Camera Pointing

21 November 2006

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

Figure **Page**

[1] `$Header: d:/Angles/RCS/CamPoint.tex,v 1.5 2006-11-21 12:09:10-08 Hamilton Exp Hamilton $`

LIST OF TABLES

1 Introduction

From time to time questions are asked about how to point a camera. This is an attempt to answer that question. Specifically an approach will be outlined here for solving the problem of: "If some one points at a location on a map, how do we get a camera to point at that location?" The method of determining the correct pointing angles for a Pelco camera requires some basic trignometry in converting from locations specified as Latitude/Longitude (or any other type of X/Y coordinates) pairs into the polar coordinate system that Pelco uses.

If there are any additional questions about the mathematics behind the solutions in this note, the reader should consult with a member of the nearest school's mathematics department.

2 How to Point a Camera

Pelco cameras are moved by several different methods:

1. Under an operator's direct control with a joy stick, or equivalent.
2. Under an operator's indirect control by any of:
   2.1. Sending a previously set preset command.
   2.2. Requesting a previously saved pattern.
   2.3. Responding to previously saved alarm response when an alarm occurs. This may result in a preset or pattern command being executed.
3. Via an external command to move to a different angular position. These commands usually are generated by a computer.

All of the above methods of causing a camera to move are operator intensive actions, **except** for the last one. Computer caused motion is the subject for this note.

2.1 Available D Protocol Controlled Motion Commands

There are several commands available for use with direct computer control of Pelco cameras. In this list commands that are usually sent by a joy stick, or equivalent, are omitted. (For more details, see the D Protocol document.)

N.B.: Between the time that a value is set and the time that it is read out, an operator may have made an inadvertent change. This may happen by them using a joy stick at the same time that commands are being sent from a computer. The internal logic of Pelco cameras is that the most recent valid command is acted on. Thus if some commands are coming in in serial, (RS-422) and other commands are coming in via the video connection (Coaxitron), then some things may not work as expected.

1. `EC_SET_ZERO`, `0x49` 73_{10}: Sets the on-screen display to have "zero degrees" be the current pan position. Does not change the values returned or used by any commands except for `EC_EVEREST` sub-opcode `ECS_EVEREST_AZIMUTH_ZERO_QRY` which reads out the difference between the position set with this command and cal_0.

[2] `$Header: d:/Angles/RCS/Intro.inc,v 1.2 2006-11-21 12:09:11-08 Hamilton Exp Hamilton $`
[3] `$Header: d:/Angles/RCS/Pointing.inc,v 1.5 2006-11-21 12:09:11-08 Hamilton Exp Hamilton $`

2. EC_SET_PAN, 0x4B 75_{10}: Set pan angular position in hundredths of degrees from cal_0. Range is $0 \rightarrow 35999$. Values increase in a clockwise direction as viewed from above a normally installed unit. (This includes an ExSite in inverted position.)

3. EC_SET_TILT, 0x4D 77_{10}: Set tilt angular position in hundredths from the horizontal. Range is $0 \rightarrow 35999$. Horizontal is 0, down is 9000 which is displayed on the screen as -90^o, up is 27000 which displays as 90^o.

4. EC_SET_ZOOM, 0x4F 79_{10}: Sets the zoom position as a ratio based on the current zoom limit setting. Usually not a worthwhile command.

5. EC_QUERY_PAN, 0x51 81_{10}: Requests the unit's current pan angular offset from cal_0. Reads out in hundredths of a degree. Returns an EC_PAN_RESP response.

6. EC_QUERY_TILT, 0x53 83_{10}: Requests the unit's current tilt angular offset from the horizontal. Reads out in hundredths of a degree. Returns an EC_TILT_RESP response.

7. EC_QUERY_ZOOM, 0x55 85_{10}: Requests the current zoom position ratio. Returns an EC_ZOOM_RESP response.

8. EC_PAN_RESP, 0x59 89_{10}: Response to a EC_QUERY_PAN command. Has the pan angular offset from cal_0 in hundredths of a degree in it.

9. EC_TILT_RESP, 0x5B 91_{10}: Response to a EC_QUERY_TILT command. Has the tilt angular offset from the horizontal in hundredths of a degree in it.

