraedge_0 is used to detect when a transition occurs in the
                      02619 ; data from the Spectra at 2400 baud.
                      02620 ;
                      02621 ; Returns with w set to the value of the transition.
                      02622 ;;pe
                      02623 ;;ts
                      02624 ;     0 = low going transition
                      02625 ;     1 = high going transition
                      02626 ;;te
                      02627 ;;ps
                      02628 ; Note that this is an unusual routine in that it has two different exit
                      02629 ; points.  For each bit, one or zero, there are different exits. Once
                      02630 ; checking for a transition, the check occurs about once every 1 us or
                      02631 ; so.
                      02632 ;;pe
0345                  02633 findspectraedge_0
0345   30FF           02634         movlw   0xFF            ; Timeout in case we come up incorrectly
0346   00B6           02635         movwf   overlong2_0     ; . . sending us any data, MSByte
0347   00B5           02636         movwf   overlong1_0     ; LSByte
0348   1A85           02637         btfsc   SPECTRA_IN      ; Is the raw data high or low?
0349   2B53           02638         goto    spectrahigh     ; High
                      02639
034A                  02640 spectralow
034A   1E85           02641         btfss   SPECTRA_IN      ; Check data line status
034B   2B4D           02642         goto    spectrawaitlo   ; Still low
034C   3401           02643         retlw   1               ; Had a transition
                      02644
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 62
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

034D                  02645 spectrawaitlo
034D   0BB5           02646         decfsz  overlong1_0,f
034E   2B4A           02647         goto    spectralow
                      02648
034F   00B5           02649         movwf   overlong1_0     ; Timeout is still in w
0350   0BB6           02650         decfsz  overlong2_0,f   ; Decrement outer loop
0351   2B4A           02651         goto    spectralow
0352   2CFA           02652         goto    resendquery
                      02653
0353                  02654 spectrahigh
0353   1A85           02655         btfsc   SPECTRA_IN      ; Check for still being high
0354   2B56           02656         goto    spectrawaithi
0355   3400           02657         retlw   0               ; Return
                      02658
0356                  02659 spectrawaithi
0356   0BB5           02660         decfsz  overlong1_0,f
0357   2B53           02661         goto    spectrahigh
                      02662
0358   00B5           02663         movwf   overlong1_0     ; Timeout is still in w
0359   0BB6           02664         decfsz  overlong2_0,f   ; Decrement outer loop
035A   2B53           02665         goto    spectrahigh
                      02666
035B   2CFA           02667         goto    resendquery
                      02668

                      02669         space 1
                      02670 ;;latex\subsubsectionindex{flipinput_0}
                      02671 ;;ps
                      02672 ; flipinput_0 is used to alternate the input flipping logic.
                      02673 ;;pe
035C                  02674 flipinput_0
035C   0AB3           02675         incf    mode1_M,f       ; Flip the least significant bit
035D   10B3           02676         bcf     RESERVED_BIT    ; Don't let it get too big, a single bit is OK
035E   1C33           02677         btfss   NORMAL_POLARITY ; Test it twice, first for clearing
035F   1105           02678         bcf     INVERT_IO       ; Oh it gets cleared
0360   1833           02679         btfsc   NORMAL_POLARITY ; Now test it for setting
0361   1505           02680         bsf     INVERT_IO       ; Well, well it needs to be set
0362   0008           02681         return
                      02682

                      02683         space 1
```

```
                     02684 ;;latex\subsubsectionindex{fromspectra_1}
                     02685 ;;ps
                     02686 ; fromspectra_1 is used to read three bytes from the Spectra. Since there
                     02687 ; is no UART available to do this, we have to bit-bang to get the message
                     02688 ; in. Normally this is used to get just the Spectra's address which
                     02689 ; requires reading only the second byte. It has been modified to now read
                     02690 ; in three bytes so that the alarm status may be read. In all cases the
                     02691 ; checksum is ignored.
                     02692 ;
                     02693 ; Now bit-bang to get the answer in.
                     02694 ; Received data format is:
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 63
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     02695 ;;pe
                     02696 ;;ts
                     02697 ;       Non return to zero (NRZ)
                     02698 ;       Least significant bit comes in first
                     02699 ;       High is a one
                     02700 ;       Low is a zero
                     02701 ;       Quiescent is high
                     02702 ;       RS-422 voltage levels
                     02703 ;       One start bit which goes low
                     02704 ;       Eight data bits
                     02705 ;       One stop bit which goes high or is already high depending on the data
                     02706 ;       No parity
                     02707 ;
                     02708 ;       First byte is a sync byte of all ones.
                     02709 ;       Second byte is the Spectra's address.
                     02710 ;       Most of the rest is the software PGM number.
                     02711 ;       Last byte is the checksum of everything else.
                     02712 ;;te
0363                 02713 fromspectra_1
0363   2345          02714         call    findspectraedge_0; Get beginning of start bit
0364   1A85          02715         btfsc   SPECTRA_IN      ; Start bits are low
0365   2CFA          02716         goto    resendquery     ; Start all over again
                     02717
                     02718         ; Check the start bit twice to reduce noise problems
0366   22F7          02719         call    delayspectra_0  ; Wait for the middle of the bit
0367   1A85          02720         btfsc   SPECTRA_IN      ; Still low? Was that transition noise?
0368   2CFA          02721         goto    resendquery     ; Nope, try again
                     02722
                     02723         ; Start getting the first byte in
0369   2345          02724         call    findspectraedge_0; Get first of eight one bits in sync byte
036A   22F7          02725         call    delayspectra_0  ; Get to the middle of the bit time
036B   1E85          02726         btfss   SPECTRA_IN      ; Should still be high
036C   2CFA          02727         goto    resendquery     ; Woops, failed a noise check. Try again.
                     02728
036D   3008          02729         movlw   8               ; Should be 7 more high sync data bits now
036E   00B6          02730         movwf   overlong2_0     ; Set bit counter
                     02731                                 ; . . delayspectra_0 uses overlong1_0
036F                 02732 nextsync
036F   22F7          02733         call    delayspectra_0  ; One for the end
0370   22F7          02734         call    delayspectra_0  ; And one more for the middle
0371   1E85          02735         btfss   SPECTRA_IN      ; All sync bits are high
0372   2CFA          02736         goto    resendquery     ; Try again, twas low
                     02737
0373   0BB6          02738         decfsz  overlong2_0,f   ; Done um all?
0374   2B6F          02739         goto    nextsync        ; Get over the sync byte
                     02740
                     02741         ; Can't keep calling findspectraedge_0 because this is NRZ data
                     02742         ; so re-syncing to the byte start here is very important
0375   2345          02743         call    findspectraedge_0; Get the start of the next start bit
0376   22F7          02744         call    delayspectra_0  ; Get to the middle
0377   1A85          02745         btfsc   SPECTRA_IN      ; All start bits are low
0378   2CFA          02746         goto    resendquery     ; Try some more
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 64
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     02747
0379   01BF          02748         clrf    spectraaddress_M; Get ready to read in the address
037A   3008          02749         movlw   8               ; Should be 8 address data bits now
037B   00B6          02750         movwf   overlong2_0     ; Set bit counter
                     02751
037C                 02752 nextaddressbit
037C   22F7          02753         call    delayspectra_0
```

```
037D   00D4           02754        movwf   thisbit_0       ; Save
037E   1403           02755        bsf     status,c        ; Set carry bit
037F   1E85           02756        btfss   SPECTRA_IN      ; Is the LSB set?
0380   1003           02757        bcf     status,c        ; Nope
0381   0CBF           02758        rrf     spectraaddress_M,f; Rotate carry into byte, cuz it comes in
                      02759                                ; LSB first
0382   22F7           02760        call    delayspectra_0
0383   0BB6           02761        decfsz  overlong2_0,f   ; Done um all yet
0384   2B7C           02762        goto    nextaddressbit
                      02763
                      02764 ; put logic to read in one more byte here for alarm information.
0385   0008           02765        return                  ; And back we go
                      02766
                      02767        space 1
                      02768 ;;latex\subsubsectionindex{getsensorbyte_1}
                      02769 ;;ps
                      02770 ; getsensorbyte_1 is used to get in a byte of keyboard data.
                      02771 ;
                      02772 ; This is where the program normally sits while waiting for input from
                      02773 ; the head end. When an input comes in, then it is processed and we
                      02774 ; return here.
                      02775 ;;pe
0386                  02776 getsensorbyte_1
0386   1E8C           02777 wait    btfss   pir1,rcif       ; Any thing in here yet?
0387   2B86           02778        goto    wait            ; Not yet
                      02779
0388   22A7           02780        call    ckioerrs_0      ; Get byte and errors
                      02781        ifdef   ALLOW_INPUT_FLIP
                      02782         btfss   INPUT_ERROR     ; Any errors?
                      02783        endif ; ifdef ALLOW_INPUT_FLIP
0389   0008           02784 goback  return                  ; Nope, return and don't hurt w or c
                      02785
038A   0BA0           02786 flipck  decfsz  badcount_M,f    ; Count bads
038B   2B86           02787        goto    wait            ; Keep on counting
                      02788
038C   235C           02789        call    flipinput_0     ; Change the input levels with an XOR shift
038D   2B86           02790        goto    wait            ; Back to looking for data
                      02791
                      02792        space 1
                      02793 ;;latex\subsubsectionindex{inlabel_3}
                      02794 ;;ps
                      02795 ; inlabel_3 is used to help in overwriting the Spectra's "CONFIGURATION
                      02796 ; DONE" message with our message on line 2.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 65
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      02797 ;pe
                      02798 ;ts
                      02799 ;             "TXB-S422 Rev X.XX AA"     <message>
                      02800 ;              123456789 1234567890      <character position>
                      02801 ;te
                      02802 ;ps
                      02803 ; Call with w holding the character to stick into the message.
                      02804 ; Returns with the message updated for location, a new
                      02805 ; character stuck in the Spectra buffer and the buffer sent to the
                      02806 ; Spectra.
                      02807 ;;pe
038E                  02808 inlabel_3
038E   0AA8           02809        incf    d_pan_speed_M,f ; Get to the next sticking place
038F   00A9           02810        movwf   d_tilt_speed_M  ; Stick new byte in
0390   2483           02811        call    tospectra_2     ; Checksum and send the message
0391   0008           02812        return
                      02813
                      02814        space 1
                      02815 ;;latex\subsubsectionindex{presetincrement_1}
                      02816 ;;ps
                      02817 ; presetincrement_1 is used to get the next preset value from EEPROM and
                      02818 ; to update the highest preset number saved in EEPROM.
                      02819 ;
                      02820 ; On return w has the new preset value.
                      02821 ;
                      02822 ; Range of stored presets is 0 --> 63, but range of Spectra presets is
                      02823 ; 1 --> 64. So on getting the preset it is always incremented by 1.
                      02824 ;
                      02825 ; Note that it takes 4 --> 8 ms to complete an EEPROM write.
                      02826 ;;pe
```

```
0392                    02827 presetincrement_1
0392   3022             02828         movlw   EE_PRESET       ; Get address to write into
0393   00AB             02829         movwf   eewritehere_0    ; Save write address too
0394   2310             02830         call    eeread_0         ; Read a byte
0395   1B33             02831         btfsc   NO_MORE_PRESETS ; May we increment this preset request
0396   2B9A             02832         goto    nopresetsnow     ; Nope
                        02833
0397   1733             02834         bsf     NO_MORE_PRESETS ; Ignore presets for a bit
0398   0A4E             02835         incf    temp1_0,w        ; Get value to write
0399   231B             02836         call    eewrite_0        ; Write it
039A                    02837 nopresetsnow
039A   084E             02838         movf    temp1_0,w        ; Get it again for range fixing
039B   393F             02839         andlw   0x3F             ; Dump slop (for just plain too big values)
039C   00CE             02840         movwf   temp1_0          ; Save new preset value
039D   3C20             02841         sublw   32               ; Is preset past the two reserved presets?
039E   1803             02842         btfsc   status,c         ; Check
039F   2BA2             02843         goto    notmorethan33    ; Nope
                        02844
03A0   0ACE             02845         incf    temp1_0,f        ; Yep, increment it
03A1   0ACE             02846         incf    temp1_0,f        ; . . past the gap
03A2                    02847 notmorethan33
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 66
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

03A2   084E             02848         movf    temp1_0,w        ; Get preset
03A3   0008             02849         return                   ; Go on back home
                        02850
                        02851         space 1
                        02852 ;;latex\subsubsectionindex{printsensorin_4}
                        02853 ;;ps
                        02854 ; printsensorin_4 is used to printout the received command, in debug mode.
                        02855 ; (This routine was changed on 29JAN02.)
                        02856 ;;pe
03A4                    02857 printsensorin_4
03A4   1A86             02858         btfsc   DEBUG_MODE_ON    ; Is ASCII terminal debugging enabled?
03A5   2BD9             02859         goto    dontprintin      ; No debugging is being done, exit
                        02860
03A6   22E7             02861         call    crlf_2           ; Print an end of line
03A7   0AB1             02862         incf    messcount_4,f    ; Increment the byte counter
03A8   0831             02863         movf    messcount_4,w    ; Get it
03A9   390F             02864         andlw   0x0F             ; Dump slop
03AA   1903             02865         btfsc   status,z         ; Stick out a blank line every 16 lines
03AB   22E7             02866         call    crlf_2           ; Print a blank line
03AC   0831             02867         movf    messcount_4,w    ; Get count to show
03AD   23F5             02868         call    send2hex_3       ; And display it
03AE   224D             02869         call    blank_2          ; Stick out a cheap delimiter
03AF   224D             02870         call    blank_2          ; . . . or two
03B0   0841             02871         movf    sproto1_M,w      ; Address, show what we just got in
03B1   23F5             02872         call    send2hex_3       ; S commands are usually 3 bytes long
03B2   0842             02873         movf    sproto2_M,w      ; Command
03B3   23F5             02874         call    send2hex_3
03B4   0843             02875         movf    sproto3_M,w      ; Checksum
03B5   23F5             02876         call    send2hex_3
03B6   0840             02877         movf    sprotolength_M,w ; How long is the message
03B7   1903             02878         btfsc   status,z         ; Check
03B8   2BD9             02879         goto    dontprintin
                        02880
                        02881         ; Now we have to deal with long messages
03B9   0844             02882         movf    sproto4_M,w
03BA   23F5             02883         call    send2hex_3
03BB   0842             02884         movf    sproto2_M,w      ; Get current command op-code
03BC   3CC3             02885         sublw   S_PROP_SPEED     ; Is this 0xC3? (i.e. 4 bytes long)
03BD   1903             02886         btfsc   status,z         ; Check
03BE   2BD9             02887         goto    dontprintin
                        02888
03BF   0845             02889         movf    sproto5_M,w
03C0   23F5             02890         call    send2hex_3
03C1   0842             02891         movf    sproto2_M,w      ; Get current command op-code
03C2   3CC0             02892         sublw   S_VARIABLE_SPEED; Is this 0xC0? (i.e. 5 bytes long)
03C3   1903             02893         btfsc   status,z         ; Check
03C4   2BD9             02894         goto    dontprintin
                        02895
03C5   0846             02896         movf    sproto6_M,w
03C6   23F5             02897         call    send2hex_3
03C7   0847             02898         movf    sproto7_M,w
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 67
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

03C8   23F5           02899         call    send2hex_3
03C9   0848           02900         movf    sproto8_M,w
03CA   23F5           02901         call    send2hex_3
03CB   0849           02902         movf    sproto9_M,w
03CC   23F5           02903         call    send2hex_3
03CD   084A           02904         movf    sproto10_M,w
03CE   23F5           02905         call    send2hex_3
03CF   084B           02906         movf    sproto11_M,w
03D0   23F5           02907         call    send2hex_3
03D1   084C           02908         movf    sproto12_M,w
03D2   23F5           02909         call    send2hex_3
03D3   0842           02910         movf    sproto2_M,w     ; Get current command op-code
03D4   3CC1           02911         sublw   S_UNKNOWN_C1    ; Is this 0xC1? (i.e. 12 bytes long)
03D5   1903           02912         btfsc   status,z        ; Check
03D6   2BD9           02913         goto    dontprintin
                      02914
03D7   084D           02915         movf    sproto13_M,w
03D8   23F5           02916         call    send2hex_3
03D9                  02917 dontprintin
03D9   3400           02918         retlw   0
                      02919

                      02920         space 1
                      02921 ;;latex\subsubsectionindex{printspectraout_4}
                      02922 ;;ps
                      02923 ; printspectraout_4 is used to printout the D protocol command, in debug
                      02924 ; mode. (This routine was changed on 29JAN02.)
                      02925 ;;pe
03DA                  02926 printspectraout_4
03DA   1A86           02927         btfsc   DEBUG_MODE_ON   ; Is ASCII terminal debugging enabled?
03DB   2BF4           02928         goto    dontprint       ; No debugging is being done, exit
                      02929
03DC   22E7           02930         call    crlf_2          ; Print an end of line
03DD   0AB1           02931         incf    messcount_4,f   ; Increment the byte counter
03DE   0831           02932         movf    messcount_4,w   ; Get it
03DF   390F           02933         andlw   0x0F            ; Dump slop
03E0   1903           02934         btfsc   status,z        ; Stick out a blank line every 16 lines
03E1   22E7           02935         call    crlf_2          ; Print a blank line
03E2   224D           02936         call    blank_2         ; Offset everything going to the Spectra
03E3   0831           02937         movf    messcount_4,w   ; Get count to show
03E4   23F5           02938         call    send2hex_3      ; And display it
03E5   224D           02939         call    blank_2         ; Stick out a creep delimiter
03E6   0824           02940         movf    d_sync_M,w      ; Now show what we are about to send out
03E7   23F5           02941         call    send2hex_3      ; D commands are 7 bytes long
03E8   0825           02942         movf    d_address_M,w   ; Address
03E9   23F5           02943         call    send2hex_3
03EA   0826           02944         movf    d_command1_M,w  ; Command 1
03EB   23F5           02945         call    send2hex_3
03EC   0827           02946         movf    d_command2_M,w  ; Command 2
03ED   23F5           02947         call    send2hex_3
03EE   0828           02948         movf    d_pan_speed_M,w ; Data 1
03EF   23F5           02949         call    send2hex_3
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 68
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

03F0   0829           02950         movf    d_tilt_speed_M,w; Data 2
03F1   23F5           02951         call    send2hex_3
03F2   082A           02952         movf    d_checksum_M,w  ; Checksum
03F3   23F5           02953         call    send2hex_3
03F4                  02954 dontprint
03F4   3400           02955         retlw   0
                      02956

                      02957         space 1
                      02958 ;;latex\subsubsectionindex{send2hex_3}
                      02959 ;;ps
                      02960 ; send2hex_3 is used to print out a pair of converted hex digits.
                      02961 ; Enter with the value to printed out in w.
                      02962 ;;pe
03F5                  02963 send2hex_3
03F5   2230           02964         call    bin2hex_0       ; Convert
03F6   082E           02965         movf    hexupper_0,w    ; Get upper
03F7   2406           02966         call    senddebugbyte_1 ; . . and print
03F8   082D           02967         movf    hexlower_0,w    ; Get lower
03F9   2406           02968         call    senddebugbyte_1 ; . . print it too
03FA   224D           02969         call    blank_2         ; Stick out a spacer
```