10. EC_ZOOM_RESP, 0x5D 93_{10}: Response to a EC_QUERY_ZOOM command. Has the ratio of the zoom in it.

11. EC_SET_MAG, 0x5F 95_{10}: Sets the zoom position as a "times" (x2, x8, x7.3, etc.) value. The zoom position is specified in hundredths of a zoom power. I.e. $123 = \times 1.23$.

12. EC_QUERY_MAG, 0x61 97_{10}: Requests the current value of the zoom settings in hundredths of a zoom power. Returns an EC_MAG_RESP response.

13. EC_MAG_RESP, 0x63 99_{10}: Response to a request for the current zoom power. Is in hundredths of a zoom power. I.e. $234 = \times 2.34$.

14. EC_EVEREST, 0x75 117_{10}: Everest project special op-code. Of the various sub-op-codes that Everest commands use, the following are the most important for computer controlled motion:

    14.1. ECS_EVEREST_AZIMUTH_ZERO_QRY, 0x00 00_{10}: Request the current displayed pan offset from cal_0. Returns a value from $0 \rightarrow 35999$ in hundredths of a degree.

    14.2. ECS_EVEREST_AZIMUTH_ZERO_RSP, 0x01 01_{10}: This is the response to an ECS_EVEREST_AZIMUTH_ZERO_QRY command.

15. From time to time other commands may be added to the protocol. For full details, and any more recent information, see the D Protocol specification. None of these motion commands for computer use are implemented in any other protocol. This includes P Protocol and Coaxitron.

2.2 Determining a Pointing Angle

Determining a pointing angle is basically an exercise in trigonometry. In a "normal" case, the user will know the location of the camera and be requested to point the camera at a target.

Typically the camera's (C) position is not moving and is well known at a location that may be specified as $CamLat$ and $CamLon$ (Camera Latitude and Camera Longitude).

The target's position (T) is a variable location and is at $TgtLat$, $TgtLon$.

The problem is now: "What angle do we now tell the camera to rotate to in pan?" A similar, but simpler, problem also exists for tilt.

A simplified explanation of what has to be done:

1. During installation, have the camera properly set up to point to "true north" by using the `EC_SET_ZERO` op-code. Or use the menu command `SET AZIMUTH` option. In an ideal world this will never change and the rest of this algorithm assumes that it does not change.

2. Use the calculations outlined in Appendix A, page A-1 to determine the target's bearing angle from the camera.

3. Use `EC_EVEREST` with an sub-opcode of `ECS_EVEREST_AZIMUTH_ZERO_QRY` to get the cal_0 offset. If there is no chance that the camera has not had the value changed since its original installation, this value may have been calculated once during installation. However sometimes the camera may be replaced and asking the camera each time through is usually a good idea. The important point here is to have the offset available for the rest of the calculations.

4. Modify the angle that was determined with the arccos function, by the cal_0 offset value and send that value back to the camera using the `EC_SET_PAN` command.