```
03FB   3400           02970         retlw   0               ; Return
                      02971
                      02972         space 1
                      02973 ;;latex\subsubsectionindex{sendack_5}
                      02974 ;;ps
                      02975 ; sendack_5 is used to print out an arrow to mark what are ACKs.
                      02976 ; (This routine was last updated on 25JAN02.)
                      02977 ;;pe
03FC                  02978 sendack_5
03FC   2225           02979         call    arrowout_3      ; Clearly indicate that this is an ACK
03FD   083F           02980         movf    spectraaddress_M,w; Get our address
03FE   246C           02981         call    tosensormatic_4 ; Send the ACK
03FF   2334           02982         call    endout_1        ; End sending it
0400   0008           02983         return
                      02984
                      02985         space 1
                      02986 ;;latex\subsubsectionindex{sendchecksum_6}
                      02987 ;;ps
                      02988 ; sendchecksum_6 is used to complete the checksum, send it, wait a short
                      02989 ; time and disable the transmitter.
                      02990 ; (This routine was last updated on 25JAN02.)
                      02991 ;;pe
0401                  02992 sendchecksum_6
0401   09A2           02993         comf    checksum_M,f    ; Now complement it
                      02994                                 ; Actual checksum is all bytes added together
                      02995                                 ; and subtracted from 0x100. But this gives
                      02996                                 ; the same answer.
0402   0A22           02997         incf    checksum_M,w    ; Not 1's complement make it 2's complement
0403   2425           02998         call    sendone_5       ; Last byte
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 69
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
0404   2334           02999         call    endout_1        ; Finsh up sending a message
0405   0008           03000         return
                      03001
                      03002         space 1
                      03003 ;;latex\subsubsectionindex{senddebugbyte_1}
                      03004 ;;ps
                      03005 ; senddebugbyte_1 is used to transmit one byte of information from the w
                      03006 ; register to the serial output port a single bit at a time.  The assumed
                      03007 ; byte characteristics are: DEBUG_PERIOD baud, one start bit, two stop bits,
                      03008 ; eight data bits and no parity. (RS-232 is negative true logic with the
                      03009 ; least significant bit being shifted out first.) It is important to note
                      03010 ; that this "RS-232" does not go thru a level shifter and thus is exactly
                      03011 ; what the receiving unit receives. I know that RS-232 is not specified
                      03012 ; as having voltage levels of 0 and +5, but it does work for short
                      03013 ; distances and the only time that this is used, is in debugging. Thus
                      03014 ; it's OK to do it. (This routine was changed on 29JAN02.)
                      03015 ;;pe
0406                  03016 senddebugbyte_1
0406   1A86           03017         btfsc   DEBUG_MODE_ON   ; Is ASCII terminal debugging enabled?
0407   2C1E           03018         goto    nodebugprint    ; No debugging is being done, exit
                      03019
0408   00BE           03020         movwf   sentbyte_1      ; Save it for working with
0409   3008           03021         movlw   8               ; Send all 8 bits in the byte
040A   00A1           03022         movwf   bytebits_1      ; Set counter
040B   1706           03023         bsf     DATA_OUT        ; Start the start bit going
040C   0000           03024         nop                     ; 1 Used to compensate for the extra
040D   0000           03025         nop                     ; 2 . stuff done to actually send
040E   0000           03026         nop                     ; 3 . . the byte's bits out
040F   0000           03027         nop                     ; 4 . . . all four are needed
0410                  03028 nextdebugbitout                 ; BitBang each bit out
0410   3004           03029         movlw   DEBUG_PERIOD    ; Get timeout counter preset
0411   2249           03030         call    bitdelay_0      ; Wait for a bit period
0412   183E           03031         btfsc   sentbyte_1,bit0 ; Is this bit a one or a zero
0413   1306           03032         bcf     DATA_OUT        ; Make output line active
0414   1C3E           03033         btfss   sentbyte_1,bit0 ; Logic for bit time compensation
0415   1706           03034         bsf     DATA_OUT        ; Its a zero, make line inactive
0416   0CBE           03035         rrf     sentbyte_1,f    ; Move it over for next time
0417   0BA1           03036         decfsz  bytebits_1,f    ; Done um all yet?
0418   2C10           03037         goto    nextdebugbitout ; Nope do more
0419   1306           03038         bcf     DATA_OUT        ; Send stop bit
                      03039                                 ; Since the last data bit in ASCII is always
                      03040                                 ; the same as a stop bit, we can get away with
                      03041                                 ; this.
041A   3004           03042         movlw   DEBUG_PERIOD    ; Get timeout counter preset
```

```
041B   2249           03043         call    bitdelay_0      ; Let last bit plus a stop bit out
041C   3004           03044         movlw   DEBUG_PERIOD    ; Get timeout counter preset
041D   2249           03045         call    bitdelay_0      ; Let last bit plus a stop bit out
041E                  03046 nodebugprint
041E   3400           03047         retlw   0               ; Byte all gone, go home
                      03048
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 70
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
                      03049         space 1
                      03050 ;;latex\subsubsectionindex{sendmany0_5}
                      03051 ;;ps
                      03052 ; sendmany0_5 is used to send several bytes of 0x00 to the head end. How
                      03053 ; many is in w on entry.
                      03054 ;;pe
041F                  03055 sendmany0_5
041F   00B0           03056         movwf   manycount_5     ; Save the send count
0420                  03057 sendonemore
0420   3000           03058         movlw   0x00            ; Get nothing
0421   246C           03059         call    tosensormatic_4 ; Send the byte
0422   0BB0           03060         decfsz  manycount_5,f   ; Have we sent them all?
0423   2C20           03061         goto    sendonemore     ; Nope
                      03062
0424   0008           03063         return                  ; Back home now
                      03064

                      03065         space 1
                      03066 ;;latex\subsubsectionindex{sendone_5}
                      03067 ;;ps
                      03068 ; sendone_5 is used to send the byte in w to the head end and to increment
                      03069 ; the checksum.
                      03070 ;;pe
0425                  03071 sendone_5
0425   07A2           03072         addwf   checksum_M,f    ; Increment the checksum
0426   246C           03073         call    tosensormatic_4 ; Send the message
0427   0008           03074         return                  ; Back to the caller
                      03075

                      03076         space 1
                      03077 ;;latex\subsubsectionindex{sendspectrabyte_0}
                      03078 ;;ps
                      03079 ; sendspectrabyte_0 is used to transmit one byte of information from the w
                      03080 ; register to the serial output port a single bit at a time.  The assumed
                      03081 ; byte characteristics are: 2400 baud, one start bit, two stop bits,
                      03082 ; eight data bits and no parity. (RS-422 is positive true logic with the
                      03083 ; least significant bit being shifted out first.)
                      03084 ;;pe
0428                  03085 sendspectrabyte_0
0428   00BE           03086         movwf   sentbyte_1      ; Save it for working with
0429   3008           03087         movlw   8               ; Send all 8 bits in the byte
042A   00A1           03088         movwf   bytebits_1      ; Set counter
042B   1185           03089         bcf     SPECTRA_OUT     ; Start the start bit going
042C   0000           03090         nop                     ; 1 Used to compensate for the extra
042D   0000           03091         nop                     ; 2 . stuff done to actually send
042E   0000           03092         nop                     ; 3 . . the byte's bits out
042F   0000           03093         nop                     ; 4 . . . all four are needed
0430                  03094 spectranextout                  ; BitBang each bit out
0430   3008           03095         movlw   DBAUD_REPEAT    ; At 2400 baud the delay is greater
0431   00AC           03096         movwf   extratimeout_0  ; . . than an 8 bit register can hold
0432                  03097 reload2400                      ; . . . so we need two counters, etc.
0432   302F           03098         movlw   DBAUD_BASIC     ; Get timeout counter preset
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 71
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0433   00D6           03099         movwf   timeout_0       ; Init counter for delay
0434                  03100 spectrawaitmore1
0434   0BD6           03101         decfsz  timeout_0,f     ; Done delaying?
0435   2C34           03102         goto    spectrawaitmore1; Nope
                      03103
0436   0BAC           03104         decfsz  extratimeout_0,f
0437   2C32           03105         goto    reload2400
                      03106
0438   183E           03107         btfsc   sentbyte_1,bit0 ; Is this bit a one or a zero
0439   1585           03108         bsf     SPECTRA_OUT     ; Its a zero, make line inactive
043A   1C3E           03109         btfss   sentbyte_1,bit0 ; Logic for bit time compensation
```

```
043B   1185           03110         bcf     SPECTRA_OUT     ; Make output line active
043C   0CBE           03111         rrf     sentbyte_1,f    ; Move it over for next time
043D   0BA1           03112         decfsz  bytebits_1,f    ; Done um all yet?
043E   2C30           03113         goto    spectranextout  ; Nope do more
                      03114
                      03115         ; Now let that last bit be on the line long enough to be acted on
043F   3008           03116         movlw   DBAUD_REPEAT    ; At 2400 baud the delay is greater
0440   00AC           03117         movwf   extratimeout_0  ; . . than an 8 bit register can hold
0441                  03118 reload2400last                  ; . . . so we need two counters, etc.
0441   302F           03119         movlw   DBAUD_BASIC     ; Get timeout counter preset
0442   00D6           03120         movwf   timeout_0       ; Init counter for delay
0443                  03121 spectrawaitlast                 ; Now wait for the last bit to go
0443   0BD6           03122         decfsz  timeout_0,f     ; Done delaying?
0444   2C43           03123         goto    spectrawaitlast ; Nope
                      03124
0445   0BAC           03125         decfsz  extratimeout_0,f; Done waiting
0446   2C41           03126         goto    reload2400last  ; Nope, do some more
                      03127
0447   1585           03128         bsf     SPECTRA_OUT     ; Yep, start sending at least 1 stop bits
0448   3008           03129         movlw   DBAUD_REPEAT
0449   00AC           03130         movwf   extratimeout_0
044A                  03131 spectrawaitstop
044A   302F           03132         movlw   DBAUD_BASIC     ; Get timeout counter preset
044B   00D6           03133         movwf   timeout_0       ; Init counter for delay
044C                  03134 spectrawaitmore2
044C   0BD6           03135         decfsz  timeout_0,f     ; Done delaying?
044D   2C4C           03136         goto    spectrawaitmore2; Nope
                      03137
044E   0BAC           03138         decfsz  extratimeout_0,f
044F   2C4A           03139         goto    spectrawaitstop
                      03140
0450   0008           03141         return                  ; Byte all gone, go home
                      03142


                      03143         space 1
                      03144 ;;latex\subsubsection{speed_save_0}
                      03145 ;;ps
                      03146 ; speed_save_0 is used to take the current pan and tilt speeds and move
                      03147 ; them to the two "save" areas.
                      03148 ;;pe
                      03149 ;;ts
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 72
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE


                      03150 ; Copies from this_pan_speed_M  into saved_pan_speed_M
                      03151 ; and         this_tilt_speed_M into saved_tilt_speed_M
                      03152 ;;te
0451                  03153 speed_save_0
0451   0852           03154         movf    this_pan_speed_M,w    ; Save the original values
0452   00BC           03155         movwf   saved_pan_speed_M
0453   0853           03156         movf    this_tilt_speed_M,w
0454   00BD           03157         movwf   saved_tilt_speed_M
0455   0008           03158         return
                      03159


                      03160         space 1
                      03161 ;;latex\subsubsectionindex{startout_1}
                      03162 ;;ps
                      03163 ; startout_1 is used to enable the controller output driver and then
                      03164 ;            delay 250 us extra.  There is always about 50 to 150 us of
                      03165 ;            overhead that is always there. (And no I don't understand
                      03166 ;            why it varies!)
                      03167 ; (This routine was updated on 25JAN02.)
                      03168 ;;pe
0456                  03169 startout_1
0456   1687           03170         bsf     ENABLE_OUTPUTS  ; Enable the output drivers
0457   1D06           03171         btfss   NODELAY         ; Do we put out a before/after delay
0458   2C5B           03172         goto    nodelay1        ; Nope, act like an UltraDome
                      03173
0459   3001           03174         movlw   1               ; Gives a 250 us delay
045A   22FC           03175         call    delayv_0        ; Wait a bit for lines to stabalize
045B                  03176 nodelay1                        ; No starting delay
045B   0008           03177         return
                      03178


                      03179         space 1
                      03180 ;;latex\subsubsectionindex{threebytemessage_6}
                      03181 ;;ps
                      03182 ; threebytemessage_6 is used to send a three byte message to the
```

```
                      03183 ; controller. The address and checksum fields are calculated or supplied
                      03184 ; by this routine.
                      03185 ; (This routine was last updated on 25JAN02.)
                      03186 ;
                      03187 ; On entry w has the value to send
                      03188 ;;pe
045C                  03189 threebytemessage_6
045C   00C2           03190         movwf   sproto2_M       ; Stick value into the message
045D   083F           03191         movf    spectraaddress_M,w ; Get my address
045E   00C1           03192         movwf   sproto1_M       ; Stick it into the message
045F   0742           03193         addwf   sproto2_M,w     ; Calculate the checksum
0460   00C3           03194         movwf   sproto3_M       ; Preliminaryly stick it into the message
0461   09C3           03195         comf    sproto3_M,f     ; Now complement it
                      03196                                 ; Actual checksum is sproto1_M plus sproto2_M
                      03197                                 ; subtracted from 0x100. But this gives the
                      03198                                 ; same answer.
0462   0AC3           03199         incf    sproto3_M,f     ; Not 1's complement make it 2's complement
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 73
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0463   2225           03200         call    arrowout_3      ; Flag the debug output
0464   0841           03201         movf    sproto1_M,w     ; Send address
0465   246C           03202         call    tosensormatic_4
0466   0842           03203         movf    sproto2_M,w     ; Send reply data
0467   246C           03204         call    tosensormatic_4
0468   0843           03205         movf    sproto3_M,w     ; Send checksum
0469   246C           03206         call    tosensormatic_4
046A   2334           03207         call    endout_1        ; Finish up sending a message
046B   0008           03208         return                  ; Go on back
                      03209

                      03210         space 1
                      03211 ;;latex\subsubsectionindex{tosensormatic_4}
                      03212 ;;ps
                      03213 ; tosensormatic_4 is used to send bytes to the Sensormatic controller.
                      03214 ; (This routine was changed on 29JAN02.)
                      03215 ;;pe
                      03216 ;;ts
                      03217 ; w = what to send
                      03218 ;;te
046C                  03219 tosensormatic_4
046C   00CE           03220         movwf   temp1_0         ; Save for debug display
046D   01CF           03221         clrf    temp2_0         ; Initialize timer counter
046E                  03222 selectbank1
046E   1683           03223         bsf     status,rp0      ;  1 RAM bank 1                                 1
046F   22EC           03224         call    delaynohang_0   ; 16 Delays 5.78 us each call                   1
0470   1803           03225         btfsc   status,c        ;  1 Time out = 1, 0 = no timeout               1
0471   2C74           03226         goto    giveupthistry   ;  2 Took too long, quit this stuff             1
                      03227
0472   1C98           03228         btfss   txsta,trmt      ;  1 Is the transmitter busy?, bank 1           1
0473   2C6E           03229         goto    selectbank1     ;  2 Yep, keep timing out                       1
                      03230                                 ; 23 Total loop time is 8.31 us                 1
                      03231
0474                  03232 giveupthistry                   ; 8.31 us * 256 = 2.13 ms, which is enough      1
0474   1283           03233         bcf     status,rp0      ; RAM bank 0                                    0
0475   0099           03234         movwf   txreg           ; Load the output buffer, bank 0                0
0476   1687           03235         bsf     ENABLE_OUTPUTS  ; Enable response drivers, bank 0               0
0477   084E           03236         movf    temp1_0,w       ; Get that byte                                 0
0478   1E86           03237         btfss   DEBUG_MODE_ON   ; Do we echo this character?                    0
0479   23F5           03238         call    send2hex_3      ; Display it                                    0
047A   01CF           03239         clrf    temp2_0         ; Initialize timer counter                      0
047B                  03240 selectbank1now
047B   1683           03241         bsf     status,rp0      ;  1 RAM bank 1                                 1
047C   22EC           03242         call    delaynohang_0   ; 16 Delays 5.78 us each call                   1
047D   1803           03243         btfsc   status,c        ;  1 Time out = 1, 0 = no timeout               1
047E   2C81           03244         goto    giveupthistrynow;  2 Took too long, quit this stuff             1
                      03245
047F   1C98           03246         btfss   txsta,trmt      ;  1 Is the transmitter busy?, bank 1           1
0480   2C7B           03247         goto    selectbank1now  ;  2 Yep, keep timing out                       1
                      03248                                 ; 23 Total loop time is 8.31 us                 1
                      03249
0481                  03250 giveupthistrynow                ; 8.31 us * 256 = 2.13 ms, which is enough      1
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 74
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0481   1283           03251         bcf     status,rp0      ; Back to the home bank                         0
```

```
0482   0008           03252         return                  ; Return, with or with out sending it          0
                      03253
                      03254         space 1
                      03255 ;;latex\subsubsectionindex{tospectra_2}
                      03256 ;;ps
                      03257 ; tospectra_2 is used to send a D protocol command to a Spectra.
                      03258 ;
                      03259 ; A special problem is fixed with this routine. That problem is that it
                      03260 ; takes so long to send a Spectra command in D protocol at 2400 baud
                      03261 ; (29.6 ms) that we miss commands from the Sensormatic controller.
                      03262 ; (Remember that there is only one UART on this baby and that we have to
                      03263 ; bit-bang data out to the Spectra and can't do anything else or the
                      03264 ; timing gets messed up.) The solution to this is to check the receive
                      03265 ; register after each transmitted byte and to store the data if any data
                      03266 ; comes in. If BYTE3 is set then all three bytes have been received and
                      03267 ; we can exit. Conveniently enough the UART logic on the PIC chip is
                      03268 ; somewhat more than double buffered, i.e. there are two holding
                      03269 ; registers (in a FIFO) arrangement in addition to the input shift
                      03270 ; register. Since the data comes in at 4800 baud and we send data out to
                      03271 ; the Spectra at 2400 baud, if we check for received data after each sent
                      03272 ; byte, we should never "loose" a three byte message.
                      03273 ;
                      03274 ; Automatically supplied fields are:
                      03275 ;;pe
                      03276 ;;ts
                      03277 ;       d_sync_M        0xFF, sync byte
                      03278 ;       d_address_M     spectraaddress_M
                      03279 ;       d_checksum_M    checksum
                      03280 ;;te
                      03281 ;;ps
                      03282 ; Fields to be provided by the caller:
                      03283 ;;pe
                      03284 ;;ts
                      03285 ;       d_command1_M    Command 1
                      03286 ;       d_command2_M    Command 2
                      03287 ;       d_pan_speed_M   Data 1
                      03288 ;       d_tilt_speed_M  Data 2
                      03289 ;;te
0483                  03290 tospectra_2
0483   1034           03291         bcf     BYTE1           ; Clear all three
0484   10B4           03292         bcf     BYTE2           ; . . message receiving
0485   1134           03293         bcf     BYTE3           ; . . . . indicators
0486   30FF           03294         movlw   D_SYNC          ; Sync byte
0487   00A4           03295         movwf   d_sync_M
0488   083F           03296         movf    spectraaddress_M,w ; Get our address
0489   00A5           03297         movwf   d_address_M     ; Put it into the message
048A   2309           03298         call    dochecksum_0    ; First do the checksum
048B   2260           03299         call    byteread_1      ; Get a possible byte or three
048C   0824           03300         movf    d_sync_M,w      ; Now send it to the Spectra
048D   2428           03301         call    sendspectrabyte_0; D commands are 7 bytes long
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 75
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

048E   2260           03302         call    byteread_1      ; Get a possible byte or three
048F   0825           03303         movf    d_address_M,w   ; Address
0490   2428           03304         call    sendspectrabyte_0
0491   2260           03305         call    byteread_1      ; Get a possible byte or three
0492   0826           03306         movf    d_command1_M,w  ; Command 1
0493   2428           03307         call    sendspectrabyte_0
0494   2260           03308         call    byteread_1      ; Get a possible byte or three
0495   0827           03309         movf    d_command2_M,w  ; Command 2
0496   2428           03310         call    sendspectrabyte_0
0497   2260           03311         call    byteread_1      ; Get a possible byte or three
0498   0828           03312         movf    d_pan_speed_M,w ; Data 1
0499   2428           03313         call    sendspectrabyte_0
049A   2260           03314         call    byteread_1      ; Get a possible byte or three
049B   0829           03315         movf    d_tilt_speed_M,w; Data 2
049C   2428           03316         call    sendspectrabyte_0
049D   2260           03317         call    byteread_1      ; Get a possible byte or three
049E   082A           03318         movf    d_checksum_M,w  ; Checksum
049F   2428           03319         call    sendspectrabyte_0
04A0   2260           03320         call    byteread_1      ; Get a possible byte or three
                      03321
04A1   1C34           03322         btfss   BYTE1           ; Any thing in yet?
04A2   2CA8           03323         goto    xittospectra    ; Nope
                      03324
04A3   01CF           03325         clrf    temp2_0         ; Start time out counter
04A4   1CB4           03326         btfss   BYTE2           ; Some msg parts are here, how about the rest?
```

```
04A5   2CA9           03327         goto    getbyte2        ; Byte 2 not in yet
                      03328
04A6   1D34           03329         btfss   BYTE3           ; Last byte in yet?
04A7   2CB4           03330         goto    getbyte3        ; Nope
                      03331
04A8                  03332 xittospectra                    ; Exit from this mess
                      03333 ;        call    printspectraout_4; Show that command for debugging
04A8   0008           03334         return                  ; Back to the caller
                      03335
04A9                  03336 getbyte2
04A9   2260           03337         call    byteread_1      ; Get byte #2
04AA   18B4           03338         btfsc   BYTE2           ; Is it in yet                              0
04AB   2CB4           03339         goto    getbyte3        ; Got 2, get 3                              0
                      03340
04AC   1683           03341         bsf     status,rp0      ; RAM bank 1                                1
04AD   22EC           03342         call    delaynohang_0   ; Wait a bit                                1
04AE   1283           03343         bcf     status,rp0      ; Back to bank 0                            0
04AF   1C03           03344         btfss   status,c        ; Timed out?                                0
04B0   2CA9           03345         goto    getbyte2        ; Try some more
                      03346
04B1                  03347 clrflagsandexit
04B1   1034           03348         bcf     BYTE1           ; Timed out, clear flags
04B2   10B4           03349         bcf     BYTE2           ; . . better make it both flags
04B3   2CA8           03350         goto    xittospectra    ; Quit, while dropping the message
                      03351
04B4                  03352 getbyte3                        ; Got bytes 1 and 2, not get the last one
04B4   01CF           03353         clrf    temp2_0         ; Start time out counter
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 76
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