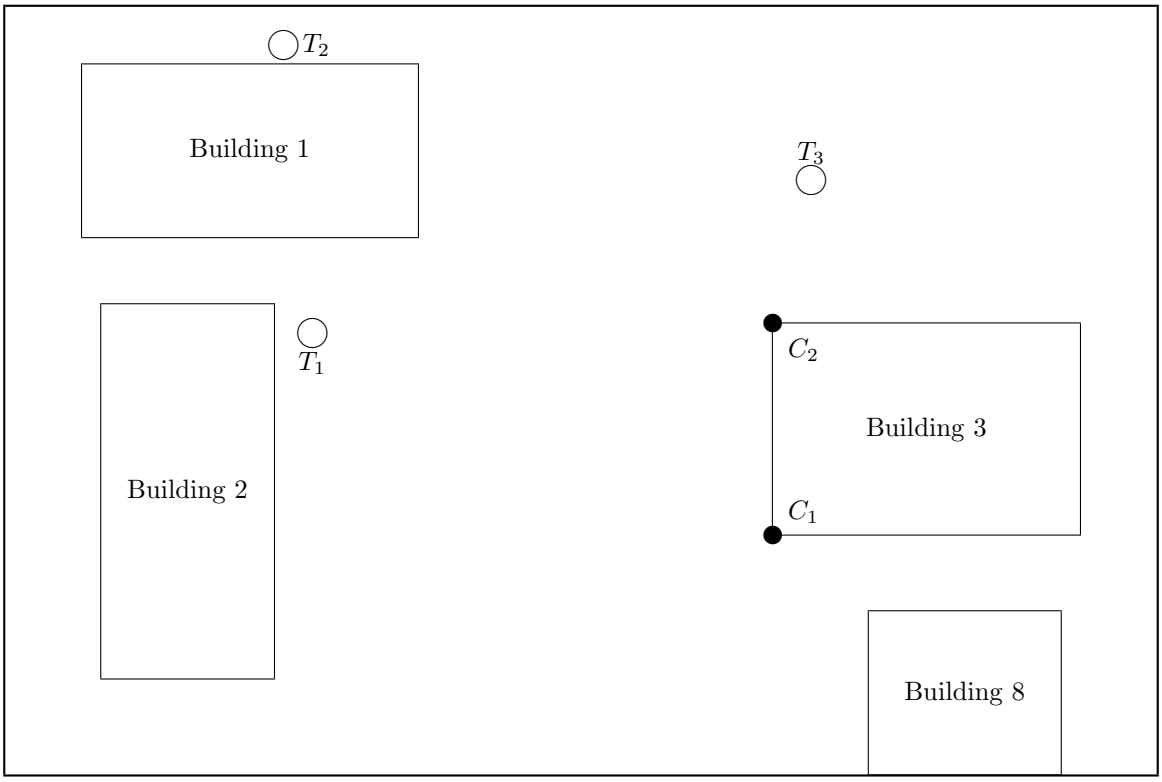
5. The camera will move to the desired location in a few seconds at "preset speed" of about 100^o/sec. (Slower for ExSite and Esprit systems.)

2.3 Some Solved Examples

For these examples, a section of the Pelco campus has been selected. A simplified map of the campus is shown in Figure 1, page 6. The basic data was obtained by running Google Earth and zooming into Pelco's location on the Earth. The center of the image used in making this simplified map is about 119^o 42'58.5" W, 36^o 47'18.8" N with an eye elevation at 1722 feet.

The two cameras and four targets used in the examples Are located as follows:

[4] `$Header: d:/Angles/RCS/Pelco.inc,v 1.4 2006-11-21 12:09:11-08 Hamilton Exp Hamilton $`



$RCSfile: Pelco.inc,v $

Figure 1. Pelco Buildings 1, 2, 3 and 8 (partial)

C_1	Camera #1	36^o 47' 15.94" N	119^o 42' 55.33" W	On top of building 3, south west corner.
C_2	Camera #2	36^o 47' 18.18" N	119^o 42' 55.39" W	On top of building 3, north west corner.
T_1	Target #1	36^o 47' 17.94" N	119^o 43' 01.55" W	Outside picnic area of the Blue Pride Cafe.
T_2	Target #2	36^o 47' 21.12" N	119^o 43' 02.10" W	Entrance to building 1.
T_3	Target #3	36^o 47' 19.78" N	119^o 42' 54.77" W	California Memorial to 9-11.
T_4	Target #4	36^o 46' 24.06" N	119^o 43' 17.51" W	Airport Control Tower.

Table 1. Camera and Target Locations expressed in Latitude and Longitude.

Between	Bearing	Distance
$C_1 \rightarrow T_1$	292	544
$C_1 \rightarrow T_2$	314	759
$C_1 \rightarrow T_3$	7	391
$C_1 \rightarrow T_4$	199	5552

Table 2. Distance and Bearings between Cameras and Targets.

2.4 Sample Worked Problems

2.4.1 Point Camera #1 at Target #1

It is assumed that camera 1 has been assigned an address of 1 for all these steps.

1. Send the following command to the camera to get any possible offset of the readout *vs.* cal_0.

```
0xFF    Sync
0x01    Address
0x00    cmnd1 is set to ECS_EVEREST_AZIMUTH_ZERO_QRY
0x75    cmnd2 is set to EC_EVEREST
0x00    data1 is set to 0
0x00    data2 is set to 0
0x76    Checksum of all bytes except for Sync
```

2. The camera would send a response that might be similar to this:

```
0xFF    Sync
0x01    Address
0x01    cmnd1 is set to ECS_EVEREST_AZIMUTH_ZERO_RSP
0x75    cmnd2 is set to EC_EVEREST
0x02    data1 is set to 0x02
0x00    data2 is set to 0x00
0x79    Checksum of all bytes except for Sync
```

   This indicates that a `EC_SET_ZERO` (`0x49`) command has been sent and has changed the north readout on the camera so that it is no longer at cal_0 but is `0x0200` or 2.56^o clockwise.

3. Solving the formulas in Appendix A, page A-1, we find that the target is about 544 feet away and is on a bearing of 292^o from the camera.

4. We now send the following to the camera to cause it to move to an azimuth angle of 292^o:

```
0xFF    Sync
0x01    Address
0x00    cmnd1 is set to 0x00
0x4B    cmnd2 is set to EC_SET_PAN
0x70    data1 is set to 0x70 (See below)
0x10    data2 is set to 0x10 (See below)
0xCC    Checksum of all bytes except for Sync
```

   4.1. $292 \times 100 = 29200$

   4.2. $29200_{10} = 7210_{16}$

   4.3. $7210_{16} - 200_{16} = 7010_{16}$

5. And get the following reply (assuming that no alarms are active):

```
0xFF    Sync
0x01    Address
0x00    No active alarms
0xCC    Checksum
```

The checksum is the checksum of the command that caused this response to be sent **with** the alarm byte added in. I.e if alarm 1 was active the checksum would then be `0xCD`.

6. At this point Camera #1 will move to point at Target #1. If an alarm occurs during the move, then the camera will do as directed by the alarm action.

2.4.2 Point Camera #1 at Target #2

At Pelco all the buildings are two stories high. Target #2 is on the ground on the back side of building #1 and may not be seen from Camera #1. The camera may point in that direction but may not actually see the target. Solving this type of problem, hidden line problem, is beyond the scope of this note.

2.4.3 Point Camera #1 at Target #3

Again Pelco building #3 is blocking the view of the ground at the 9-11 memorial. However Camera #1 will be able to see the upper portion of the memorial flag pole because it is 100' high and the building is only 30' high. To do the pointing proceed in the same way as in Section 2.4.1, page 8.

2.4.4 Point Camera #2 at Target #3

Proceed in the same way as in Section 2.4.1, page 8. Note that if two users want to move the same camera to different places, it becomes a problem that is outside the scope of this short note.

APPENDIX A

A Calculating Camera Bearings

When the location of your camera is known and you want to point the camera at a given location. Such as might arise when a user "clicks" on a map. The angular direction from the camera must be calculated. This is called the **bearing angle** from the camera to the target. Pelco cameras refer to the angle in azimuth as a "pan" angle and it is valued in hundredths of a degree. I.e. to point at 37^o North, the value sent to a Pelco camera would be 3700.

Note

1. The various formulas have the following uses:
   - 1.1. Eq. 1 is used to determine the distance in radians between the camera and the target. To convert *Dist* to linear measure, one of the equations 6, 7 or 8 must be used.
   - 1.2. Eq. 2 is used to determine the bearing in radians from the camera to the target. The output of the equation is a value from $0 \rightarrow 1$ radian.
     When the value of sin(*DeltaLon*) is negative, the correct range of the output should be $1 \rightarrow 2$ radian, to force the bearing angle into the correct range subtract *CamAngle* from 2π.
     To convert from radians to degrees, use Eq. 4.
   - 1.3. Eq. 3 is used to convert $Degrees_{10}$ into radians. Math functions typically require that their arguments be expressed as radians not degrees.
   - 1.4. Eq. 4 is used to convert radians into $Degrees_{10}$ for use with camera pointing commands.
   - 1.5. Eq. 5 is used for converting from `DDD MM SS.SS` format into $Degrees_{10}$ format.
   - 1.6. Eq. 6, 7 and 8 are used to convert $Dist_{10}$ into various types of linear distances. Note that the output of these algorithms are in radians and it must be converted to $Degrees_{10}$ prior to using any of these equations.
2. Formula notes:
   - 2.1. These formulas were taken from the 1982 ARRL antenna book and were copied/modified from the formulas shown at: "`http://www.ac6v.com/GreatCircle.htm`".
   - 2.2. Longitude must be in the range of $\pm 180^o$. (By convention longitude values in the Americas will almost always be negative values.)
   - 2.3. Latitude must be in the range of $\pm 90^o$. (By convention latitude values north of the equator are always positive. Southern latitudes are always negative.)
   - 2.4. The resulting bearing will be in the range of $0 \rightarrow 360^o$.
   - 2.5. In the original write ups for these formulas, it is consistently indicated that the Latitude/-Longitude values are in degrees. However when calling math functions, the arguments **must** be expressed in radians.