04B5                  03354 byte3again
04B5   2260           03355         call    byteread_1      ; Get it
04B6   1934           03356         btfsc   BYTE3           ; Got it yet?                               0
04B7   2CA8           03357         goto    xittospectra    ; More looking                              0
                      03358
04B8   1683           03359         bsf     status,rp0      ; RAM bank 1                                1
04B9   22EC           03360         call    delaynohang_0   ; Wait a bit                                1
04BA   1283           03361         bcf     status,rp0      ; Back to bank 0                            0
04BB   1C03           03362         btfss   status,c        ; Timed out?                                0
04BC   2CB5           03363         goto    byte3again      ; Try some more
                      03364
04BD   1134           03365         bcf     BYTE3           ; Make sure it's cleared
04BE   2CB1           03366         goto    clrflagsandexit ; Timed out, clear flags and quit this all
                      03367
                      03368
                      03369 ; End of the subroutines
                      03370
                      03371
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 77
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      03372         page
                      03373 ;;latex\subsection{System initialization}
                      03374 ;;latex\subsubsectionindex{Step 1 of initialization}
                      03375 ;;ps
                      03376 ; Initialize IO ports, etc.
                      03377 ;;pe
04BF                  03378 init_M                          ; Master clear, reset, power up, etc., entrance
04BF   1283           03379         bcf     status,rp0      ; Select RAM bank 0                         0
04C0   1303           03380         bcf     status,rp1      ; . . as the "standard RAM bank"            0
04C1   01B3           03381         clrf    mode1_M         ; Zero out what doesn't get properly
04C2   01B4           03382         clrf    mode2_M         ; . and automatically initialized
04C3   01B2           03383         clrf    mkmode1_M       ; Get multi-key mode control
04C4   0185           03384         clrf    porta           ; IO ports
04C5   0186           03385         clrf    portb
04C6   0187           03386         clrf    portc
                      03387
                      03388         ; Initialize the motion controls
04C7   3033           03389         movlw PAN_DEFAULT
04C8   00BC           03390         movwf saved_pan_speed_M
04C9   00D2           03391         movwf this_pan_speed_M
04CA   3024           03392         movlw TILT_DEFAULT
04CB   00BD           03393         movwf saved_tilt_speed_M
04CC   00D3           03394         movwf this_tilt_speed_M
04CD   01D0           03395         clrf this_command1_M
04CE   01D1           03396         clrf this_command2_M
```

```
                        03397
04CF   3000             03398          movlw  S_PROTO_LENGTH  ; Get the length of a normal S protocol cmnd
04D0   00C0             03399          movwf  sprotolength_M  ; Setup the normal message length.......
                        03400 ;;ps
                        03401 ; In this program the standard RAM bank is bank 0.  There will be some
                        03402 ; changes to RAM bank 1, but these will be rare and when this happens the
                        03403 ; is marked in the right hand column with the current RAM bank #.
                        03404 ;
                        03405 ; Do much of RAM bank 1 setup first, then select RAM bank 0 for the rest
                        03406 ; of setting up.
                        03407 ;;pe
                        03408 ; Start this in bank 1                                                    0
04D1   1683             03409          bsf     status,rp0      ; Select RAM bank 1                     1
                        03410 ; Be sure to disable the A/D converter                                    1
04D2   3006             03411          movlw   b'00000110'     ; Disables the A/D converter's input pins      1
04D3   009F             03412          movwf   adcon1          ; All analog inputs are now digital            1
                        03413 ;;ts
                        03414 ; porta is first                 It is only a 6 bit port, ignore upper bits     1
                        03415 ;                 0              7 x (IO pin does not exist)                    1
                        03416 ;                  0             6 x (IO pin does not exist)                    1
                        03417 ;                   1            5 Serial data from Spectra                     1
                        03418 ;                    0           4 spare                                        1
                        03419 ;                                                                             1
                        03420 ;                     0          3 Serial data to Spectra                       1
                        03421 ;                      0         2 Used to invert input/output data             1
                        03422 ;                       0        1 RS-422/485 input enable                      1
                        03423 ;                        0       0 spare                                        1
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 78
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                        03424 ;;te
                        03425 ; The following line is the correct one to use, however current boards are set 1
                        03426 ; up to allow direct monitoring of the input data by the CPU. If bit 4 is made 1
                        03427 ; an output it then drags down the output of the max chip.                    1
                        03428 ;        movlw   b'00100000'     ; Bits 5 = in, 0,1,2,3,4 = out, 6,7 don't care 1
04D4   3030             03429          movlw   b'00110000'     ; Bits 5 = in, 0,1,2,3,4 = out, 6,7 don't care 1
04D5   0085             03430          movwf   trisa           ; An set um up                                 1
                        03431 ;;ts
                        03432 ; portb is second                                                             1
                        03433 ;                 0              7 ICSP and spare otherwise                     1
                        03434 ;                  0             6 ICSP and debug output                        1
                        03435 ;                   1            5 Debug enable                                 1
                        03436 ;                    1           4 Permit using address 64                      1
                        03437 ;                                                                             1
                        03438 ;                     1          3 Address bit 3                                1
                        03439 ;                      1         2 Address bit 2                                1
                        03440 ;                       1        1 Address bit 1                                1
                        03441 ;                        1       0 Address bit 0                                1
                        03442 ;;te
04D6   303F             03443          movlw   b'00111111'     ; Bits 0 --> 5 are inputs, others are outputs  1
04D7   0086             03444          movwf   trisb           ;                                              1
                        03445 ;;ts
                        03446 ; portc is last                                                               1
                        03447 ;                1               7 Input data from the controller via the UART  1
                        03448 ;                 0              6 Output data to the head end, via the USART   1
                        03449 ;                  0             5 RS-422/485 output enable                     1
                        03450 ;                   0            4 spare                                        1
                        03451 ;                                                                             1
                        03452 ;                    0           3 spare                                        1
                        03453 ;                     0          2 spare                                        1
                        03454 ;                      0         1 spare                                        1
                        03455 ;                       0        0 spare                                        1
                        03456 ;;te
04D8   3080             03457          movlw   b'10000000'     ; 7 is an input, others are outputs            1
04D9   0087             03458          movwf   trisc           ;                                              1
                        03459 ;;ts
                        03460 ; Set timer to instruction counter with no prescale, page 31                  1
                        03461 ;                1               7 Disable portb pull-ups                       1
                        03462 ;                 0              6 Interrupt on falling edge                    1
                        03463 ;                  0             5 tmr0 clock source is instruction cycle clock 1
                        03464 ;                   0            4 pos transition                               1
                        03465 ;                    0           3 Prescaler to Timer0                          1
                        03466 ;                     000    012 Prescaler is set to 1:2                       1
                        03467 ;;te
04DA   3080             03468          movlw   b'10000000'     ; Set up tmr0                                  1
04DB   0081             03469          movwf   option_reg      ; Select the options                           1
04DC   018C             03470          clrf    pie1            ; Interrupts to disable                        1
04DD   018D             03471          clrf    pie2            ; Last interrupt to disable in RAM bank 1      1
04DE   018E             03472          clrf    pcon            ; Aren't using the brown out logic, so clr bits1
```

```
                     03473         ; Now its off to the standard RAM bank of 0                          1
04DF   1283          03474         bcf     status,rp0      ; Select RAM bank 0                          0
04E0   018B          03475         clrf    intcon          ; Disable all interrupts, intcon is in both ram0
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 79
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     03476 ;       clrf    pir1            ; More interrupts to disable                 0
                     03477 ;;ts
                     03478 ; Peripheral Interrupt Register 1                                            0
                     03479 ;       ---       0             7 Reserved                                   0
                     03480 ;       ---        0            6 Reserved                                   0
                     03481 ;       RCIF        0           5 Receive Interrupt flag                     0
                     03482 ;       TXIF         0          4 Transmit Interrupt flag                    0
                     03483 ;       SSPIF         0         3 Synchronous Serial Port Interrupt Flag     0
                     03484 ;       CCP1IF         0        2 Capture flag                               0
                     03485 ;       TMR2IF          0       1 TMR2 flag                                  0
                     03486 ;       TMR1IF           0      0 TMR1 flag                                  0
                     03487 ;;te
04E1   3000          03488         movlw   b'00000000'     ; Set up pir1                                0
04E2   008C          03489         movwf   pir1            ; Do it                                      0
04E3   018D          03490         clrf    pir2            ; Last interrupt to disable in RAM bank 0    0
                     03491 ;;ts
                     03492 ; Setup timer 1                                                              0
                     03493 ;       unused    0             7 Unimplemented                              0
                     03494 ;       unused     0            6 Unimplemented                              0
                     03495 ;       T1CKPS1     1           5 Timer 1 input clock prescale select bit    0
                     03496 ;       T1CKPS0      1          4 . . prescale by 8                          0
                     03497 ;       T1OSCEN       0         3 Oscillator on/off with off selected        0
                     03498 ;       T1SYNC         0        2 Ignored if TMR1CS = 0                      0
                     03499 ;       TMR1CS          0       1 Use internal/external clock, internal used 0
                     03500 ;       TMR1ON           0      0 Start/stop timer, 1 = start, 0 = stop      0
                     03501 ;;te
04E4   3030          03502         movlw   b'00110000'     ; Configure timer 1                          0
04E5   0090          03503         movwf   t1con           ; Do it                                      0
04E6   0192          03504         clrf    t2con           ; Clear out the timer #2's controls          0
04E7   0197          03505         clrf    ccp1con         ; Clear out the capture/compare control registe0
04E8   019D          03506         clrf    ccp2con         ; . . do both                                0
04E9   0194          03507         clrf    sspcon          ; Clear synchronous serial control register  0
                     03508 ;;ts
                     03509 ; Setup the USART, transmit first, page 99                                   0
                     03510 ;       CSRC      0             7 Synchronous mode clock select, async = nop 0
                     03511 ;       TX9        0            6 0 = 8 bit transmit mode, 1 = 9             0
                     03512 ;       TXEN        1           5 Transmit enable = 1, set to 1 to start xmit 0
                     03513 ;       SYNC         0          4 Asynchronous mode = 0, sync = 1            0
                     03514 ;       ---           0         3 Spare, unimplemented                       0
                     03515 ;       BRGH           0        2 Low speed mode for the baud rate generator 0
                     03516 ;       TRMT            0       1 Status of xmit reg                         0
                     03517 ;       TX9D             0      0 9th bit xmitted, spare in 8 bit mode       0
                     03518 ;;te
04EA   3020          03519         movlw   b'00100000'     ; Set up transmit status and control register 0
04EB   1683          03520         bsf     status,rp0      ; RAM bank 1                                 1
04EC   0098          03521         movwf   txsta           ; Load the xmit register                     1
04ED   3022          03522         movlw   BAUD4800        ; Sensormatic protocol only runs at 4800 baud 1
04EE   0099          03523         movwf   spbrg           ; Stick in the speed.                        1
04EF   1283          03524         bcf     status,rp0      ; RAM bank 0                                 0
                     03525 ;;ts
                     03526 ; Setup the USART, now the receive stuff, page 100                           0
                     03527 ;       SPEN      1             7 Serial port enable                         0
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 80
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     03528 ;       RX9        0            6 8 bit receive mode = 0, 1 = 9              0
                     03529 ;       SREN        0           5 Sync mode receive bit, async = nop         0
                     03530 ;       CREN         1          4 Asynchronous enable continuous receive     0
                     03531 ;       ---           0         3 Spare                                      0
                     03532 ;       FERR           0        2 Framing status error bit                   0
                     03533 ;       OERR            0       1 Overrun status error bit                   0
                     03534 ;       RX9D             0      0 Parity bit, not used in 8 bit mode         0
                     03535 ;;te
04F0   3090          03536         movlw   b'10010000'     ; Set up receive status and control register  0
04F1   0098          03537         movwf   rcsta           ; Load the receive register                  0
04F2   01AF          03538         clrf    lastchecksum_M  ; Force the last checksum to be OK (no msg yet)0
                     03539 ;        ; Now do we do debug mode?                                          0
                     03540 ;        bsf     DEBUG_MODE      ; Set global debug flag                     0
                     03541 ;        btfsc   DEBUG_MODE_ON   ; Test switch, bits come in backward        0
                     03542 ;        bcf     DEBUG_MODE      ; Nope, not debug mode                      0
```

```
                     03543 ;
04F3   1283          03544         bcf     status,rp0      ; RAM bank 0                                    0
04F4   1303          03545         bcf     status,rp1      ; . . Make double sure RAM bank 0               0
                     03546         ifdef   DEBUG_ADDRESS    ; In debugging mode, force the address        0
                     03547            goto    forceaddress ; Force it                                      0
                     03548         endif ; ifdef DEBUG_ADDRESS                                            0
                     03549
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE 81
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     03550         page
                     03551 ;;ps
                     03552 ; At this point all IO is properly set up.
                     03553 ;;pe
                     03554
                     03555
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE 82
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     03556         page
                     03557 ;;latex\subsubsectionindex{Step 2 of initialization}
                     03558 ;;ps
                     03559 ; Now  get the Spectra's address.  This is done by sending a query
                     03560 ; command out to the Spectra. Then waiting until the first byte comes in.
                     03561 ; The first byte is thrown away and the second byte is masked to seven
                     03562 ; active bits and is used as the Spectra's address.
                     03563 ;
                     03564 ;;latex\begin{enumerate}
                     03565 ;;ps
                     03566 ;;\item
                     03567 ; First wait until the Spectra is working.
                     03568 ;;pe
04F5                 03569 waituntilhigh                   ;
04F5   1E85          03570         btfss   SPECTRA_IN      ; Has the Spectra come on line yet?
04F6   2CF5          03571         goto    waituntilhigh   ; Nope
                     03572
04F7   22F7          03573         call    delayspectra_0  ; Now wait a bit until it calms down a bit
04F8   1E85          03574         btfss   SPECTRA_IN      ;
04F9   2CF5          03575         goto    waituntilhigh   ;
                     03576
                     03577 ;;ps
                     03578 ;;\item
                     03579 ; Now send a query command so as to get the Spectra's address.
                     03580 ;;pe
04FA                 03581 resendquery                     ; First wait in case the Spectra is transmitting
04FA   30C8          03582         movlw   200             ; 50 ms timeout waiting for Spectra to stop
04FB   22FC          03583         call    delayv_0
04FC   30C8          03584         movlw   200             ; 50 ms timeout waiting for Spectra to stop
04FD   22FC          03585         call    delayv_0        ; 100 ms total delay
                     03586         ; It takes about 29.6 ms to send this command
04FE   01A5          03587         clrf    d_address_M     ; Address byte
04FF   01A6          03588         clrf    d_command1_M    ; Command 1 byte
0500   3045          03589         movlw   D_QUERY         ; Command 2 byte, a query command
0501   00A7          03590         movwf   d_command2_M
0502   01A8          03591         clrf    d_pan_speed_M   ; Data 1 byte
0503   01A9          03592         clrf    d_tilt_speed_M  ; Data 2 byte
0504   2483          03593         call    tospectra_2     ; Send the query
0505   2363          03594         call    fromspectra_1   ; Get the Spectra's reply
                     03595         ifdef     DEBUG_ADDRESS ; Do we force the address
                     03596 forceaddress                      ; Yep
                     03597            movlw   DEBUG_ADDRESS   ; Get forced address
                     03598            movwf   spectraaddress_M; Stick it in
                     03599            movlw   0x00            ; Send a first byte
                     03600            call sendspectrabyte_0  ; (but I don't know why it is needed!)
                     03601            movlw   0x00            ; Send a first byte, twice
                     03602            call sendspectrabyte_0  ; (and I still don't know why it is needed!)
                     03603         endif ; ifdef DEBUG_ADDRESS
0506   22A7          03604         call    ckioerrs_0         ; Clear any transmitter/receiver errors
0507   22BD          03605         call cleareememory_1       ; Clear EE memory on first call, after that
                     03606                                    ; . . just keep reading it out.
                     03607
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE 83
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE
```

```
4175
4176                        03608 ;
4177                        03609 ;;ps
4178                        03610 ;;\item
4179                        03611 ; Display the software rev and current address switch settings.
4180                        03612 ;;pe
4181 0508   1633           03613         bsf     CLEAR_DISPLAY   ; Set flag to clear the Spectra display
4182 0509   0806           03614         movf    portb,w         ; Get block address switch
4183 050A   00CE           03615         movwf   temp1_0         ; Save it away
4184 050B   094E           03616         comf    temp1_0,w       ; Flip it, it comes in backward, i.e. 1's is 0's
4185 050C   393F           03617         andlw   0x3F            ; Dump slop
4186 050D   2230           03618         call    bin2hex_0       ; Convert switch value to hex
4187 050E   01A6           03619         clrf    d_command1_M    ; Byte #3 stays at zero
4188 050F   3015           03620         movlw   D_WRITE_CHAR    ; Command to write a character to the screen
4189 0510   00A7           03621         movwf   d_command2_M    ; Only need this till the data is displayed
4190 0511   3013           03622         movlw   19              ; Starting character position
4191 0512   00A8           03623         movwf   d_pan_speed_M   ; Stick it in too
4192                        03624
4193                        03625 ;;ps
4194                        03626 ;;\item
4195                        03627 ;       Identify ourselves on the second line.
4196                        03628 ;;pe
4197 0513   3054           03629         movlw   "T"             ; Get software rev put up
4198 0514   238E           03630         call    inlabel_3
4199 0515   3058           03631         movlw   "X"
4200 0516   238E           03632         call    inlabel_3
4201 0517   3042           03633         movlw   "B"
4202 0518   238E           03634         call    inlabel_3
4203 0519   302D           03635         movlw   "-"
4204 051A   238E           03636         call    inlabel_3
4205 051B   3053           03637         movlw   "S"
4206 051C   238E           03638         call    inlabel_3
4207 051D   3034           03639         movlw   "4"
4208 051E   238E           03640         call    inlabel_3
4209 051F   3032           03641         movlw   "2"
4210 0520   238E           03642         call    inlabel_3
4211 0521   3032           03643         movlw   "2"
4212 0522   238E           03644         call    inlabel_3
4213 0523   3020           03645         movlw   " "
4214 0524   238E           03646         call    inlabel_3
4215                        03647
4216                        03648 ;;ps
4217                        03649 ;;\item
4218                        03650 ;       Stick out the word "Rev  ".
4219                        03651 ;;pe
4220 0525   3052           03652         movlw   "R"
4221 0526   238E           03653         call    inlabel_3
4222 0527   3065           03654         movlw   "e"
4223 0528   238E           03655         call    inlabel_3
4224 0529   3076           03656         movlw   "v"
4225 052A   238E           03657         call    inlabel_3
4226 052B   3020           03658         movlw   " "
4227 052C   238E           03659         call    inlabel_3
4228 MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 84
4229 $Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
4230 Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
4231 LOC  OBJECT CODE     LINE SOURCE TEXT
4232   VALUE
4233
4234                        03660
4235                        03661 ;;ps
4236                        03662 ;;\item
4237                        03663 ; If bit 6 of the switch is set then say that this is debug mode.
4238                        03664 ; (This routine was changed on 29JAN02.)
4239                        03665 ;;pe
4240 052D   1A86           03666         btfsc  DEBUG_MODE_ON    ; Is ASCII terminal debugging enabled?
4241 052E   2D3A           03667         goto   notdebugmode     ; Nope, skip over this then
4242                        03668
4243 052F   3044           03669         movlw   "D"
4244 0530   238E           03670         call    inlabel_3
4245 0531   3065           03671         movlw   "e"
4246 0532   238E           03672         call    inlabel_3
4247 0533   3062           03673         movlw   "b"
4248 0534   238E           03674         call    inlabel_3
4249 0535   3075           03675         movlw   "u"
4250 0536   238E           03676         call    inlabel_3
4251 0537   3067           03677         movlw   "g"
4252 0538   238E           03678         call    inlabel_3
4253 0539   2D44           03679         goto    displayaddress
4254                        03680
4255 053A                   03681 notdebugmode
4256                        03682         ifndef    DEBUG
```