[5] `$Header: d:/Angles/RCS/Bearing.inc,v 1.2 2006-11-21 12:09:10-08 Hamilton Exp Hamilton $`

$$Dist = \arccos(\sin(CamLat) \times \sin(TgtLat)) + (\cos(CamLat) \times \cos(TgtLat) \times \cos(\Delta Lon))) \quad (1)$$

$$CamAngle = \arccos(\frac{(\sin(TgtLat) - (\sin(CamLat) \times \cos(Dist)))}{(\cos(CamLat) \times \sin(Dist))}) \quad (2)$$

$$Radians = Degrees_{10} \times (\frac{\pi}{180}) \quad (3)$$

$$Degrees_{10} = Radians \times \frac{180}{\pi} \quad (4)$$

$$Degrees_{10} = degree + \frac{minutes}{60} + \frac{seconds}{3600} \quad (5)$$

$$Dist_{miles} = Dist \times 69.041 \quad (6)$$

$$Dist_{feet} = Dist \times (69.041 \times 5280) \quad (7)$$

$$Dist_{kilometers} = Dist \times 111.111 \quad (8)$$

Where:

1. $Degrees_{10}$ = Latitude or Longitude expressed as `dd.mmssss`.
   1.1. `dd` = full degrees
   1.2. `mm` = minutes
   1.3. `ssss` = seconds and hundredths of a second
2. $CamLat$ = Camera's location in latitude in radians.
3. $TgtLat$ = latitude of the target in radians.
4. ΔLon = Camera's longitude, in radians, minus that of the other location. (Algebraic difference.)
5. $Dist$ = Angular distance along path in radians.
6. $CamAngle$ = True bearing from north if the value for $\sin \Delta Lon$ is positive. If $\sin \Delta Lon$ is negative, true bearing is 360^o - $CamAngle$.

A.1 Sample Algorithm Fragment

A sample program fragment to determine the bearing angle, in Basic, follows:

```
' Convert Latitude/Longitude data from degrees to radians
CamLat   = CamLat * (Pi/180)
CamLon   = CamLon * (Pi/180)
TgtLat   = TgtLat * (Pi/180)
TgtLon   = TgtLon * (Pi/180)

' Get difference in longitude values
DeltaLon = TgtLon - CamLon

' First get the distance between the camera and the target
' Use two temps to keep the line lengths short
Temp1 = (sin(CamLat) * sin(TgtLat))
Temp2 = (cos(CamLat) * cos(TgtLat) * cos(DeltaLon))
Dist  = acos(Temp1 + Temp2)

' Get the distance in feet
Dist_foot = Dist * (180/Pi) * (69.041*5280)

' Now get the bearing angle of the target from the camera
' Use two temps to keep the line lengths short
Temp1    = ((sin(TgtLat) - sin(CamLat) * cos(Dist)))
Temp2    = (cos(CamLat) * sin(Dist))
CamAngle = acos(Temp1/Temp2) * (180/Pi)

' Compensate for values in the third and fourth quadrants
if sin(DeltaLon) < 0 then CamAngle = 360 - CamAngle
```

APPENDIX B

B Converting Latitude and Longitude Locations to Radians

The first step is to change the format of Latitude/Longitude data into $Degrees_{10}$ format. We then convert the latitude/longitude values into radians. For camera C_1 we have:

1. Latitude = 36^o 47' 15.94" N: $36 + \frac{47}{60} + \frac{15.94}{3600} = 36.787761111$
   $36.787761111 \times \frac{\pi}{180} = 0.64206755$
2. Longitude = 119^o 42' 55.33" W: $119 + \frac{42}{60} + frac55.333600 = 119.71536944$
   $119.71536944 \times \frac{\pi}{180} = 2.089427361$ Since this is a West longitude it is negative -2.089427361.