```
                          03683           ; Stick out the rev level
053A   3031               03684           movlw   "1"
053B   238E               03685           call    inlabel_3
053C   302E               03686           movlw   "."
053D   238E               03687           call    inlabel_3
053E   3030               03688           movlw   "0"
053F   238E               03689           call    inlabel_3
0540   3030               03690           movlw   "0"
0541   238E               03691           call    inlabel_3
0542   3020               03692           movlw   " "
0543   238E               03693           call    inlabel_3
                          03694
                          03695 ;;ps
                          03696 ;;\item
                          03697 ;        A rev level of BETA indicates that this is a trial version.
                          03698 ;;pe
                          03699 ; reference tag is "r44" to make finding it easer.
                          03700 ;;         movlw   "b"
                          03701 ;;         call    inlabel_3
                          03702 ;;         movlw   "E"
                          03703 ;;         call    inlabel_3
                          03704 ;;         movlw   "t"
                          03705 ;;         call    inlabel_3
                          03706 ;;         movlw   "a"
                          03707 ;;         call    inlabel_3
                          03708 ;;         movlw   " "
                          03709 ;;         call    inlabel_3
                          03710         endif ; ifndef DEBUG
                          03711
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 85
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                          03712         ifdef    DEBUG
                          03713 ;;ps
                          03714 ;;\item
                          03715 ;          A rev level of TEST indicates that this is a test version.
                          03716 ;;pe
                          03717            movlw   "T"
                          03718            call    inlabel_3
                          03719            movlw   "e"
                          03720            call    inlabel_3
                          03721            movlw   "S"
                          03722            call    inlabel_3
                          03723            movlw   "T"
                          03724            call    inlabel_3
                          03725            movlw   " "
                          03726            call    inlabel_3
                          03727         endif ; ifdef DEBUG
                          03728
                          03729 ;;ps
                          03730 ;;\item
                          03731 ;        Now get the block address that is loaded into the switch.
                          03732 ;;pe
0544                      03733 displayaddress
0544   082E               03734         movf    hexupper_0,w
0545   238E               03735         call    inlabel_3
0546   082D               03736         movf    hexlower_0,w
0547   238E               03737         call    inlabel_3
                          03738 ;;ps
                          03739 ;;\item
                          03740 ;       Done labeling the Spectra.
                          03741 ;
                          03742 ;;\item
                          03743 ;       Select normal mode inputs and disable comm outputs
                          03744 ;;\end{enumerate}
                          03745 ; (This routine was changed on 29JAN02.)
                          03746 ;;pe
0548   1287               03747         bcf     ENABLE_OUTPUTS  ; Disable response drivers
0549   1085               03748         bcf     ENABLE_INPUTS   ; Enable RS-485 input receivers
054A   1E86               03749         btfss   DEBUG_MODE_ON   ; Is ASCII terminal debugging enabled?
054B   22E7               03750         call    crlf_2          ; Always start on a new line
054C   30C1               03751         movlw   S_POWERED_UP    ; Say that we are working
054D   245C               03752         call    threebytemessage_6 ; Send it
054E   018E               03753         clrf    tmr1l           ; Clear timer 1's
054F   018F               03754         clrf    tmr1h           ; . . counting registers
0550   01A6               03755         clrf    d_command1_M    ; Start with these
0551   01A7               03756         clrf    d_command2_M    ; . . cleared
0552   2801               03757         goto    startstart      ; And get going
                          03758
```

```
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 86
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      03759         page
                      03760 ;;ps
                      03761 ; Read in a controller message.
                      03762 ;
                      03763 ; Moved to the end so that a computed goto will work. With this where it
                      03764 ; would most logically go, the computed goto for input command processing
                      03765 ; will be too high in memory.
                      03766 ;;pe
0553                  03767 getinput
                      03768         ; Start looking for data
0553   08AF           03769         movf    lastchecksum_M,f; Test old checksum
0554   1D03           03770         btfss   status,z        ; If old checksum was OK, don't resync
0555   22A7           03771         call    ckioerrs_0      ; Check and clear any possible errors
0556   2386           03772         call    getsensorbyte_1 ; Get address
0557   19B3           03773         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
0558   2803           03774         goto    again
                      03775
0559   00C1           03776         movwf   sproto1_M       ; Save address
055A   023F           03777         subwf   spectraaddress_M,w; Is this our address?
055B   1D03           03778         btfss   status,z        ; Check
055C   2803           03779         goto    again           ; Nope
                      03780
055D   2386           03781         call    getsensorbyte_1 ; Get command
055E   19B3           03782         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
055F   2803           03783         goto    again
                      03784
                      03785         ; Most commands are 3 bytes long, some are longer. This is the
                      03786         ; logic to get more bytes in
0560   00C2           03787         movwf   sproto2_M       ; Save command, then check for long commands
0561   3CA6           03788         sublw   S_GOTO_POSITION ; 0xA6
0562   1D03           03789         btfss   status,z        ; Check
0563   2D66           03790         goto    checkC0         ; Nope
                      03791
0564   300D           03792         movlw   S_GOTO_POSITION_SIZE
0565   2D77           03793         goto    getchecksum     ; Stick in new length and get the rest
                      03794
0566                  03795 checkC0
0566   0842           03796         movf    sproto2_M,w     ; Get command
0567   3CC0           03797         sublw   S_VARIABLE_SPEED; 0xC0
0568   1D03           03798         btfss   status,z
0569   2D6C           03799         goto    checkC1
                      03800
056A   3005           03801         movlw   S_VARIABLE_SPEED_SIZE
056B   2D77           03802         goto    getchecksum
                      03803
056C                  03804 checkC1
056C   0842           03805         movf    sproto2_M,w     ; Get command
056D   3CC1           03806         sublw   S_UNKNOWN_C1    ; 0xC1
056E   1D03           03807         btfss   status,z
056F   2D72           03808         goto    checkC3
                      03809
0570   30C1           03810         movlw   S_UNKNOWN_C1
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 87
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0571   2D77           03811         goto    getchecksum
                      03812
0572                  03813 checkC3
0572   0842           03814         movf    sproto2_M,w     ; Get command
0573   3CC3           03815         sublw   S_PROP_SPEED    ; 0xC3
0574   1D03           03816         btfss   status,z
0575   2D78           03817         goto    notlong
                      03818
0576   30C3           03819         movlw   S_PROP_SPEED
0577                  03820 getchecksum
0577   00C0           03821         movwf   sprotolength_M  ; Stick it in
0578                  03822 notlong
0578   2386           03823         call    getsensorbyte_1 ; Get the checksum
0579   19B3           03824         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
057A   2803           03825         goto    again           ; Yep, retry
                      03826
057B   00C3           03827         movwf   sproto3_M       ; Save checksum
057C   0742           03828         addwf   sproto2_M,w     ; Check the
```

```
057D   1903           03829         btfsc   status,z        ; . partial checksum if = zero
057E   2803           03830         goto    again           ; . . is an error
                      03831
057F   0741           03832         addwf   sproto1_M,w     ; Get full checksum and
0580   00AF           03833         movwf   lastchecksum_M  ; . save it
0581   0840           03834         movf    sprotolength_M,w; Have we read in the full message? 0 = normal
0582   1903           03835         btfsc   status,z        ; Check
0583   2DC5           03836         goto    allin
                      03837
0584   2386           03838         call    getsensorbyte_1 ; Get next byte
0585   19B3           03839         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
0586   2803           03840         goto    again
                      03841
0587   00C4           03842         movwf   sproto4_M       ; Save byte
0588   0842           03843         movf    sproto2_M,w     ; Get number of bytes to get
0589   3CC3           03844         sublw   S_PROP_SPEED    ; Is this 0xC3? (i.e. 4 bytes long)
058A   1903           03845         btfsc   status,z        ; Check
058B   2DC0           03846         goto    checksumC3
                      03847
058C   2386           03848         call    getsensorbyte_1 ; Get next byte
058D   19B3           03849         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
058E   2803           03850         goto    again
                      03851
058F   00C5           03852         movwf   sproto5_M       ; Save next byte
0590   0842           03853         movf    sproto2_M,w     ; Get number of bytes to get
0591   3CC0           03854         sublw   S_VARIABLE_SPEED; Is this 0xC0? (i.e. 5 bytes long)
0592   1903           03855         btfsc   status,z        ; Check
0593   2DBF           03856         goto    checksumC0
                      03857
0594   2386           03858         call    getsensorbyte_1 ; Get a byte
0595   19B3           03859         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
0596   2803           03860         goto    again
                      03861
0597   00C6           03862         movwf   sproto6_M       ; Save a byte
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 88
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0598   2386           03863         call    getsensorbyte_1 ; Get a byte
0599   19B3           03864         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
059A   2803           03865         goto    again
                      03866
059B   00C7           03867         movwf   sproto7_M       ; Save a byte
059C   2386           03868         call    getsensorbyte_1 ; Get a byte
059D   19B3           03869         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
059E   2803           03870         goto    again
                      03871
059F   00C8           03872         movwf   sproto8_M       ; Save a byte
05A0   2386           03873         call    getsensorbyte_1 ; Get a byte
05A1   19B3           03874         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
05A2   2803           03875         goto    again
                      03876
05A3   00C9           03877         movwf   sproto9_M       ; Save a byte
05A4   2386           03878         call    getsensorbyte_1 ; Get a byte
05A5   19B3           03879         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
05A6   2803           03880         goto    again
                      03881
05A7   00CA           03882         movwf   sproto10_M      ; Save a byte
05A8   2386           03883         call    getsensorbyte_1 ; Get a byte
05A9   19B3           03884         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
05AA   2803           03885         goto    again
                      03886
05AB   00CB           03887         movwf   sproto11_M      ; Save a byte
05AC   2386           03888         call    getsensorbyte_1 ; Get a byte
05AD   19B3           03889         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
05AE   2803           03890         goto    again
                      03891
05AF   00CC           03892         movwf   sproto12_M      ; Save a byte
05B0   0842           03893         movf    sproto2_M,w     ; Get number of bytes to get
05B1   3CC1           03894         sublw   S_UNKNOWN_C1    ; Is this 0xC1? (i.e. 12 bytes long)
05B2   1903           03895         btfsc   status,z        ; Check
05B3   2DB8           03896         goto    checksumC1
                      03897
05B4   2386           03898         call    getsensorbyte_1 ; Get a byte
05B5   19B3           03899         btfsc   INPUT_ERROR     ; Did we have an error, 1 = yes, 0 = no
05B6   2803           03900         goto    again
                      03901
05B7   00CD           03902         movwf   sproto13_M      ; Must be 0xA6, save it (i.e. 13 bytes long)
                      03903 ;
                      03904 ; Note that in this checksum logic that we always get to the
```

```
                     03905 ; checksumxx entries with w = 0. Otherwise this won't work.
                     03906 ;
05B8                 03907 checksumC1
05B8   074C          03908         addwf   sproto12_M,w    ; Do checksum
05B9   074B          03909         addwf   sproto11_M,w
05BA   074A          03910         addwf   sproto10_M,w
05BB   0749          03911         addwf   sproto9_M,w
05BC   0748          03912         addwf   sproto8_M,w
05BD   0747          03913         addwf   sproto7_M,w
05BE   0746          03914         addwf   sproto6_M,w
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 89
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

05BF                 03915 checksumC0
05BF   0745          03916         addwf   sproto5_M,w
05C0                 03917 checksumC3
05C0   0744          03918         addwf   sproto4_M,w
05C1   0743          03919         addwf   sproto3_M,w
05C2   0742          03920         addwf   sproto2_M,w
05C3   0741          03921         addwf   sproto1_M,w
05C4   00AF          03922         movwf   lastchecksum_M  ; Save checksum results 0 = good, 0-not = bad
05C5                 03923 allin
05C5   1E06          03924         btfss   PERMIT64        ; Do we permit processing of address 64?
05C6   2DCB          03925         goto    sixtyfourok     ; Yep
                     03926
05C7   0841          03927         movf    sproto1_M,w     ; Get the command's address
05C8   3C40          03928         sublw   IGNORE_ADDRESS  ; Is this for address 64?
05C9   1903          03929         btfsc   status,z        ; Test
05CA   2803          03930         goto    again           ; Address 64 is always ignored
                     03931
05CB                 03932 sixtyfourok
05CB   083F          03933         movf    spectraaddress_M,w; Get our address
05CC   0241          03934         subwf   sproto1_M,w     ; Is this for us?
05CD   1D03          03935         btfss   status,z        ; Check
05CE   2803          03936         goto    again           ; Nope, retry
                     03937
05CF   082F          03938         movf    lastchecksum_M,w; Is the checksum any good?
05D0   1D03          03939         btfss   status,z        ; Check
05D1   2803          03940         goto    again           ; Bad checksum
                     03941
05D2   23A4          03942         call    printsensorin_4 ; Show it
05D3   0842          03943         movf    sproto2_M,w     ; Good address, get opcode
05D4   2813          03944         goto    decodeinput_M   ;
                     03945
                     03946
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 90
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     03947         page
                     03948 ;;latex\subsectionindex{Multibyte responses}
                     03949 ;;ps
                     03950 ; Multibyte responses are sent from here.
                     03951 ;
                     03952 ; There are three "long" responses to deal with. They are:
                     03953 ;;pe
                     03954 ;;ts
                     03955 ;       software version        10 byte response
                     03956 ;       preset position         12 byte response
                     03957 ;       memory dump             104 byte response
                     03958 ;;te
                     03959
                     03960         space 1
                     03961 ;;latex\subsubsectionindex{getposition}
                     03962 ;;ps
                     03963 ; getposition is used to provide a fake value to the controller as to
                     03964 ; where the Spectra is pointing. There is a basic problem with
                     03965 ; Sensormatic vs. Pelco in the way in which presets are handled. The
                     03966 ; problem is that on the newer domes, Sensormatic sends a command to the
                     03967 ; dome that asks "where are you pointing". The dome then tells the
                     03968 ; controller where it is pointing, etc., and the controller does the
                     03969 ; remembering. When the operator wants to go to, say, preset 5, the
                     03970 ; controller has to send all pointing information to the dome. Pelco's
                     03971 ; approach requires that the dome remember where it is when it is told to
                     03972 ; "set preset 5" and then the head end just sends "goto preset 5" to the
                     03973 ; dome.
```

```
                     03974 ;
                     03975 ; In order to fake out the Sensormatic system, we generate a fake reply
                     03976 ; message as to where we are pointing. The fake message is generated by
                     03977 ; using an old valid reply message from dome #5 and changing some of the
                     03978 ; fields so that the changed fields indicate a usable preset number.
                     03979 ;
                     03980 ; The basic message from dome #5, a SpeedDome 2000, is:
                     03981 ;;pe
                     03982 ;;ts
                     03983 ;      0x01 0x00 0x00 0x12 0x4C 0x26 0x5D 0x28 0x37 0x31 0x37 0x57
                     03984 ;;te
                     03985 ;;ps
                     03986 ; Translated this becomes:
                     03987 ;;pe
                     03988 ;;ts
                     03989 ;      Iris xunning    = 0x00       (Haven't yet figured out that first letter).
                     03990 ;      Zoom limit      = 0x00
                     03991 ;      Tilt Position   = 0x12 0x4C
                     03992 ;      Zoom Position   = 0x26 0x5D
                     03993 ;      Electronic Zoom = 0x28 0x37
                     03994 ;      Pan Position    = 0x31 0x37
                     03995 ;;te
                     03996 ;;ps
                     03997 ; After the ACK the response is:
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 91
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     03998 ; (All hex fields are common to all position requests.)
                     03999 ;;pe
                     04000 ;;ts
                     04001 ;            Byte  Use                  Contents
                     04002 ;             1    Camera ID            Camera ID
                     04003 ;             2    Iris Data            0x00
                     04004 ;             3    Zoom Limit           0x00
                     04005 ;            4,5   Tilt Position        0x12, Preset ID
                     04006 ;            6,7   Zoom Position        0x26, Preset ID
                     04007 ;            8,9   Electronic Zoom      0x28, Preset ID
                     04008 ;           10,11  Pan Position         0x31, Preset ID
                     04009 ;            12    Checksum             Checksum
                     04010 ;;te
                     04011 ;;ps
                     04012 ; Values that get stuck in the reply message consist of the preset ID and
                     04013 ; various constants from a real breakout dump from dome #5. If there are
                     04014 ; any "strange" value range checks made, this should confuse them and get
                     04015 ; the controller to pass the values as OK.
                     04016 ; (This routine was last updated on 25JAN02.)
                     04017 ;;pe
05D5                 04018 getposition                    ; 0xA5 Request Dome position Coordinates
                     04019                                ; 3 bytes in, 12 bytes back
05D5   23FC         04020         call    sendack_5      ; First off send the ACK
05D6   083F         04021         movf    spectraaddress_M,w ; Get my address
05D7   00C1         04022         movwf   sproto1_M      ; Stick it into the message
05D8   3000         04023         movlw   0x00
05D9   00C2         04024         movwf   sproto2_M      ; Iris data
05DA   00C3         04025         movwf   sproto3_M      ; Zoom limit
05DB   3012         04026         movlw   0x12           ; Tilt position
05DC   00C4         04027         movwf   sproto4_M
05DD   2392         04028         call    presetincrement_1 ; Get this Preset number
05DE   00A9         04029         movwf   d_tilt_speed_M  ; Stick this preset number into D protocol msg
05DF   00C5         04030         movwf   sproto5_M      ; Stick this preset number everywhere
05E0   00C7         04031         movwf   sproto7_M      ; I put the preset number in many places
05E1   00C9         04032         movwf   sproto9_M      ; . and stuck information from breakout dumps
05E2   00CB         04033         movwf   sproto11_M     ; . . in other places. Hope that this is OK.
05E3   3026         04034         movlw   0x26           ; Zoom position
05E4   00C6         04035         movwf   sproto6_M
05E5   3028         04036         movlw   0x28           ; Electronic zoom
05E6   00C8         04037         movwf   sproto8_M
05E7   3033         04038         movlw   0x33           ; Pan position
05E8   00CA         04039         movwf   sproto10_M
05E9   074B         04040         addwf   sproto11_M,w
05EA   0749         04041         addwf   sproto9_M,w
05EB   0748         04042         addwf   sproto8_M,w
05EC   0747         04043         addwf   sproto7_M,w
05ED   0746         04044         addwf   sproto6_M,w
05EE   0745         04045         addwf   sproto5_M,w
05EF   0744         04046         addwf   sproto4_M,w
                     04047 ;       addwf   sproto3_M,w    ; Are zero so not checksummed
                     04048 ;       addwf   sproto2_M,w    ; Are zero so not checksummed
05F0   0741         04049         addwf   sproto1_M,w
```

```
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 92
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

05F1   00CC           04050         movwf   sproto12_M      ; Preliminaryly stick it into the message
05F2   09CC           04051         comf    sproto12_M,f    ; Now complement it
                      04052                                 ; Actual checksum is all bytes added together
                      04053                                 ; and subtracted from 0x100. But this gives
                      04054                                 ; the same answer.
05F3   0ACC           04055         incf    sproto12_M,f    ; Not 1's complement make it 2's complement
05F4   2225           04056         call    arrowout_3      ; Enable output drivers
05F5   0841           04057         movf    sproto1_M,w     ; Send message
05F6   246C           04058         call    tosensormatic_4
05F7   0842           04059         movf    sproto2_M,w
05F8   246C           04060         call    tosensormatic_4
05F9   0843           04061         movf    sproto3_M,w
05FA   246C           04062         call    tosensormatic_4
05FB   0844           04063         movf    sproto4_M,w
05FC   246C           04064         call    tosensormatic_4
05FD   0845           04065         movf    sproto5_M,w
05FE   246C           04066         call    tosensormatic_4
05FF   0846           04067         movf    sproto6_M,w
0600   246C           04068         call    tosensormatic_4
0601   0847           04069         movf    sproto7_M,w
0602   246C           04070         call    tosensormatic_4
0603   0848           04071         movf    sproto8_M,w
0604   246C           04072         call    tosensormatic_4
0605   0849           04073         movf    sproto9_M,w
0606   246C           04074         call    tosensormatic_4
0607   084A           04075         movf    sproto10_M,w
0608   246C           04076         call    tosensormatic_4
0609   084B           04077         movf    sproto11_M,w
060A   246C           04078         call    tosensormatic_4
060B   084C           04079         movf    sproto12_M,w    ; Send checksum
060C   246C           04080         call    tosensormatic_4
                      04081         ; Send D protocol message, d_tilt_speed_M is already set up
060D   01A6           04082         clrf    d_command1_M    ; Get ready to send the preset to the Spectra
060E   3003           04083         movlw   D_SET_PRESET    ; Get op-code
060F   00A7           04084         movwf   d_command2_M    ; Stick op-code in
0610   01A8           04085         clrf    d_pan_speed_M
0611   2334           04086         call    endout_1        ; Endup sending the message
0612   2483           04087         call    tospectra_2
                      04088
                      04089 ;;ps
                      04090 ; Set up for a timer to run, sos I will not increment the preset
                      04091 ; ID during the next one or two requests for my position.
                      04092 ;;pe
0613   018E           04093         clrf    tmr1l           ; Zero should be a maximum value
0614   018F           04094         clrf    tmr1h           ; . . timer increments on each instruction
0615   01A3           04095         clrf    continuationtimer_M ; Start the long duration timer at zero
0616   1410           04096         bsf     t1con,tmr1on    ; Start the timer going
0617   2A0B           04097         goto    alldone
                      04098


                      04099         space 1
                      04100 ;;latex\subsubsectionindex{version}
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 93
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE


                      04101 ;;ps
                      04102 ; This is the reply made by dome #5.
                      04103 ; (This routine was last updated on 25JAN02.)
                      04104 ;;pe
                      04105 ;;ts
                      04106 ; Version Response is:
                      04107 ;
                      04108 ;          Byte  Use                    Contents
                      04109 ;           1    Camera ID              Camera ID
                      04110 ;           2    Op Code                0xC9
                      04111 ;           3    Unknown byte           0x06
                      04112 ;         4 --> 9 Software rev          0x07, 0x01, 0x00, 0x01, 0x03, 0x16
                      04113 ;           10   Checksum               Checksum
                      04114 ;;te
0618                  04115 version                         ; 0xC9 Get software version number from dome
                      04116                                 ; 3 bytes in, 10 bytes back
0618   23FC           04117         call    sendack_5       ; First off send the ACK
0619   2225           04118         call    arrowout_3      ; Enable output drivers
```

```
061A   01A2          04119        clrf    checksum_M      ; Start out with a clean checksum
061B   083F          04120        movf    spectraaddress_M,w ; Get my address
061C   2425          04121        call    sendone_5
061D   30C9          04122        movlw   S_SOFTWARE_VERSION; Op code
061E   2425          04123        call    sendone_5
061F   3006          04124        movlw   0x06
0620   2425          04125        call    sendone_5
0621   3007          04126        movlw   0x07
0622   2425          04127        call    sendone_5
0623   3001          04128        movlw   0x01
0624   2425          04129        call    sendone_5
0625   3000          04130        movlw   0x00
0626   2425          04131        call    sendone_5
0627   3001          04132        movlw   0x01
0628   2425          04133        call    sendone_5
0629   3003          04134        movlw   0x03
062A   2425          04135        call    sendone_5
062B   3016          04136        movlw   0x16
062C   2425          04137        call    sendone_5
062D   2401          04138        call    sendchecksum_6
062E   29BF          04139        goto    donecommand
                     04140
                     04141
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE 94
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     04142        page
                     04143 ;;latex\subsubsectionindex{gotoposition}
                     04144 ;;ps
                     04145 ; Input goto command is: (Only bytes 1, 2, 4 and 13 are used.)
                     04146 ;;pe
                     04147 ;;ts
                     04148 ;           Byte  Use                  Contents
                     04149 ;
                     04150 ;            1    Camera ID            Camera ID
                     04151 ;            2    Op Code              0xA6
                     04152 ;           3,4   Pan Position         0x33, Preset ID only preset ID used
                     04153 ;           5,6   Tilt Position        0x06, Preset ID . . is in byte 4
                     04154 ;           7,8   Zoom Position        0x26, Preset ID
                     04155 ;           9,10  Digital Zoom         0x28, Preset ID
                     04156 ;            11   Iris Offset         0x00
                     04157 ;            12   Zoom Limit          0x00
                     04158 ;            13   Checksum            Checksum
                     04159 ;;te
062F                 04160 gotoposition                  ; 0xA6 Goto absolute position
                     04161                               ; 13 bytes in, 3 bytes back (ACK)
062F   01A6          04162        clrf    d_command1_M    ; Get ready to send the preset to the Spectra
0630   3007          04163        movlw   D_GOTO_PRESET   ; Get op-code
0631   00A7          04164        movwf   d_command2_M    ; Stick op-code in
0632   01A8          04165        clrf    d_pan_speed_M
0633   0844          04166        movf    sproto4_M,w     ; Get the preset ID
0634   00A9          04167        movwf   d_tilt_speed_M  ; Stick preset ID in message
0635   29F5          04168        goto    sendcommand     ; Send an ACK and the Spectra's command
                     04169
                     04170
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE 95
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     04171        page
                     04172 ;;latex\subsubsectionindex{memorydump}
                     04173 ;;ps
                     04174 ; Reply to a memory dump request (0xA4).
                     04175 ;
                     04176 ; The format of the reply is copied from dome #5, a SpeedDome 2000,
                     04177 ; immediately following a power up. I.e. a "clean" dome with no motion
                     04178 ; commands, or anything else, loaded into it.
                     04179 ; (This routine was last updated on 25JAN02.)
                     04180 ;
                     04181 ; The message's content was: (The first byte is the address of the dome
                     04182 ; and the last byte is the checksum.)
                     04183 ;;pe
                     04184 ;;ts
                     04185 ;          1    2    3    4    5    6    7    8    9   10
                     04186 ;       1-01 1-66 1-0E 1-80 1-00 1-00 1-00 1-00 1-00 1-00
                     04187 ;         11   12   13   14   15   16   17   18   19   20
                     04188 ;       1-00 1-00 1-72 1-00 1-00 1-00 1-00 1-00 1-00 1-00
```

```
                    04189 ;         21   22   23   24   25   26   27   28   29   30
                    04190 ;       1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
                    04191 ;         31   32   33   34   35   36   37   38   39   40
                    04192 ;       1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
                    04193 ;         41   42   43   44   45   46   47   48   49   50
                    04194 ;       1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
                    04195 ;         51   52   53   54   55   56   57   58   59   60
                    04196 ;       1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-0E 1-00
                    04197 ;         61   62   63   64   65   66   67   68   69   70
                    04198 ;       1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
                    04199 ;         71   72   73   74   75   76   77   78   79   80
                    04200 ;       1-DF 1-FF 1-22 1-0E 1-00 1-00 1-00 1-00 1-00 1-00
                    04201 ;         81   82   83   84   85   86   87   88   89   90
                    04202 ;       1-00 1-00 1-00 1-00 1-00 1-DF 1-FF 1-21 1-0E 1-00
                    04203 ;         91   92   93   94   95   96   97   98   99  100
                    04204 ;       1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00 1-00
                    04205 ;        101  102  103  104
                    04206 ;       1-DF 1-FF 1-20 1-72   < 1: Dome memory dump msg length = 104>
                    04207 ;;te
0636                04208 memorydump
0636   2225         04209         call    arrowout_3      ; Stick out an arrow and start outputs
0637   01A2         04210         clrf    checksum_M      ; Start out with a clean checksum
0638   083F         04211         movf    spectraaddress_M,w ; Get my address, Byte 1
0639   2425         04212         call    sendone_5       ; Send it
063A   3066         04213         movlw   0x66            ; Byte 2
063B   2425         04214         call    sendone_5
063C   300E         04215         movlw   0x0E            ; Byte 3
063D   2425         04216         call    sendone_5
063E   3080         04217         movlw   0x80            ; Byte 4
063F   2425         04218         call    sendone_5
0640   3008         04219         movlw   8               ; Bytes 5 through 12
0641   241F         04220         call    sendmany0_5
0642   3072         04221         movlw   0x72            ; Byte 13
0643   2425         04222         call    sendone_5
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 96
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

0644   302D         04223         movlw   45              ; Bytes 14 through 58
0645   241F         04224         call    sendmany0_5
0646   300E         04225         movlw   0x0E            ; Byte 59
0647   2425         04226         call    sendone_5
0648   300B         04227         movlw   11              ; Bytes 60 through 70
0649   241F         04228         call    sendmany0_5
064A   30DF         04229         movlw   0xDF            ; Byte 71
064B   2425         04230         call    sendone_5
064C   30FF         04231         movlw   0xFF            ; Byte 72
064D   2425         04232         call    sendone_5
064E   3022         04233         movlw   0x22            ; Byte 73
064F   2425         04234         call    sendone_5
0650   300E         04235         movlw   0x0E            ; Byte 74
0651   2425         04236         call    sendone_5
0652   300B         04237         movlw   11              ; Bytes 75 through 85
0653   241F         04238         call    sendmany0_5
0654   30DF         04239         movlw   0xDF            ; Byte 86
0655   2425         04240         call    sendone_5
0656   30FF         04241         movlw   0xFF            ; Byte 87
0657   2425         04242         call    sendone_5
0658   3021         04243         movlw   0x21            ; Byte 88
0659   2425         04244         call    sendone_5
065A   300E         04245         movlw   0x0E            ; Byte 89
065B   2425         04246         call    sendone_5
065C   300B         04247         movlw   11              ; Bytes 90 through 100
065D   241F         04248         call    sendmany0_5
065E   30DF         04249         movlw   0xDF            ; Byte 101
065F   2425         04250         call    sendone_5
0660   30FF         04251         movlw   0xFF            ; Byte 102
0661   2425         04252         call    sendone_5
0662   3020         04253         movlw   0x20            ; Byte 103
0663   2425         04254         call    sendone_5
0664   2401         04255         call    sendchecksum_6  ; Send the checksum
0665   2803         04256         goto    again
                    04257
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 97
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                    04258         page
```

```
                     04259 ;        org     PROGRAM_PAGE1
0666   0043 006F 0070 04260         de      "Copyright Pelco $Id: txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
                           "
       0079 0072 0069
       0067 0068 0074
       0020 0050 0065
       006C 0063 006F
       0020 0024 0049
       0064 003A 0020
       0074 0078 0062
       0073 0034 0032
       0032 002E 0061
       0073 006D 002C
       0076 0020 0031
       002E 0031 0032
       0031 0020 0032
       0030 0030 0032
       002D 0030 0033
       002D 003
                     04261
                     04262
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 98
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                     04263         page
                     04264 ;;latex\subsection{Outline of operation of TXB-S422 Translator\label{outline}}
                     04265 ;;ps
                     04266 ; When initialized the TXB-S422 Translator, sets up some IO configuration
                     04267 ; information.  It then reads the DIP switch and obtains configuration
                     04268 ; information to use and saves it away. It then sends a query command to
                     04269 ; the Spectra, waits for its reply and uses the resulting address
                     04270 ; information to determine what address the Spectra is set to. The
                     04271 ; TXB-S422 then places its ID message on the Spectra's screen and enters
                     04272 ; an endless loop looking for messages from the controller/head-end. The
                     04273 ; endless loop is contained in the getsensorbyte_1 routine. The endless
                     04274 ; loop is used to receive commands in and translate them into commands
                     04275 ; for the Spectra. Translated commands are sent to the Spectra in
                     04276 ;  D-Protocol at 2400 baud and we go back for some more.
                     04277 ;
                     04278 ; Information about the applicable part numbers for this project is at
                     04279 ; the end of this program listing as is a change log.
                     04280 ;;pe
                     04281
                     04282 ;;latex\subsection{Change log}
                     04283 ;;ps
                     04284 ; Various phrasing of the word "beta" are used to differentiate different
                     04285 ; test versions of this code. The following types of "beta" have been
                     04286 ; used:
                     04287 ;;pe
                     04288 ;;ts
                     04289 ;  1       2       3       4       5       6       7       8       9
                     04290 ; "BETA", "Beta", "BEta", "BETa", "bETA", "beTA", "beTx", "betA", "beta"
                     04291 ;
                     04292 ;  10      11
                     04293 ; "beTa", "bEta"
                     04294 ;
                     04295 ; beta's not yet used:
                     04296 ; BEtA BeTA BeTa BetA bETa bEtA
                     04297 ;;te
                     04298 ;;ps
                     04299 ; As additional "beta test" versions of the code are needed, be sure to
                     04300 ; use a different spelling of "beta" and update the above table. The
                     04301 ; various spellings of "beta" are somewhere beteemn line numbers 3700 and
                     04302 ; 3900 of this listing. "There is a reference tag of 'r44' located
                     04303 ; there."
                     04304 ;
                     04305 ; Prior to rev "beTx", I thought that I could remember all the diffrences
                     04306 ; between the diffrent beta-revs. Time has proved me wrong, so I'm
                     04307 ; writing down the various things that I think have been fixed. (I have
                     04308 ; also written down what I have changed and added too.)
                     04309 ;;pe
                     04310 ;;latex\begin{enumerate}
                     04311 ;;ps
                     04312 ;;\item "beTx" is a special version of the TXBS422 code that has enabled switch
                     04313 ; position 3 to control one of two versions of pan/tilt speeds. If the
                     04314 ; switch is off then the option 1 for speeds is used, if it is on the
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 99
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
```

```
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      04315 ; option 2 is used. The use of two options for speed calculations is
                      04316 ; being done in an attempt to make the Spectra behave more like an
                      04317 ; UltraDome in its operator response.
                      04318 ;
                      04319 ; Speed option 1: Input Pan speed is multiplied by 3/4, has 8 added to it
                      04320 ; and is limit checked at 63, anything more than 63 is turned into 64 for
                      04321 ; turbo mode. Input Tilt speed is multiplied by 1/2 and has 6 added to
                      04322 ; it, it is then limit checked at 63, nothing greater than 63 is
                      04323 ; permitted for tilt speeds.
                      04324 ;
                      04325 ; Speed option 2: Input Pan speed has 8 added to it and is limit checked
                      04326 ; at 63, anything more than 63 is turned into 64 for turbo mode. Input
                      04327 ; Tilt speed has 6 added to it, is then limit checked at 63, nothing
                      04328 ; greater than 63 is permitted for tilt speeds.
                      04329 ;
                      04330 ;;\item "betA" changes the speeds once again and now bits 0 and 1 of the
                      04331 ; address switch are used to select the divisor values for pan and tilt
                      04332 ; speeds. 0 = none, 1 = 3/4, 2 = 1/2 and 3 = 1/4. (I did this in response
                      04333 ; to a conversation I had with a potential customer.) Logic for adding 8
                      04334 ; or 6 and range limit checking has not changed.
                      04335 ;
                      04336 ;;\item "beta" makes it so that when going from normal speeds, to fast speeds
                      04337 ; and then back to normal speeds, that the unit dosen't stop movment.
                      04338 ; (Actually it didn't stop, it just went verrrry slooowly. The slow speed
                      04339 ; was caused by sending a speed command of "0" to the Spectra.)
                      04340 ;
                      04341 ; An additional use has been found for the switch. And that is to provide
                      04342 ; for either SpeedDome or UltraDome timing/modes of operation. The
                      04343 ; primary diffrence is with the timing of output signals. If SW-3 is off,
                      04344 ; which is the default, then all messages to the controller assert the
                      04345 ; communications line for about 250 us before sending the first bit and
                      04346 ; for about 1 ms after the last bit. This is about what a SpeedDome does
                      04347 ; and the reply byte for a dome type query request (poll) is set to 0xF8,
                      04348 ; the SpeedDome's response.  If SW-3 is on, then there are almost no
                      04349 ; assertion delays before and after sending something to the controller,
                      04350 ; which is what an UltraDome/DeltaDome does, and the reply for a dome
                      04351 ; type query request (poll) is set to 0xF5. It should be noted that these
                      04352 ; two changes are the only diffrence between SpeedDome and UltraDome
                      04353 ; modes of operation. I.e. there are no "real" changes made to the
                      04354 ; TXB-S422's operation nor are there any additions made.
                      04355 ;
                      04356 ; Some time or other SW-7 will be used to terminate/unterminate the
                      04357 ; communications line from the controller. And SW-8 will be used to
                      04358 ; terminate/unterminate the communications line going to the controller.
                      04359 ; (Both of these are hardware changes that can not be monitored, or made
                      04360 ; from software.)
                      04361 ;
                      04362 ; Currently only SW-4 is unused.
                      04363 ;
                      04364 ; It should be noted that all bit positions are "live", i.e. they may be
                      04365 ; changed at any time "on the fly" and not just prior to power up time.
                      04366 ; With the Spectra it is difficult to change the switch positions with
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 100
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      04367 ; the Spectra installed in its back-box, however it is possiable to
                      04368 ; change the switch positions on an Esprit type unit. (Or if you have a
                      04369 ; special extension cable to use with a Spectra.)
                      04370 ;
                      04371 ;;\item In the "beTa" rev logic was changed to correctly interpert variable
                      04372 ; speed commands from an RC216.
                      04373 ;
                      04374 ; Also aux 4 becomes redircted to aux 1 for wiper motor control.
                      04375 ;
                      04376 ; And the "trial pattern run" command was honored. (Previously it was
                      04377 ; ignored.)
                      04378 ;
                      04379 ;;\item In rev "bEta" the tilt speed was fixed in RC216 mode. (Found a place
                      04380 ; where I used a "movf" instead of a "movwf" to save the tilt speed
                      04381 ; value.
                      04382 ;
                      04383 ; In the logic for identifying an RC216 type controller, speed 2 was not
                      04384 ; being decoded as a trigger.
                      04385 ;
                      04386 ; In an attempt to speed up tilt speeds in RC216 mode, I added an
                      04387 ; additional offset of 6 to the input value. (All speeds get incremented
```

```
                      04388 ; by 6 to get over the many slow starting tilt values. This makes it 12
                      04389 ; to force speeds further up the decode table.)
                      04390 ;
                      04391 ; Stopped clearing saved pan/tilt status (i.e. motion controls) on
                      04392 ; getting a "fast" type stop command. Now multiple fast/faster/fastest
                      04393 ; commands may be used.
                      04394 ;
                      04395 ; Started to clear both d_command1_M and d_command2_M before decoding
                      04396 ; commands. Now I hope to not get both right and left pan bits on at the
                      04397 ; same time.
                      04398 ;
                      04399 ; Fixed many little problems that turned up as I fixed other problems.
                      04400 ; (Some of these new ones were caused by the previous fixes, etc.)
                      04401 ;
                      04402 ; Finally gave up and completly redid the motion control logic.
                      04403 ;;\end{enumerate}
                      04404 ;;pe
                      04405
                      04406 ;;latex\subsectionindex{Part numbers}
                      04407 ;;ps
                      04408 ; The following Pelco part numbers apply to this project. Each time
                      04409 ; a software rev is made, the rev number on the IC51, PG51, BH51 and
                      04410 ; FW00 parts, must be updated.
                      04411 ;
                      04412 ; This listing must be updated EACH TIME before releasing the code.
                      04413 ;;pe
                      04414 ;;ts
                      04415 ; PA11-0006-00x0 Is the fully assembled board with many other things on it.
                      04416 ; IC51-0029-0100 The programmed IC that gets stuck on the board.
                      04417 ; IC01-1928-0876 PIC 16F876 chip that gets stuff written into.
                      04418 ; PG51-0042-0100 All combined binary hex files. There is only one .HEX
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 101
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      04419 ;               file on this project.
                      04420 ; BH51-0022-0100 Single binary hex file in .HEX format
                      04421 ; FW00-0150-0100 Source code for this project. Consists of four files:
                      04422 ;               The main source file, TXBS422.ASM
                      04423 ;               MicroChip's chip include file, P16C876.INC
                      04424 ; optional      A file of Sensormatic protocol definitions, SDEFS.INC
                      04425 ;               A file of Pelco protocol definitions, DPROTO.INC
                      04426 ;;te
                      04427 ;;ps
                      04428 ; To operate correctly the crystal running this chip must be cut for
                      04429 ; 11.0592 MHz
                      04430 ;;pe
                      04431 ;;ts
                      04432 ;       Instruction Memory Banks
                      04433 ;
                      04434 ;       Bank    Start     End
                      04435 ;         0     0x0000   0x07FF
                      04436 ;         1     0x0800   0x0FFF
                      04437 ;;te
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 102
$Header: d:/sears/RCS/txbs422.asm,v 1.121 2002-03-05 16:06:12-08 Hamilton Exp Hamilton $
Copyright by Pelco, 2001-2002, FW00-0150-0100 for U2 in the TXB-S422
LOC  OBJECT CODE     LINE SOURCE TEXT
  VALUE

                      04438         page
                      04439         end


MEMORY USAGE MAP ('X' = Used,  '-' = Unused)

0000 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0080 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
00C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0100 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0140 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0180 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
01C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0200 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0240 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0280 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
02C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0300 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0340 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
```

```
0380 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
03C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0400 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0440 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0480 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
04C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0500 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0540 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0580 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
05C0 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0600 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0640 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
0680 : XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXX----
2000 : -------X-------- ---------------- ---------------- ----------------

All other memory blocks unused.