And for target T_1 we have:

1. Latitude = 36^o 47' 17.94" N: $36 + \frac{47}{60} + \frac{17.94}{3600} = 36.7883166$
   $36.7883166 \times \frac{\pi}{180} = 0.64207725$
2. Longitude = 119^o 43' 01.55" W: $119 + \frac{43}{60} + \frac{1.55}{3600} = 119.71709722$
   $119.71709722 \times \frac{\pi}{180} = 2.089457517$
   Since this is a West longitude it is negative -2.089457517.

If the supplied latitude/longitude pair are in other formats, they must be converted into radians before continuing.

[6] `$Header: d:/Angles/RCS/CvtLL.inc,v 1.1 2006-11-21 12:05:58-08 Hamilton Exp Hamilton $`

APPENDIX C

C Interpreting Pan and Tilt D Protocol Readout Replies

Pan and tilt angle values comes in in two bytes as degrees times 100 "hungrees".

Position	D reads out as	Spectra displays as
90^o up	27000	90^o
45^o up	31500	45^o
Horizontal - 1^o	35900	1^o
Horizontal	000	0^o
45^o down	4500	-45^o
90^o down	9000	-90^o

Position Pointing direction of the enclosure/camera

D reads out as D protocol returned value for this angle

Spectra displays as What is displayed on the Spectra screen

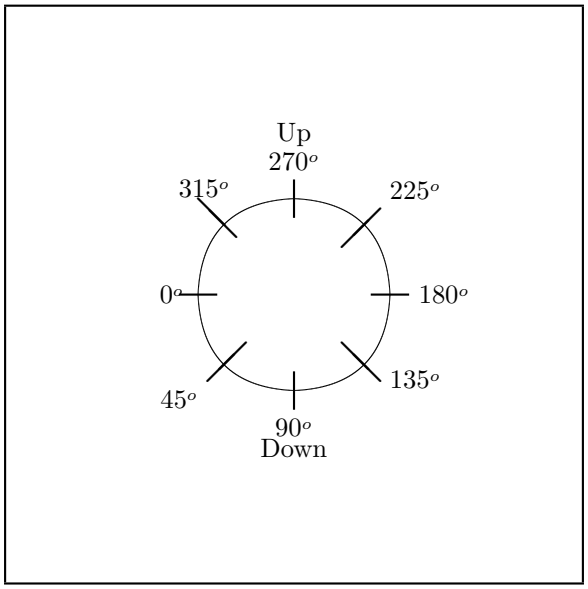


Figure C-1. D Protocol Tilt Readout Positions

[7] $Header: d:/Angles/RCS/ReadOut.inc,v 1.3 2006-11-21 12:09:11-08 Hamilton Exp Hamilton $

C.1 Sample Pan Angle Correction Software

In the below example several variables and functions are used:

1. Protocol Command Values:
   1.1. `SDcmnd1` and `SDcmnd2`, UNSIGNED CHARs are used to hold commands to the Spectra.
2. Calculated Intermediate Values:
   2.1. `offset` is a SIGNED INT which holds the results of asking the Spectra what the Azimuth offset is.
   2.2. `temp` is a SIGNED LONG which holds the result of modifying the reported value from the Spectra by the offset.
3. Macros/defines used:
   3.1. `EC_EXTENDED_REPLY_LENGTH` is the length of a D Protocol reply that contains Azimuth or Elevation data. The reply is 7 (seven) bytes in length.
   3.2. `DREPLY_DATA1` with an index value of 5 and
   3.3. `DREPLY_DATA2` which has an index value of 6.
4. Arrays used:
   4.1. `Dreply` is a 7 UNSIGNED CHAR to receive the Spectra reply into. The two positions used here are:
      4.1.1. `DREPLY_DATA1` with an index value of 5 and
      4.1.2. `DREPLY_DATA2` which has an index value of 6.
5. Functions called:
   5.1. `SCheckSumD()`
   5.2. `GetDReply()`
6. The results are in two UNSIGNED CHARs:
   6.1. `HpanU` this is the upper half of the pan angle when modified by the Set Azimuth Zero value.
   6.2. `HpanL` this is the lower half of the pan angle when modified by the Set Azimuth Zero value.

```
// Get pan angle offset from zero
SDcmnd1 = ECS_EVEREST_AZIMUTH_ZERO_OFFSET_QRY;
SDcmnd2 = EC_EVEREST;      // This is an Everest op-code
SCheckSumD(YES_REPLY);
GetDReply(EC_EXTENDED_REPLY_LENGTH);  // Put it in the reply buffer
offset = ((DReply[DREPLY_DATA1]*256) + DReply[DREPLY_DATA2]);

// Get pan angle SDcmnd1, SDdata1 and SData2 don't change anymore
SDcmnd1 = 0x00;
SDcmnd2 = EC_QUERY_PAN;   // What is the current azimuth reading
SCheckSumD(YES_REPLY);
GetDReply(EC_EXTENDED_REPLY_LENGTH);  // Put it in the reply buffer

// Pan angle comes in in two bytes as degrees times 100 "hungrees"
// Value has to be rounded (i.e. that is why there is a "+ 50" here)
//
// The pan angle reported by an EC_QUERY_PAN command is not
// offset by the EC_SET_ZERO command. But the on screen display
// is. So here we have to modify the reported output by the
// changed pan offset value.
//
// If EC_SET_ZERO has been used to set pan zero to 25 degrees,
// and the on-screen display is now reading 50 degrees in pan,
// then the reply from a EC_QUERY_PAN command will be 75 degrees.
// In general we should have the angle reported to the outside
// world match what is seen on the screen. Thus there is logic to
// request the actual offset value and to use that in modifying
// the reported value so that it matches the on-screen value.
//
temp  = ((DReply[DREPLY_DATA1]*256) + DReply[DREPLY_DATA2]);

temp -= offset;              // Get difference of real vs display
if (temp < 0)                // Too small
{
    temp += 36000;           // Yep, let it wrap up
}
temp += 50;                  // Round
temp /= 100;                 // Convert from hungrees to decimal
HpanU    = (unsigned char) (temp/256);
HpanL    = temp & 0xFF;
```

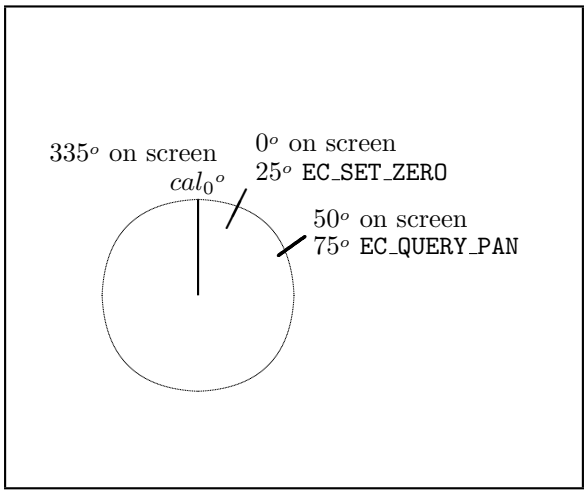


Figure C-2. D Protocol Pan Readout Positions

APPENDIX D

D Web References

All of the following web sites provided useful information for this short note. In general they have more information that will probably be useful and it is recommended that the data they have available be examined carefully. These web references are in alphabetical order.

1. http://jacq.istos.com.au/sundry/navcel.html
2. http://www.ac6v.com/GreatCircle.htm
3. http://www.gazza.co.nz
4. http://www.movable-type.co.uk/scripts/LatLong.html
5. http://www.movable-type.co.uk/scripts/GIS-FAQ-5.1.html
6. http://www.pilotsweb.com/navigate/dis_dir.htm
7. http://www.satsig.net/ssazran.htm

[10] $Header: d:/Angles/RCS/Refs.inc,v 1.2 2006-11-21 12:09:12-08 Hamilton Exp Hamilton $

APPENDIX E

Index