Program Memory Words Used:  1725
Program Memory Words Free:  6467


Errors   :     0
Warnings :     0 reported,     0 suppressed
Messages :     0 reported,    28 suppressed

```

APPENDIX B

B TXBS422 Cross Reference Listing

```
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE  1

                        MPASM Cross Reference File

LABEL                           TYPE FILE NAME    SOURCE FILE REFERENCES
-----                           ---- ---------    ----------------------

BAD_TRY                          M   TXBS422.ASM    245*  756.
BAUD1200                         M   TXBS422.ASM    487*
BAUD19200                        M   TXBS422.ASM    491*
BAUD2400                         M   TXBS422.ASM    488*
BAUD4800                         M   TXBS422.ASM    489* 3522.
BAUD9600                         M   TXBS422.ASM    490*
BLANK                            M   TXBS422.ASM    246* 2028.
BYTE1                            M   TXBS422.ASM    396* 2095. 2103. 2126. 3291. 3322. 3348.
BYTE2                            M   TXBS422.ASM    397* 2107. 2111. 2127. 3292. 3326. 3338. 3349.
BYTE3                            M   TXBS422.ASM    398* 1832. 1863. 1866. 2120. 2128. 3293. 3329. 3356. 3365.
CLEAR_DISPLAY                    M   TXBS422.ASM    361* 1833. 1836. 3613.
CR                               M   TXBS422.ASM    247* 2397.
DATA_OUT                         M   TXBS422.ASM    327* 3023. 3032. 3034. 3038.
DBAUD_BASIC                      M   TXBS422.ASM    444* 3098. 3119. 3132.
DBAUD_REPEAT                     M   TXBS422.ASM    445* 3095. 3116. 3129.
DEBUG_MODE_ON                    M   TXBS422.ASM    326* 1957. 2858. 2927. 3017. 3237. 3666. 3749.
DEBUG_PERIOD                     M   TXBS422.ASM    451* 3029. 3042. 3044.
D_ALTUSE                         M   DPROTO.INC      21*
D_AUTO_SCAN_ALT                  M   DPROTO.INC      24*
D_CAMERA_OFF                     M   DPROTO.INC      27*
D_CAMERA_ON_ALT                  M   DPROTO.INC      26*
D_CLEAR_AUXILIARY                M   TXBS422.ASM   1753. 1762. 1779.
                                     DPROTO.INC      53*
D_CLEAR_PERSET                   M   DPROTO.INC      54*
D_CLEAR_SCREEN                   M   TXBS422.ASM   2039.
                                     DPROTO.INC      55*
D_END_PATTERN                    M   TXBS422.ASM   1476.
                                     DPROTO.INC      56*
D_EXTENDED                       M   DPROTO.INC      39*
D_FLIP                           M   TXBS422.ASM   1366.
                                     DPROTO.INC      57*
D_FOCUS_FAR                      M   DPROTO.INC      32*
D_FOCUS_NEAR                     M   DPROTO.INC      30*
D_GOTO_PRESET                    M   TXBS422.ASM   1368. 1409. 1523. 4163.
                                     DPROTO.INC      58*
D_HOME                           M   TXBS422.ASM   1407.
                                     DPROTO.INC      59*
D_IRIS_CLOSE                     M   DPROTO.INC      28*
D_IRIS_OPEN                      M   DPROTO.INC      29*
D_MANUAL_SCAN                    M   DPROTO.INC      25*
D_MENU                           M   TXBS422.ASM   1445.
                                     DPROTO.INC      60*
D_MOTION                         M   TXBS422.ASM   1571.
                                     DPROTO.INC      43*
D_PAN_LEFT                       M   DPROTO.INC      37*
D_PAN_MOTION                     M   DPROTO.INC      42*
D_PAN_RIGHT                      M   DPROTO.INC      38*
D_PAN_SPEED                      M   DPROTO.INC      47*
D_QUERY                          M   TXBS422.ASM   3589.
                                     DPROTO.INC      61*
D_RESET                          M   TXBS422.ASM   1392. 1531.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12        PAGE  2

                        MPASM Cross Reference File

LABEL                           TYPE FILE NAME    SOURCE FILE REFERENCES
-----                           ---- ---------    ----------------------

                                     DPROTO.INC      62*
D_RUN_PATTERN                    M   TXBS422.ASM   1486. 1496.
                                     DPROTO.INC      63*
D_SET_AUXILIARY                  M   TXBS422.ASM   1751. 1760. 1777.
                                     DPROTO.INC      64*
D_SET_PRESET                     M   TXBS422.ASM   1447. 1511. 4083.
                                     DPROTO.INC      65*
D_SPARE5                         M   DPROTO.INC      23*
D_SPARE6                         M   DPROTO.INC      22*
```

```
D_START_PATTERN                 M   TXBS422.ASM   1470.
                                    DPROTO.INC      66*
D_STOP                          M   DPROTO.INC      45*
D_SYNC                          M   TXBS422.ASM   3294.
                                    DPROTO.INC      50*
D_TILT_DOWN                     M   DPROTO.INC      35*
D_TILT_MOTION                   M   DPROTO.INC      41*
D_TILT_SPEED                    M   DPROTO.INC      48*
D_TILT_UP                       M   DPROTO.INC      36*
D_WRITE_CHAR                    M   TXBS422.ASM   3620.
                                    DPROTO.INC      67*
D_ZOOM_IN                       M   DPROTO.INC      34*
D_ZOOM_OUT                      M   DPROTO.INC      33*
EE_PRESET                       M   TXBS422.ASM    184* 2385. 2828.
EE_SET_MEM0                     M   TXBS422.ASM    185* 2339. 2368.
EE_SET_MEM1                     M   TXBS422.ASM    186* 2345. 2372.
EE_SET_MEM2                     M   TXBS422.ASM    187* 2351. 2376.
EE_SET_MEM3                     M   TXBS422.ASM    188* 2357. 2380.
ENABLE_INPUTS                   M   TXBS422.ASM    274* 3748.
ENABLE_OUTPUTS                  M   TXBS422.ASM    335* 2611. 3170. 3235. 3747.
FOCUS_FAR                       M   TXBS422.ASM    236* 1420. 1426. 1592.
FOCUS_NEAR                      M   TXBS422.ASM    234* 1421. 1425. 1591.
IGNORE_ADDRESS                  M   TXBS422.ASM    250* 3928.
INPUT_ERROR                     M   TXBS422.ASM    360* 2092. 2299. 2303. 2314. 2782. 3773. 3782. 3824. 3839. 3849. 3859. 3864.
                                                  3869. 3874. 3879. 3884. 3889. 3899.
INVERT_IO                       M   TXBS422.ASM    278* 2678. 2680.
IN_FAST                         M   TXBS422.ASM    399* 1151. 1286. 1381.
IN_MOVEMENT                     M   TXBS422.ASM    376* 1113. 1238. 1332. 1343. 1364. 1372. 1572. 1575.
IRIS_CLOSE                      M   TXBS422.ASM    238* 1398. 1414. 1602.
IRIS_OPEN                       M   TXBS422.ASM    237* 1399. 1413. 1601.
LAST_FAST                       M   TXBS422.ASM    422*  803.  815. 1331. 1361. 1365. 1577. 1586.
LF                              M   TXBS422.ASM    251* 2399.
LONG_TIME_OUT                   M   TXBS422.ASM    252*  749.
NODELAY                         M   TXBS422.ASM    311* 1548. 2582. 3171.
NORMAL_POLARITY                 M   TXBS422.ASM    358* 2677. 2679.
NO_MORE_PRESETS                 M   TXBS422.ASM    369*  740.  753. 2831. 2834.
PAN_DEFAULT                     M   TXBS422.ASM    210* 1287. 1559. 1608. 3389.
PAN_FAST                        M   TXBS422.ASM    213* 1336.
PAN_FASTER                      M   TXBS422.ASM    212* 1347.
PAN_FASTEST                     M   TXBS422.ASM    211* 1376.
PAN_LEFT                        M   TXBS422.ASM    229* 1279. 1283. 1558. 1698.
PAN_RIGHT                       M   TXBS422.ASM    228* 1278. 1284. 1557. 1694.
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE  3

                           MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

PELCO_SPEED_DOME                 M   TXBS422.ASM    253* 1549.
PELCO_ULTRA_DOME                 M   TXBS422.ASM    254* 1547.
PERMIT64                         M   TXBS422.ASM    325* 3924.
PROGRAM_PAGE0                    M   TXBS422.ASM    155*
PROGRAM_PAGE1                    M   TXBS422.ASM    156*
RAM0                             M   TXBS422.ASM    166*  547.
RAM1                             M   TXBS422.ASM    167*  624.
RAM2                             M   TXBS422.ASM    168*  629.
RAM3                             M   TXBS422.ASM    169*  634.
RC216_MODE                       M   TXBS422.ASM    364* 1049. 1110. 1175. 2206.
RESERVED_BIT                     M   TXBS422.ASM    359* 2676.
RESET_VECTOR                     M   TXBS422.ASM    154*  645.
SPECTRA_IN                       M   TXBS422.ASM    289* 2637. 2641. 2655. 2715. 2720. 2726. 2735. 2745. 2756. 3570. 3574.
SPECTRA_OUT                      M   TXBS422.ASM    287* 3089. 3108. 3110. 3128.
SPEEDCONTROL0                    M   TXBS422.ASM    299* 2256. 2262.
SPEEDCONTROL1                    M   TXBS422.ASM    300* 2253.
SPEEDMASK                        M   TXBS422.ASM    307*
STEP1MENU                        M   TXBS422.ASM    425*  826.  829. 1397.
STEP1RESET                       M   TXBS422.ASM    419*  844.  847. 1451.
STEP2MENU                        M   TXBS422.ASM    428*  840. 1441. 1444.
STEP2RESET                       M   TXBS422.ASM    420*  853. 1388. 1391. 1403. 1406.
S_ACK                            M   SDEFS.INC       43*
S_ALL_STOP                       M   SDEFS.INC       40*
S_BOUNDARY0CROSSED               M   TXBS422.ASM   1877.
                                     SDEFS.INC      118*
S_BOUNDARY1CROSSED               M   SDEFS.INC      119*
S_BOUNDARY2CROSSED               M   SDEFS.INC      120*
S_BOUNDARY3CROSSED               M   SDEFS.INC      121*
S_BOUNDARY_CONFUSION             M   SDEFS.INC      122*
S_DEFINE1                        M   SDEFS.INC       58*
S_DEFINE2                        M   SDEFS.INC       59*
S_DEFINE3                        M   SDEFS.INC       60*
S_DEFINE_BOUNDARY                M   SDEFS.INC       50*
```

```
S_DOME_ALARM0                    M   SDEFS.INC      128*
S_DOME_ALARM1                    M   SDEFS.INC      129*
S_DOME_ALARM2                    M   SDEFS.INC      130*
S_DOME_ALARM3                    M   SDEFS.INC      131*
S_DOME_TYPE_IS                   M   SDEFS.INC      117*
S_DUMP_DOME_MEMORY               M   SDEFS.INC       63*
S_FAST                           M   SDEFS.INC       26*
S_FASTER                         M   SDEFS.INC       47*
S_FASTER_STOP                    M   SDEFS.INC       49*
S_FASTEST                        M   TXBS422.ASM    813.
                                     SDEFS.INC       28*
S_FAST_STOP                      M   SDEFS.INC       30*
S_FOCUS_FAR                      M   TXBS422.ASM    849.
                                     SDEFS.INC       21*
S_FOCUS_NEAR                     M   TXBS422.ASM    831.
                                     SDEFS.INC       20*
S_FOCUS_STOP                     M   SDEFS.INC       22*
S_GET_ALARMS                     M   SDEFS.INC       42*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  4

                            MPASM Cross Reference File

LABEL                              TYPE FILE NAME    SOURCE FILE REFERENCES
-----                              ---- ---------    ----------------------

S_GET_DOME_TYPE                  M   SDEFS.INC       41*
S_GET_POSITION                   M   SDEFS.INC       64*
S_GOTO1                          M   SDEFS.INC       80*
S_GOTO2                          M   SDEFS.INC       81*
S_GOTO3                          M   SDEFS.INC       82*
S_GOTO4                          M   SDEFS.INC       83*
S_GOTO5                          M   SDEFS.INC       89*
S_GOTO6                          M   SDEFS.INC       90*
S_GOTO7                          M   SDEFS.INC       91*
S_GOTO_PAT1                      M   SDEFS.INC       70*
S_GOTO_PAT2                      M   SDEFS.INC       71*
S_GOTO_PAT3                      M   SDEFS.INC       72*
S_GOTO_PAT4                      M   SDEFS.INC       73*
S_GOTO_POSITION                  M   TXBS422.ASM   3788.
                                     SDEFS.INC       65*
S_GOTO_POSITION_SIZE             M   TXBS422.ASM    262* 3792.
S_IRIS_CLOSE                     M   SDEFS.INC       35*
S_IRIS_OPEN                      M   SDEFS.INC       32*
S_IRIS_STOP                      M   SDEFS.INC       38*
S_MARK1                          M   SDEFS.INC       66*
S_MARK2                          M   SDEFS.INC       67*
S_MARK3                          M   SDEFS.INC       68*
S_MARK4                          M   SDEFS.INC       69*
S_MARK5                          M   SDEFS.INC       86*
S_MARK6                          M   SDEFS.INC       87*
S_MARK7                          M   SDEFS.INC       88*
S_MARK_BOUNDARY                  M   SDEFS.INC       53*
S_NEW_PATTERN                    M   SDEFS.INC       61*
S_ON_AIR                         M   SDEFS.INC       54*
S_ON_AIR_RESET                   M   SDEFS.INC       57*
S_OUTPUT0                        M   TXBS422.ASM    993.  998.
                                     SDEFS.INC      107*
S_OUTPUT1                        M   SDEFS.INC      108*
S_OUTPUT2                        M   SDEFS.INC      109*
S_OUTPUT3                        M   SDEFS.INC      110*
S_OUTPUT4                        M   SDEFS.INC      111*
S_PAN_LEFT                       M   SDEFS.INC       12*
S_PAN_RIGHT                      M   SDEFS.INC       14*
S_PAN_STOP                       M   SDEFS.INC       16*
S_PATTERN_DONE                   M   SDEFS.INC      124*
S_PATTERN_END                    M   SDEFS.INC       84*
S_POWERED_UP                     M   TXBS422.ASM   3751.
                                     SDEFS.INC      126*
S_PROP_DOWN                      M   TXBS422.ASM   1678.
                                     SDEFS.INC      100*
S_PROP_LEFT                      M   TXBS422.ASM   1663.
                                     SDEFS.INC       97*
S_PROP_RIGHT                     M   TXBS422.ASM   1668.
                                     SDEFS.INC       98*
S_PROP_SPEED                     M   TXBS422.ASM    976. 2885. 3815. 3819. 3844.
                                     SDEFS.INC       95*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  5

                            MPASM Cross Reference File

LABEL                              TYPE FILE NAME    SOURCE FILE REFERENCES
-----                              ---- ---------    ----------------------
```

```
S_PROP_SPEED_SIZE                 M   TXBS422.ASM    263*
S_PROP_UP                         M   TXBS422.ASM   1673.
                                      SDEFS.INC       99*
S_PROTO_LENGTH                    M   TXBS422.ASM    264*  758. 3398.
S_RESET                           M   TXBS422.ASM    981.
                                      SDEFS.INC      102*
S_RUN1                            M   SDEFS.INC       74*
S_RUN2                            M   SDEFS.INC       75*
S_RUN3                            M   SDEFS.INC       76*
S_RUN_NEW                         M   SDEFS.INC       77*
S_SOFTWARE_VERSION                M   TXBS422.ASM    987. 4122.
                                      SDEFS.INC      106*
S_TERM_PAT                        M   TXBS422.ASM   1003.
                                      SDEFS.INC      112*
S_TILT_DOWN                       M   SDEFS.INC       18*
S_TILT_STOP                       M   SDEFS.INC       19*
S_TILT_UP                         M   SDEFS.INC       17*
S_UNKNOWN80                       M   SDEFS.INC       11*
S_UNKNOWN98                       M   SDEFS.INC       45*
S_UNKNOWN99                       M   SDEFS.INC       46*
S_UNKNOWN_C1                      M   TXBS422.ASM    971. 2911. 3806. 3810. 3894.
                                      SDEFS.INC       94*
S_UNKNOWN_C1_SIZE                 M   TXBS422.ASM    265*
S_UNKNOWN_C4                      M   SDEFS.INC      101*
S_VARIABLE_SPEED                  M   TXBS422.ASM    966. 2892. 3797. 3854.
                                      SDEFS.INC       92*
S_VARIABLE_SPEED_SIZE             M   TXBS422.ASM    266* 3801.
S_ZOOM_IN                         M   SDEFS.INC       23*
S_ZOOM_OUT                        M   SDEFS.INC       24*
S_ZOOM_STOP                       M   SDEFS.INC       25*
TILT_DEFAULT                      M   TXBS422.ASM    216* 1152. 1566. 1610. 3392.
TILT_DOWN                         M   TXBS422.ASM    231* 1144. 1148. 1565. 1682.
TILT_FAST                         M   TXBS422.ASM    219* 1338.
TILT_FASTER                       M   TXBS422.ASM    218* 1349.
TILT_FASTEST                      M   TXBS422.ASM    217* 1378.
TILT_UP                           M   TXBS422.ASM    230* 1143. 1149. 1564. 1690.
UART_IN                           M   TXBS422.ASM    340*
UART_OUT                          M   TXBS422.ASM    339*
ZOOM_IN                           M   TXBS422.ASM    232* 1437. 1452. 1596.
ZOOM_OUT                          M   TXBS422.ASM    233* 1436. 1453. 1597.
__16F876                          V   TXBS422.ASM      0*  129.
                                      P16F876.INC      40.
_boden_off                        C   P16F876.INC    358*
_boden_on                         C   TXBS422.ASM    120.
                                      P16F876.INC    357*
_cp_all                           C   TXBS422.ASM    120.
                                      P16F876.INC    340*
_cp_half                          C   P16F876.INC    341*
_cp_off                           C   P16F876.INC    343*
_cp_upper_256                     C   P16F876.INC    342*
_cpd_off                          C   P16F876.INC    352*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  6

                           MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

_cpd_on                           C   P16F876.INC    351*
_debug_off                        C   P16F876.INC    346*
_debug_on                         C   P16F876.INC    345*
_hs_osc                           C   TXBS422.ASM    120.
                                      P16F876.INC    368*
_lp_osc                           C   P16F876.INC    366*
_lvp_off                          C   P16F876.INC    355*
_lvp_on                           C   P16F876.INC    354*
_pwrte_off                        C   P16F876.INC    360*
_pwrte_on                         C   TXBS422.ASM    120.
                                      P16F876.INC    361*
_rc_osc                           C   P16F876.INC    369*
_wdt_off                          C   TXBS422.ASM    120.
                                      P16F876.INC    364*
_wdt_on                           C   P16F876.INC    363*
_wrt_enable_off                   C   TXBS422.ASM    120.
                                      P16F876.INC    349*
_wrt_enable_on                    C   P16F876.INC    348*
_xt_osc                           C   P16F876.INC    367*
ackdonecommand                    A   TXBS422.ASM   1333. 1344. 1627* 1864. 1883.
ackdt                             C   P16F876.INC    263*
acken                             C   P16F876.INC    264*
ackstat                           C   P16F876.INC    262*
```

```
adcon0                            C   P16F876.INC     85*
adcon1                            C   TXBS422.ASM   3412.
                                      P16F876.INC    101*
adcs0                             C   P16F876.INC    217*
adcs1                             C   P16F876.INC    216*
adden                             C   P16F876.INC    199*
adfm                              C   P16F876.INC    308*
adie                              C   P16F876.INC    239*
adif                              C   P16F876.INC    135*
adon                              C   P16F876.INC    224*
adresh                            C   P16F876.INC     84*
adresl                            C   P16F876.INC    100*
again                             A   TXBS422.ASM    739*  797.  907.  910.  911. 1630. 3774. 3779. 3783. 3825. 3830. 3840. 3850.
                                                    3860. 3865. 3870. 3875. 3880. 3885. 3890. 3900. 3930. 3936. 3940. 4256.
alldone                           A   TXBS422.ASM   1680. 1861* 4097.
allin                             A   TXBS422.ASM   3836. 3923*
arrowout_3                        A   TXBS422.ASM   1952* 2979. 3200. 4056. 4118. 4209.
backtonormal                      A   TXBS422.ASM    741.  744.  750.  755*
badcount_M                        C   TXBS422.ASM    549*  757. 2786.
bclie                             C   P16F876.INC    250*
bclif                             C   P16F876.INC    146*
bf                                C   P16F876.INC    291*
bin2hex_0                         A   TXBS422.ASM   1978* 2964. 3618.
bit0                              M   TXBS422.ASM    498*  844.  847. 1420. 1426. 1451. 1592. 1752. 2095. 2103. 2126. 2256. 2262.
                                                    2677. 2679. 3031. 3033. 3107. 3109. 3291. 3322. 3348.
bit1                              M   TXBS422.ASM    499*  749.  853. 1278. 1284. 1388. 1391. 1399. 1403. 1406. 1413. 1557. 1601.
                                                    1694. 1761. 2107. 2111. 2127. 2253. 2676. 3292. 3326. 3338. 3349. 3748.
bit2                              M   TXBS422.ASM    500*  803.  815. 1279. 1283. 1331. 1361. 1365. 1398. 1414. 1548. 1558. 1577.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  7

                          MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

                                                    1586. 1602. 1698. 1832. 1863. 1866. 2120. 2128. 2582. 2678. 2680. 3171. 3293.
                                                    3329. 3356. 3365.
bit3                               M   TXBS422.ASM    501*  826.  829. 1143. 1149. 1151. 1286. 1381. 1397. 1564. 1690. 1778. 2092.
                                                    2299. 2303. 2314. 2782. 3089. 3108. 3110. 3128. 3773. 3782. 3824. 3839. 3849.
                                                    3859. 3864. 3869. 3874. 3879. 3884. 3889. 3899.
bit4                               M   TXBS422.ASM    502*  840. 1144. 1148. 1441. 1444. 1565. 1682. 1833. 1836. 3613. 3924.
bit5                               M   TXBS422.ASM    503* 1049. 1110. 1175. 1437. 1452. 1596. 1957. 2206. 2611. 2637. 2641. 2655.
                                                    2715. 2720. 2726. 2735. 2745. 2756. 2858. 2927. 3017. 3170. 3235. 3237. 3570.
                                                    3574. 3666. 3747. 3749.
bit6                               M   TXBS422.ASM    504*  740.  753. 1436. 1453. 1597. 2831. 2834. 3023. 3032. 3034. 3038.
bit7                               M   TXBS422.ASM    505*  743. 1113. 1238. 1332. 1343. 1364. 1372. 1421. 1425. 1572. 1575. 1591.
bitdelay_0                         A   TXBS422.ASM   2015* 3030. 3043. 3045.
bitwait                            A   TXBS422.ASM   2017* 2018.
blank_2                            A   TXBS422.ASM   1966. 2027* 2869. 2870. 2936. 2939. 2969.
blankit_3                          A   TXBS422.ASM   1845. 2037*
boundarymark                       A   TXBS422.ASM    916. 1915*
boundarystart                      A   TXBS422.ASM    915. 1910*
brgh                               C   P16F876.INC    301*
build_command_0                    A   TXBS422.ASM   1114. 1239. 1382. 1430. 1587. 2061*
byte3again                         A   TXBS422.ASM   3354* 3363.
bytebits_1                         C   TXBS422.ASM    550* 3022. 3036. 3088. 3112.
byteread_1                         A   TXBS422.ASM   2086* 3299. 3302. 3305. 3308. 3311. 3314. 3317. 3320. 3337. 3355.
c                                  C   TXBS422.ASM    796.  994.  999. 1054. 1058. 1062. 1066. 1070. 1074. 1078. 1135. 1180. 1184.
                                                    1188. 1192. 1196. 1200. 1204. 1266. 1990. 2001. 2265. 2267. 2274. 2276. 2278.
                                                    2284. 2433. 2439. 2596. 2755. 2757. 2842. 3225. 3243. 3344. 3362.
                                      P16F876.INC    120*
ccp1con                            C   TXBS422.ASM   3505.
                                      P16F876.INC     77*
ccp1ie                             C   P16F876.INC    243*
ccp1if                             C   P16F876.INC    139*
ccp1m0                             C   P16F876.INC    188*
ccp1m1                             C   P16F876.INC    187*
ccp1m2                             C   P16F876.INC    186*
ccp1m3                             C   P16F876.INC    185*
ccp1x                              C   P16F876.INC    183*
ccp1y                              C   P16F876.INC    184*
ccp2con                            C   TXBS422.ASM   3506.
                                      P16F876.INC     83*
ccp2ie                             C   P16F876.INC    251*
ccp2if                             C   P16F876.INC    147*
ccp2m0                             C   P16F876.INC    212*
ccp2m1                             C   P16F876.INC    211*
ccp2m2                             C   P16F876.INC    210*
ccp2m3                             C   P16F876.INC    209*
ccp2x                              C   P16F876.INC    207*
ccp2y                              C   P16F876.INC    208*
ccpr1h                             C   P16F876.INC     76*
```

```
ccpr1l                            C   P16F876.INC     75*
ccpr2h                            C   P16F876.INC     82*
ccpr2l                            C   P16F876.INC     81*
check4rc216_0                     A   TXBS422.ASM   1686. 1702. 2184*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  8

                            MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

checkC0                           A   TXBS422.ASM   3790. 3795*
checkC1                           A   TXBS422.ASM   3799. 3804*
checkC3                           A   TXBS422.ASM   3808. 3813*
checkrate_0                       A   TXBS422.ASM   1131. 1262. 2251*
checksumC0                        A   TXBS422.ASM   3856. 3915*
checksumC1                        A   TXBS422.ASM   3896. 3907*
checksumC3                        A   TXBS422.ASM   3846. 3917*
checksum_M                        C   TXBS422.ASM    551* 2993. 2997. 3072. 4119. 4210.
chs0                              C   P16F876.INC    220*
chs1                              C   P16F876.INC    219*
chs2                              C   P16F876.INC    218*
ck4more                           A   TXBS422.ASM   2101. 2104. 2112*
cke                               C   P16F876.INC    273*
ckioerrs_0                        A   TXBS422.ASM   2091. 2298* 2780. 3604. 3771.
ckp                               C   P16F876.INC    175*
clearcandexit                     A   TXBS422.ASM   2431. 2438*
clearee                           A   TXBS422.ASM   2343. 2349. 2355. 2361. 2367*
cleareememory_1                   A   TXBS422.ASM   2338* 3605.
clrflagsandexit                   A   TXBS422.ASM   3347* 3366.
continuationtimer_M               C   TXBS422.ASM    552*  748.  749. 4095.
cormore                           A   TXBS422.ASM    862.  964*
cren                              C   TXBS422.ASM    697. 2308. 2309. 2319. 2320.
                                      P16F876.INC    198*
crlf_2                            A   TXBS422.ASM   2396* 2861. 2866. 2930. 2935. 3750.
csrc                              C   P16F876.INC    295*
d                                 C   P16F876.INC    274*
d1                                A   TXBS422.ASM   2468* 2470.
d2                                A   TXBS422.ASM   2467* 2473.
d_a                               C   P16F876.INC    278*
d_address_M                       C   TXBS422.ASM    557* 2500. 2942. 3297. 3303. 3587.
d_checksum_M                      C   TXBS422.ASM    563* 2505. 2952. 3318.
d_command1_M                      C   TXBS422.ASM    558*  793. 1837. 1847. 2038. 2063. 2487. 2501. 2944. 3306. 3588. 3619. 3755.
                                                    4082. 4162.
d_command2_M                      C   TXBS422.ASM    559*  794. 1393. 1532. 1754. 1763. 1780. 1830. 1839. 1849. 2040. 2065. 2502.
                                                    2946. 3309. 3590. 3621. 3756. 4084. 4164.
d_pan_speed_M                     C   TXBS422.ASM    561*  857. 1841. 1851. 2041. 2067. 2488. 2503. 2809. 2948. 3312. 3591. 3623.
                                                    4085. 4165.
d_sync_M                          C   TXBS422.ASM    556* 2940. 3295. 3300.
d_tilt_speed_M                    C   TXBS422.ASM    562*  858. 1367. 1408. 1446. 1463. 1465. 1468. 1475. 1483. 1484. 1495. 1504.
                                                    1505. 1506. 1507. 1508. 1509. 1510. 1516. 1517. 1518. 1519. 1520. 1521. 1522.
                                                    1843. 1853. 2042. 2069. 2489. 2504. 2810. 2950. 3315. 3592. 4029. 4167.
data_address                      C   P16F876.INC    279*
dc                                C   P16F876.INC    119*
decodeinput_M                     A   TXBS422.ASM    792* 1867. 3944.
delaynohang_0                     A   TXBS422.ASM   2425* 2595. 3224. 3242. 3342. 3360.
delayspectra_0                    A   TXBS422.ASM   2449* 2719. 2725. 2733. 2734. 2744. 2753. 2760. 3573.
delayv_0                          A   TXBS422.ASM   1536. 2464* 2586. 2606. 3175. 3583. 3585.
displayaddress                    A   TXBS422.ASM   3679. 3733*
do216pan                          A   TXBS422.ASM   1050. 1129*
do216tilt                         A   TXBS422.ASM   1176. 1260*
doaux_3                           A   TXBS422.ASM   1756. 1765. 1783. 2486*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE  9

                            MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

docheckSum_0                      A   TXBS422.ASM   2499* 3298.
doflip                            A   TXBS422.ASM   1364* 1373.
donconvert                        A   TXBS422.ASM   2002. 2006*
doneclearing                      A   TXBS422.ASM   2363* 2389.
donecommand                       A   TXBS422.ASM   1629* 1784. 1934. 4139.
dontprint                         A   TXBS422.ASM   2928. 2954*
dontprintin                       A   TXBS422.ASM   2859. 2879. 2887. 2894. 2913. 2917*
eeadr                             C   TXBS422.ASM   2521. 2556.
                                      P16F876.INC    104*
eeadrh                            C   TXBS422.ASM   2522. 2557.
                                      P16F876.INC    106*
eecon1                            C   TXBS422.ASM   2524. 2525. 2552. 2563. 2564. 2569. 2570.
                                      P16F876.INC    108*
```

```
eecon2                            C   TXBS422.ASM   2566. 2568.
                                      P16F876.INC    109*
eedata                            C   TXBS422.ASM   2527. 2561.
                                      P16F876.INC    103*
eedath                            C   P16F876.INC    105*
eeie                              C   P16F876.INC    249*
eeif                              C   P16F876.INC    145*
eepgd                             C   TXBS422.ASM   2524. 2563.
                                      P16F876.INC    316*
eeprombusy                        A   TXBS422.ASM   2551* 2553.
eeread_0                          A   TXBS422.ASM   2340. 2346. 2352. 2358. 2519* 2830.
eewrite_0                         A   TXBS422.ASM   2371. 2375. 2379. 2383. 2388. 2546* 2836.
eewritehere_0                     C   TXBS422.ASM    565* 2369. 2373. 2377. 2381. 2386. 2548. 2829.
endout_1                          A   TXBS422.ASM   2581* 2982. 2999. 3207. 4086.
exitnohang                        A   TXBS422.ASM   2434* 2440.
extratimeout_0                    C   TXBS422.ASM    566* 3096. 3104. 3117. 3125. 3130. 3138.
f                                 C   P16F876.INC     51*
fast                              A   TXBS422.ASM    895. 1330*
faster                            A   TXBS422.ASM    912. 1342*
fastest                           A   TXBS422.ASM    896. 1360*
ferr                              C   TXBS422.ASM   2311.
                                      P16F876.INC    200*
findspectraedge_0                 A   TXBS422.ASM   2633* 2714. 2724. 2743.
fixed_pan                         A   TXBS422.ASM   1280. 1285*
fixed_tilt                        A   TXBS422.ASM   1145. 1150*
flipck                            A   TXBS422.ASM   2786*
flipinput_0                       A   TXBS422.ASM   2674* 2789.
focusfar                          A   TXBS422.ASM    889. 1424*
focusnear                         A   TXBS422.ASM    887. 1419*
frameok                           A   TXBS422.ASM   2312. 2321*
fromspectra_1                     A   TXBS422.ASM   2713* 3594.
fsr                               C   P16F876.INC     59*
gcen                              C   P16F876.INC    261*
getalarms                         A   TXBS422.ASM    905. 1929*
getbyte2                          A   TXBS422.ASM   3327. 3336* 3345.
getbyte3                          A   TXBS422.ASM   3330. 3339. 3352*
getchecksum                       A   TXBS422.ASM   3793. 3802. 3811. 3820*
getdometype                       A   TXBS422.ASM    904. 1546*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 10

                           MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

getinput                          A   TXBS422.ASM    760. 3767*
getposition                       A   TXBS422.ASM    927. 4018*
getsensorbyte_1                   A   TXBS422.ASM   2776* 3772. 3781. 3823. 3838. 3848. 3858. 3863. 3868. 3873. 3878. 3883. 3888.
                                                    3898.
gie                               C   P16F876.INC    124*
giveupthistry                     A   TXBS422.ASM   3226. 3232*
giveupthistrynow                  A   TXBS422.ASM   3244. 3250*
go                                C   P16F876.INC    221*
go_done                           C   P16F876.INC    223*
goback                            A   TXBS422.ASM   2784*
gotfirst                          A   TXBS422.ASM   1991. 1995*
gotoposition                      A   TXBS422.ASM    928. 4160*
hexlower_0                        C   TXBS422.ASM    567* 1979. 1984. 1996. 1998. 2000. 2005. 2967. 3736.
hexupper_0                        C   TXBS422.ASM    568* 1980. 1981. 1983. 1985. 1987. 1989. 1994. 2965. 3734.
i2c_data                          C   P16F876.INC    275*
i2c_read                          C   P16F876.INC    285*
i2c_start                         C   P16F876.INC    283*
i2c_stop                          C   P16F876.INC    281*
indf                              C   P16F876.INC     55*
init_M                            A   TXBS422.ASM    646. 1538. 3378*
inlabel_3                         A   TXBS422.ASM   2808* 3630. 3632. 3634. 3636. 3638. 3640. 3642. 3644. 3646. 3653. 3655. 3657.
                                                    3659. 3670. 3672. 3674. 3676. 3678. 3685. 3687. 3689. 3691. 3693. 3735. 3737.
intcon                            C   TXBS422.ASM   3475.
                                      P16F876.INC     65*
inte                              C   P16F876.INC    127*
intedg                            C   P16F876.INC    229*
intf                              C   P16F876.INC    130*
irisclose                         A   TXBS422.ASM    900. 1402*
irisopen                          A   TXBS422.ASM    899. 1387*
irp                               C   P16F876.INC    113*
iscleared                         A   TXBS422.ASM   1834. 1859*
isfocus                           A   TXBS422.ASM    833.  839*
lastchecksum_M                    C   TXBS422.ASM    569* 3538. 3769. 3833. 3922. 3938.
lastnotfast                       A   TXBS422.ASM    804.  825*
lastnotmenu1                      A   TXBS422.ASM    827.  843*
manycount_5                       C   TXBS422.ASM    570* 3056. 3060.
mark1                             A   TXBS422.ASM    931. 1510*
```

```
mark2                             A   TXBS422.ASM    932. 1509*
mark3                             A   TXBS422.ASM    933. 1508*
mark4                             A   TXBS422.ASM    934. 1507*
mark5                             A   TXBS422.ASM    952. 1506*
mark6                             A   TXBS422.ASM    953. 1505*
mark7                             A   TXBS422.ASM    954. 1504*
memorydump                        A   TXBS422.ASM    926. 4208*
messcount_4                       C   TXBS422.ASM    571* 2862. 2863. 2867. 2931. 2932. 2937.
minorstop                         A   TXBS422.ASM   1576* 1593. 1598. 1603.
mkmode1_M                         C   TXBS422.ASM    572*  799.  803.  815.  826.  829.  840.  844.  847.  853. 1331. 1361. 1365.
                                                    1388. 1391. 1397. 1403. 1406. 1441. 1444. 1451. 1577. 1586. 3383.
mode1_M                           C   TXBS422.ASM    574*  740.  753. 1049. 1110. 1113. 1175. 1238. 1332. 1343. 1364. 1372. 1572.
                                                    1575. 1833. 1836. 2092. 2206. 2299. 2303. 2314. 2675. 2676. 2677. 2679. 2831.
                                                    2834. 3381. 3613. 3773. 3782. 3824. 3839. 3849. 3859. 3864. 3869. 3874. 3879.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 11

                          MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

                                                    3884. 3889. 3899.
mode2_M                           C   TXBS422.ASM    575* 1151. 1286. 1381. 1832. 1863. 1866. 2095. 2103. 2107. 2111. 2120. 2126.
                                                    2127. 2128. 3291. 3292. 3293. 3322. 3326. 3329. 3338. 3348. 3349. 3356. 3365.
                                                    3382.
nextaddressbit                    A   TXBS422.ASM   2752* 2762.
nextdebugbitout                   A   TXBS422.ASM   3028* 3037.
nextsync                          A   TXBS422.ASM   2732* 2739.
noarrow                           A   TXBS422.ASM   1958. 1967*
nodebugprint                      A   TXBS422.ASM   3018. 3046*
nodelay1                          A   TXBS422.ASM   3172. 3176*
nodelay2                          A   TXBS422.ASM   2583. 2589*
nomorewaiting                     A   TXBS422.ASM   2587. 2610*
nooverrun                         A   TXBS422.ASM   2301. 2310*
nopresetsnow                      A   TXBS422.ASM   2832. 2837*
normalirisclose                   A   TXBS422.ASM   1404. 1412*
normalirisopen                    A   TXBS422.ASM   1389. 1396*
normalzoomout                     A   TXBS422.ASM   1442. 1450*
not_a                             C   P16F876.INC    276*
not_address                       C   P16F876.INC    277*
not_bo                            C   P16F876.INC    256*
not_bor                           C   P16F876.INC    257*
not_done                          C   P16F876.INC    222*
not_pd                            C   P16F876.INC    117*
not_por                           C   P16F876.INC    255*
not_rbpu                          C   P16F876.INC    228*
not_rc8                           C   P16F876.INC    195*
not_t1sync                        C   P16F876.INC    154*
not_to                            C   P16F876.INC    116*
not_tx8                           C   P16F876.INC    297*
not_w                             C   P16F876.INC    286*
not_write                         C   P16F876.INC    287*
notaflip                          A   TXBS422.ASM   1362. 1371*
notdebugmode                      A   TXBS422.ASM   3667. 3681*
notexist1                         M   TXBS422.ASM    290*
notexist2                         M   TXBS422.ASM    291*
nothinghere                       A   TXBS422.ASM   2088. 2093. 2116. 2124. 2129*
notlong                           A   TXBS422.ASM   3817. 3822*
notmorethan33                     A   TXBS422.ASM   2843. 2847*
notmultikey                       A   TXBS422.ASM    801.  816.  837.  841.  845.  851.  855*
notrc216speed                     A   TXBS422.ASM   2203. 2207*
oerr                              C   TXBS422.ASM   2300.
                                      P16F876.INC    201*
onair                             A   TXBS422.ASM    917. 1876*
onairreset                        A   TXBS422.ASM    919. 1881*
onehalfvalue                      A   TXBS422.ASM   2263. 2283*
option_reg                        C   TXBS422.ASM   3469.
                                      P16F876.INC     87*
output                            A   TXBS422.ASM   1000. 1734*
overlong1_0                       C   TXBS422.ASM    576* 2451. 2453. 2636. 2646. 2649. 2660. 2663.
overlong2_0                       C   TXBS422.ASM    577* 2635. 2650. 2664. 2730. 2738. 2750. 2761.
p                                 C   P16F876.INC    280*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 12

                          MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

panleft                           A   TXBS422.ASM    880. 1282*
panright                          A   TXBS422.ASM    881. 1277*
pattern1                          A   TXBS422.ASM    921. 1466*
```

```
pattern2                          A   TXBS422.ASM    922. 1464*
pattern3                          A   TXBS422.ASM    923. 1462*
patternaccept                     A   TXBS422.ASM    924. 1882*
patternend                        A   TXBS422.ASM    951. 1473*
patternwas_M                      C   TXBS422.ASM    580* 1469. 1474. 1494.
pcfg0                             C   P16F876.INC    312*
pcfg1                             C   P16F876.INC    311*
pcfg2                             C   P16F876.INC    310*
pcfg3                             C   P16F876.INC    309*
pcl                               C   TXBS422.ASM    874.
                                      P16F876.INC     57*
pclath                            C   P16F876.INC     64*
pcon                              C   TXBS422.ASM   3472.
                                      P16F876.INC     93*
peie                              C   P16F876.INC    125*
pen                               C   P16F876.INC    266*
pie1                              C   TXBS422.ASM   3470.
                                      P16F876.INC     91*
pie2                              C   TXBS422.ASM   3471.
                                      P16F876.INC     92*
pir1                              C   TXBS422.ASM   2087. 2113. 2777. 3489.
                                      P16F876.INC     66*
pir2                              C   TXBS422.ASM   3490.
                                      P16F876.INC     67*
porta                             C   TXBS422.ASM   2637. 2641. 2655. 2678. 2680. 2715. 2720. 2726. 2735. 2745. 2756. 3089. 3108.
                                                     3110. 3128. 3384. 3570. 3574. 3748.
                                      P16F876.INC     60*
portb                             C   TXBS422.ASM   1548. 1957. 2253. 2256. 2262. 2582. 2858. 2927. 3017. 3023. 3032. 3034. 3038.
                                                     3171. 3237. 3385. 3614. 3666. 3749. 3924.
                                      P16F876.INC     61*
portc                             C   TXBS422.ASM   2611. 3170. 3235. 3386. 3747.
                                      P16F876.INC     62*
pr2                               C   P16F876.INC     95*
presetincrement_1                 A   TXBS422.ASM   2827* 4028.
printsensorin_4                   A   TXBS422.ASM   2857* 3942.
printspectraout_4                 A   TXBS422.ASM   1534. 1758. 1766. 1862. 2926*
propleft                          A   TXBS422.ASM   1665. 1697*
proppan                           A   TXBS422.ASM   1695. 1699*
propright                         A   TXBS422.ASM   1670. 1693*
propspeed                         A   TXBS422.ASM    978. 1618*
proptilt                          A   TXBS422.ASM   1683* 1691.
propup                            A   TXBS422.ASM   1675. 1689*
ps0                               C   P16F876.INC    235*
ps1                               C   P16F876.INC    234*
ps2                               C   P16F876.INC    233*
psa                               C   P16F876.INC    232*
pspeed1                           A   TXBS422.ASM   1055. 1103*
pspeed2                           A   TXBS422.ASM   1059. 1100*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 13

                            MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

pspeed3                           A   TXBS422.ASM   1063. 1097*
pspeed4                           A   TXBS422.ASM   1067. 1094*
pspeed5                           A   TXBS422.ASM   1071. 1091*
pspeed6                           A   TXBS422.ASM   1075. 1088*
pspeed7                           A   TXBS422.ASM   1079. 1085*
pspeed8                           A   TXBS422.ASM   1082* 1136.
r                                 C   P16F876.INC    284*
r_w                               C   P16F876.INC    288*
rbie                              C   P16F876.INC    128*
rbif                              C   P16F876.INC    131*
rc8_9                             C   P16F876.INC    196*
rc9                               C   P16F876.INC    194*
rcd8                              C   P16F876.INC    203*
rcen                              C   P16F876.INC    265*
rcie                              C   P16F876.INC    240*
rcif                              C   TXBS422.ASM   2087. 2113. 2777.
                                      P16F876.INC    136*
rcreg                             C   TXBS422.ASM   2306. 2307. 2317. 2318. 2321.
                                      P16F876.INC     80*
rcsta                             C   TXBS422.ASM    696.  697. 2300. 2304. 2308. 2309. 2311. 2315. 2319. 2320. 3537.
                                      P16F876.INC     78*
rd                                C   TXBS422.ASM   2525.
                                      P16F876.INC    320*
read_write                        C   P16F876.INC    289*
readsomemore                      A   TXBS422.ASM   2090* 2114.
reload2400                        A   TXBS422.ASM   3097* 3105.
reload2400last                    A   TXBS422.ASM   3118* 3126.
```

```
resendmotion                      A   TXBS422.ASM   1340. 1351. 1380*
resendquery                       A   TXBS422.ASM   2652. 2667. 2716. 2721. 2727. 2736. 2746. 3581*
reset                             A   TXBS422.ASM    983. 1526*
returnvalue                       A   TXBS422.ASM   2257. 2269. 2281. 2286*
rp0                               C   TXBS422.ASM   2426. 2435. 2523. 2526. 2550. 2555. 2562. 2571. 2594. 2604. 3223. 3233. 3241.
                                                    3251. 3341. 3343. 3359. 3361. 3379. 3409. 3474. 3520. 3524. 3544.
                                      P16F876.INC    115*
rp1                               C   TXBS422.ASM   2520. 2528. 2549. 2558. 2560. 2572. 3380. 3545.
                                      P16F876.INC    114*
rsen                              C   P16F876.INC    267*
run1                              A   TXBS422.ASM    936.  941. 1485*
run2                              A   TXBS422.ASM    937.  942. 1484*
run3                              A   TXBS422.ASM    938.  939.  943. 1483*
runreview                         A   TXBS422.ASM    944. 1493*
rx9                               C   P16F876.INC    193*
rx9d                              C   P16F876.INC    202*
s                                 C   P16F876.INC    282*
save3_M                           C   TXBS422.ASM    581* 1838. 1846.
save4_M                           C   TXBS422.ASM    582* 1840. 1848.
save5_M                           C   TXBS422.ASM    583* 1842. 1850.
save6_M                           C   TXBS422.ASM    584* 1844. 1852.
saved_pan_speed_M                 C   TXBS422.ASM    585* 1582. 3155. 3390.
saved_tilt_speed_M                C   TXBS422.ASM    586* 1584. 3157. 3393.
selectbank1                       A   TXBS422.ASM   3222* 3229.
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 14

                            MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

selectbank1now                    A   TXBS422.ASM   3240* 3247.
sen                               C   P16F876.INC    268*
send2hex_3                        A   TXBS422.ASM   2868. 2872. 2874. 2876. 2883. 2890. 2897. 2899. 2901. 2903. 2905. 2907. 2909.
                                                    2916. 2938. 2941. 2943. 2945. 2947. 2949. 2951. 2953. 2963* 3238.
sendack                           A   TXBS422.ASM   1883*
sendack_5                         A   TXBS422.ASM   1628. 1757. 1901. 1906. 1911. 1916. 1921. 2978* 4020. 4117.
sendchecksum_6                    A   TXBS422.ASM   2992* 4138. 4255.
sendcommand                       A   TXBS422.ASM   1115. 1240. 1383. 1431. 1588. 1831* 4168.
senddebugbyte_1                   A   TXBS422.ASM   1961. 1963. 1965. 2029. 2398. 2400. 2966. 2968. 3016*
sendmany0_5                       A   TXBS422.ASM   3055* 4220. 4224. 4228. 4238. 4248.
sendnonmotion                     A   TXBS422.ASM   1394. 1400. 1415. 1422. 1427* 1438. 1454. 1578.
sendone_5                         A   TXBS422.ASM   2998. 3071* 4121. 4123. 4125. 4127. 4129. 4131. 4133. 4135. 4137. 4212. 4214.
                                                    4216. 4218. 4222. 4226. 4230. 4232. 4234. 4236. 4240. 4242. 4244. 4246. 4250.
                                                    4252. 4254.
sendonemore                       A   TXBS422.ASM   3057* 3061.
sendpancommand                    A   TXBS422.ASM   1048* 1703.
sendspectrabyte_0                 A   TXBS422.ASM   3085* 3301. 3304. 3307. 3310. 3313. 3316. 3319.
sendthree                         A   TXBS422.ASM   1550. 1878. 1903. 1908. 1913. 1918. 1923. 1932*
sendtiltcommand                   A   TXBS422.ASM   1174* 1687.
sentbyte_1                        C   TXBS422.ASM    587* 3020. 3031. 3033. 3035. 3086. 3107. 3109. 3111.
set_d_command2                    A   TXBS422.ASM   1369. 1410. 1448. 1471. 1477. 1487. 1497. 1512. 1524. 1829*
setrc216                          A   TXBS422.ASM   2191. 2195. 2199. 2205*
sixtyfourok                       A   TXBS422.ASM   3925. 3932*
slowhalf                          A   TXBS422.ASM   2254. 2261*
smp                               C   P16F876.INC    272*
spare11                           M   TXBS422.ASM    330*
spare12                           M   TXBS422.ASM    331*
spare13                           M   TXBS422.ASM    332*
spare14                           M   TXBS422.ASM    333*
spare15                           M   TXBS422.ASM    334*
spare2                            M   TXBS422.ASM    273*
spare24                           M   TXBS422.ASM    324*
spare28                           M   TXBS422.ASM    328*
spare6                            M   TXBS422.ASM    288*
spbrg                             C   TXBS422.ASM   3523.
                                      P16F876.INC     99*
spectraaddress_M                  C   TXBS422.ASM    588* 2099. 2748. 2758. 2980. 3191. 3296. 3777. 3933. 4021. 4120. 4211.
spectrahigh                       A   TXBS422.ASM   2638. 2654* 2661. 2665.
spectralow                        A   TXBS422.ASM   2640* 2647. 2651.
spectranextout                    A   TXBS422.ASM   3094* 3113.
spectrawait                       A   TXBS422.ASM   2452* 2454.
spectrawaithi                     A   TXBS422.ASM   2656. 2659*
spectrawaitlast                   A   TXBS422.ASM   3121* 3123.
spectrawaitlo                     A   TXBS422.ASM   2642. 2645*
spectrawaitmore1                  A   TXBS422.ASM   3100* 3102.
spectrawaitmore2                  A   TXBS422.ASM   3134* 3136.
spectrawaitstop                   A   TXBS422.ASM   3131* 3139.
speed_save_0                      A   TXBS422.ASM   1335. 1346. 1375. 3153*
spen                              C   TXBS422.ASM    696.
                                      P16F876.INC    192*
sproto10_M                        C   TXBS422.ASM    600* 2904. 3882. 3910. 4039. 4075.
```

```
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 15

                          MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

sproto11_M                         C   TXBS422.ASM    601* 2906. 3887. 3909. 4033. 4040. 4077.
sproto12_M                         C   TXBS422.ASM    602* 2908. 3892. 3908. 4050. 4051. 4055. 4079.
sproto13_M                         C   TXBS422.ASM    603* 2915. 3902.
sproto1_M                          C   TXBS422.ASM    591* 2098. 2121. 2871. 3192. 3201. 3776. 3832. 3921. 3927. 3934. 4022. 4049.
                                                     4057.
sproto2_M                          C   TXBS422.ASM    592*  830.  848.  856.  869.  965.  970.  975.  980.  986.  992.  997. 1002.
                                                     1752. 1761. 1778. 2110. 2122. 2873. 2884. 2891. 2910. 3190. 3193. 3203. 3787.
                                                     3796. 3805. 3814. 3828. 3843. 3853. 3893. 3920. 3943. 4024. 4059.
sproto3_M                          C   TXBS422.ASM    593* 1662. 1667. 1672. 1677. 2119. 2875. 3194. 3195. 3199. 3205. 3827. 3919.
                                                     4025. 4061.
sproto4_M                          C   TXBS422.ASM    594* 1684. 1700. 2882. 3842. 3918. 4027. 4046. 4063. 4166.
sproto5_M                          C   TXBS422.ASM    595* 2889. 3852. 3916. 4030. 4045. 4065.
sproto6_M                          C   TXBS422.ASM    596* 2896. 3862. 3914. 4035. 4044. 4067.
sproto7_M                          C   TXBS422.ASM    597* 2898. 3867. 3913. 4031. 4043. 4069.
sproto8_M                          C   TXBS422.ASM    598* 2900. 3872. 3912. 4037. 4042. 4071.
sproto9_M                          C   TXBS422.ASM    599* 2902. 3877. 3911. 4032. 4041. 4073.
sprotolength_M                     C   TXBS422.ASM    589*  759. 2877. 3399. 3821. 3834.
sren                               C   P16F876.INC    197*
sspadd                             C   P16F876.INC     96*
sspbuf                             C   P16F876.INC     73*
sspcon                             C   TXBS422.ASM   3507.
                                       P16F876.INC     74*
sspcon2                            C   P16F876.INC     94*
sspen                              C   P16F876.INC    174*
sspie                              C   P16F876.INC    242*
sspif                              C   P16F876.INC    138*
sspm0                              C   P16F876.INC    179*
sspm1                              C   P16F876.INC    178*
sspm2                              C   P16F876.INC    177*
sspm3                              C   P16F876.INC    176*
sspov                              C   P16F876.INC    173*
sspstat                            C   P16F876.INC     97*
startout_1                         A   TXBS422.ASM   1953. 3169*
startstart                         A   TXBS422.ASM    694* 3757.
status                             C   TXBS422.ASM    796.  800.  814.  832.  836.  850.  861.  967.  972.  977.  982.  988.  994.
                                                      999. 1004. 1054. 1058. 1062. 1066. 1070. 1074. 1078. 1135. 1180. 1184. 1188.
                                                     1192. 1196. 1200. 1204. 1266. 1573. 1664. 1669. 1674. 1679. 1990. 2001. 2100.
                                                     2123. 2190. 2194. 2198. 2202. 2265. 2267. 2274. 2276. 2278. 2284. 2342. 2348.
                                                     2354. 2360. 2426. 2433. 2435. 2439. 2520. 2523. 2526. 2528. 2549. 2550. 2555.
                                                     2558. 2560. 2562. 2571. 2572. 2594. 2596. 2604. 2755. 2757. 2842. 2865. 2878.
                                                     2886. 2893. 2912. 2934. 3223. 3225. 3233. 3241. 3243. 3251. 3341. 3343. 3344.
                                                     3359. 3361. 3362. 3379. 3380. 3409. 3474. 3520. 3524. 3544. 3545. 3770. 3778.
                                                     3789. 3798. 3807. 3816. 3829. 3835. 3845. 3855. 3895. 3929. 3935. 3939.
                                       P16F876.INC     58*
stickpanin                         A   TXBS422.ASM   1086. 1089. 1092. 1095. 1098. 1101. 1109* 1288.
stickpanin64                       A   TXBS422.ASM   1083. 1111* 1139.
sticktiltin                        A   TXBS422.ASM   1153. 1209. 1212. 1215. 1218. 1221. 1224. 1227. 1236* 1270.
stopall                            A   TXBS422.ASM    902. 1605*
stopfast                           A   TXBS422.ASM    897. 1580*
stopfaster                         A   TXBS422.ASM    913. 1581*
stopfocus                          A   TXBS422.ASM    890. 1590*
MPASM 03.00 Released          TXBS422.ASM   3-13-2002  14:10:12         PAGE 16

                          MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

stopiris                           A   TXBS422.ASM    901. 1600*
stopmovement                       A   TXBS422.ASM   1574* 1612.
stoppan                            A   TXBS422.ASM    882. 1556*
stoppantilt                        A   TXBS422.ASM   1561. 1568*
stoptilt                           A   TXBS422.ASM    886. 1563*
stopwaiting                        A   TXBS422.ASM   2597. 2603*
stopzoom                           A   TXBS422.ASM    894. 1595*
store3                             A   TXBS422.ASM   2108. 2118*
sync                               C   P16F876.INC    300*
t0cs                               C   P16F876.INC    230*
t0ie                               C   P16F876.INC    126*
t0if                               C   P16F876.INC    129*
t0se                               C   P16F876.INC    231*
t1ckps0                            C   P16F876.INC    152*
t1ckps1                            C   P16F876.INC    151*
t1con                              C   TXBS422.ASM    752. 3503. 4096.
                                       P16F876.INC     70*
```

```
t1insync                          C   P16F876.INC    155*
t1oscen                           C   P16F876.INC    153*
t1sync                            C   P16F876.INC    156*
t2ckps0                           C   P16F876.INC    168*
t2ckps1                           C   P16F876.INC    167*
t2con                             C   TXBS422.ASM   3504.
                                      P16F876.INC     72*
target1                           A   TXBS422.ASM    946. 1522*
target2                           A   TXBS422.ASM    947. 1521*
target3                           A   TXBS422.ASM    948. 1520*
target4                           A   TXBS422.ASM    949. 1519*
target5                           A   TXBS422.ASM    956. 1518*
target6                           A   TXBS422.ASM    957. 1517*
target7                           A   TXBS422.ASM    958. 1516*
temp1_0                           C   TXBS422.ASM    605*  871. 1133. 1138. 1264. 1269. 2185. 2189. 2252. 2266. 2268. 2272. 2279.
                                                     2285. 2322. 2467. 2468. 2529. 2547. 2559. 2835. 2838. 2840. 2845. 2846. 2848.
                                                     3220. 3236. 3615. 3616.
temp2_0                           C   TXBS422.ASM    606* 2273. 2275. 2277. 2280. 2430. 2465. 2472. 2592. 3221. 3239. 3325. 3353.
terminatepattern                  A   TXBS422.ASM   1005. 1920*
this_command1_M                   C   TXBS422.ASM    607* 1398. 1399. 1413. 1414. 1420. 1426. 1529. 1592. 1601. 1602. 1606. 2062.
                                                     3395.
this_command2_M                   C   TXBS422.ASM    608* 1143. 1144. 1148. 1149. 1278. 1279. 1283. 1284. 1421. 1425. 1436. 1437.
                                                     1452. 1453. 1530. 1557. 1558. 1564. 1565. 1569. 1591. 1596. 1597. 1607. 1682.
                                                     1690. 1694. 1698. 2064. 3396.
this_pan_speed_M                  C   TXBS422.ASM    609* 1052. 1112. 1130. 1337. 1348. 1377. 1560. 1583. 1585. 1609. 1701. 2066.
                                                     3154. 3391.
this_tilt_speed_M                 C   TXBS422.ASM    610* 1178. 1237. 1261. 1339. 1350. 1379. 1567. 1611. 1685. 2068. 3156. 3394.
thisbit_0                         C   TXBS422.ASM    611* 2754.
thisioerror_0                     C   TXBS422.ASM    612* 2305. 2316.
threebytemessage_6                A   TXBS422.ASM   1933. 3189* 3752.
threequartervalue                 A   TXBS422.ASM   2259. 2271*
tiltdown                          A   TXBS422.ASM    885. 1147*
tiltup                            A   TXBS422.ASM    884. 1142*
timeout_0                         C   TXBS422.ASM    613* 2016. 2017. 3099. 3101. 3120. 3122. 3133. 3135.
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE 17

                          MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

tmr0                              C   P16F876.INC     56*
tmr1cs                            C   P16F876.INC    157*
tmr1h                             C   TXBS422.ASM    743.  747. 3754. 4094.
                                      P16F876.INC     69*
tmr1ie                            C   P16F876.INC    245*
tmr1if                            C   P16F876.INC    141*
tmr1l                             C   TXBS422.ASM    746. 3753. 4093.
                                      P16F876.INC     68*
tmr1on                            C   TXBS422.ASM    752. 4096.
                                      P16F876.INC    158*
tmr2                              C   P16F876.INC     71*
tmr2ie                            C   P16F876.INC    244*
tmr2if                            C   P16F876.INC    140*
tmr2on                            C   P16F876.INC    166*
tosensormatic_4                   A   TXBS422.ASM   2981. 3059. 3073. 3202. 3204. 3206. 3219* 4058. 4060. 4062. 4064. 4066. 4068.
                                                     4070. 4072. 4074. 4076. 4078. 4080.
tospectra_2                       A   TXBS422.ASM   1533. 1860. 2043. 2490. 2811. 3290* 3593. 4087.
toutps0                           C   P16F876.INC    165*
toutps1                           C   P16F876.INC    164*
toutps2                           C   P16F876.INC    163*
toutps3                           C   P16F876.INC    162*
trisa                             C   TXBS422.ASM   3430.
                                      P16F876.INC     88*
trisb                             C   TXBS422.ASM   3444.
                                      P16F876.INC     89*
trisc                             C   TXBS422.ASM   3458.
                                      P16F876.INC     90*
trmt                              C   TXBS422.ASM   2599. 3228. 3246.
                                      P16F876.INC    302*
try2                              A   TXBS422.ASM   2096. 2106*
tspeed1                           A   TXBS422.ASM   1181. 1229*
tspeed2                           A   TXBS422.ASM   1185. 1226*
tspeed3                           A   TXBS422.ASM   1189. 1223*
tspeed4                           A   TXBS422.ASM   1193. 1220*
tspeed5                           A   TXBS422.ASM   1197. 1217*
tspeed6                           A   TXBS422.ASM   1201. 1214*
tspeed7                           A   TXBS422.ASM   1205. 1211*
tspeed8                           A   TXBS422.ASM   1208* 1267.
tx8_9                             C   P16F876.INC    298*
tx9                               C   P16F876.INC    296*
tx9d                              C   P16F876.INC    303*
```

```
txd8                            C   P16F876.INC    304*
txen                            C   P16F876.INC    299*
txie                            C   P16F876.INC    241*
txif                            C   P16F876.INC    137*
txreg                           C   TXBS422.ASM   3234.
                                    P16F876.INC     79*
txsta                           C   TXBS422.ASM   2599. 3228. 3246. 3521.
                                    P16F876.INC     98*
ua                              C   P16F876.INC    290*
unknown                         A   TXBS422.ASM    929.  959.  995. 1007. 1622*
MPASM 03.00 Released         TXBS422.ASM   3-13-2002  14:10:12         PAGE 18

                           MPASM Cross Reference File

LABEL                             TYPE FILE NAME    SOURCE FILE REFERENCES
-----                             ---- ---------    ----------------------

unknown80                          A   TXBS422.ASM    879. 1900*
unknown96                          A   TXBS422.ASM    906. 1905*
unknownc1                          A   TXBS422.ASM    973. 1616*
variablespeed                      A   TXBS422.ASM    968. 1660*
version                            A   TXBS422.ASM    989. 4115*
w                                  C   P16F876.INC     50*
wait                               A   TXBS422.ASM   2777* 2778. 2787. 2790.
waitforlastbit                     A   TXBS422.ASM   2593* 2600.
waituntilhigh                      A   TXBS422.ASM   3569* 3571. 3575.
wcol                               C   P16F876.INC    172*
wr                                 C   TXBS422.ASM   2552. 2569.
                                       P16F876.INC    319*
wren                               C   TXBS422.ASM   2564. 2570.
                                       P16F876.INC    318*
wrerr                              C   P16F876.INC    317*
xittospectra                       A   TXBS422.ASM   3323. 3332* 3350. 3357.
z                                  C   TXBS422.ASM    800.  814.  832.  836.  850.  861.  967.  972.  977.  982.  988. 1004. 1573.
                                                     1664. 1669. 1674. 1679. 2100. 2123. 2190. 2194. 2198. 2202. 2342. 2348. 2354.
                                                     2360. 2865. 2878. 2886. 2893. 2912. 2934. 3770. 3778. 3789. 3798. 3807. 3816.
                                                     3829. 3835. 3845. 3855. 3895. 3929. 3935. 3939.
                                       P16F876.INC    118*
zoomin                             A   TXBS422.ASM    891. 1435*
zoomout                            A   TXBS422.ASM    892. 1440*


LABEL TYPES
-----------
A  Address
C  Constant
E  External
M  Macro
S  Segment
U  Unknown
V  Variable
X  Other
```

APPENDIX C

Index