

SMT DSP code Documentation

Tom Sargent, Steward Observatory

Introduction

This document describes the software that runs in the the Traquair HEPC2104 PC/104 DSP card. This is a PC/104 CPU card that is interfaced both to the A/D interfaces reading the outputs of the SMT shaft encoders and to the ISA bus of the "smmts" Linux host computer. So, the DSP is able to read the shaft encoder signals, compute the shaft rotation angles and report those shaft positions to the host Linux machine.

Physically, the DSP card is in a PC/104 stack with 2 other cards. That stack in turn is connected to the smmts computer's ISA bus. The first PC/104 card is the Brandywine IRIG interface which is used by the Linux host as a precision time reference. The second is the HEBASE104, which has cables that connect to two remotely mounted HEGD2 A/D modules which digitize the voltages coming from the shaft encoders.

Each shaft encoder outputs 3 analog voltages: sine, cosine and reference pulse. The analog signals from the 3 shaft encoders (2 on elevation and 1 on azimuth) comprise a total of 9 signals. Each A/D has 8 channels. One A/D handles the 3 signals from the Azimuth encoder and the other handles the 6 signals from both elevation encoders. Every millisecond, the A/D interfaces digitize all 9 signals.

The software described in this document is in a file called track.c.

Purpose of the DSP Code

The purpose of the DSP code is to report to the Linux host where the telescope is pointing at any given moment. It constantly reads the outputs from the shaft encoders (2 elevation encoders and 1 azimuth encoder) corrects the readings for known errors, and reports the telescope position to the Linux host program nominally 100 times/second. It is intended to be able to point the telescope with an accuracy of 1 arcsecond or better.

The Shaft Encoder Data

The sine wave and cosine waves output by the shaft encoder go through one full wave for each 36.00 arcseconds of shaft rotation (≈ 0.01 degrees). The position of each axis is reported by the DSP to the Linux host as two 32-bit numbers:

- 1) an integer count of the number of whole sine waves since the zero point ("fiduciary" or "index") was passed, and
- 2) a number of milliarcseconds representing the fractional portion of the current sine wave that has been seen so far.

So, for example, a value of 4322 waves plus 25200 milliarcseconds works out to be 43.22 degrees plus 25.2 arcseconds. The origin point to which the 4322 is relative is the point indicated by the fiducial signal (the reference pulse) that is generated by the encoder as the shaft turns past that particular position. The shaft encoder outputs are sampled by 16-bit D/A converters 1000 times/sec. That sample rate is constant regardless of how fast the telescope is moving or even if the telescope is moving.

The data from the encoders is subject to several sources of error which, together, prevent it from being accurate to better than a few arcseconds. The DSP code attempts to improve this accuracy to about 1 arcsecond or better by correcting the encoder output for 3 major sources of error. These are:

- 1- offset: A constant voltage offset added to each measurement.
- 2- amplitude: the peak to peak amplitude of the sine and cosine signals, which is different for sine and cosine and which varies slowly as a function of position.
- 3- phase error: the phase difference between sine and cosine signals is not quite 0. It also seems to be characteristically different in the forward and reverse directions.

All of these sources of error also probably drift very slowly with time and/or ambient temperature.

The sine/cosine signals that would be produced by a perfect encoder (that is, one without the 3 sources of error listed above) can be thought of as the X, Y coordinates of the points on the circumference of a circle where X is the cosine signal and Y is the sine signal. So, each pair of simultaneous sine/cosine readings forms a unit vector from the center of the circle to its circumference. This unit vector winds around the circle once every time the shaft rotates 36.0 arcsec. The signals are sampled at a constant rate, so when the telescope is moving very slowly the 1KHz sample rate produces hundreds or thousands of samples around that circle. When it is moving at full slew speed, the 1KHz sample rate catches only 6 or 7 samples per cycle. If we were to add the 3 types of errors listed above to the perfect data, instead of a circle we would see the X, Y points describe an ellipse of low eccentricity. It would be offset slightly from the origin and rotated at a small angle. The rotation angle of the ellipse is the phase error. The amplitude difference of the sine and cosine waves determines the eccentricity of the ellipse. The offsets displace the ellipse from the origin. So, in order to determine the correction factors for these 3 types of error, the software has to continuously determine the parameters of the ellipse that the raw encoder data describes.

A 2KHz clock derived from the IRIG interface is used to trigger the A/D converters to sample the analog signals from the encoders. The azimuth encoder and elevation encoder 0 are sampled on one 2KHz clock by the first A/D and the elevation 1 encoder is sampled on the next clock by the second A/D. So, each encoder is sampled 1000 times per second. The DSP processor gets an interrupt 1000 times per second for the elevation encoder data and 1000 times per second for the azimuth encoder data.

The interrupt routine corrects the raw values for the 3 sources of error using the most current values of the correction factors. The interrupt routine then compares the corrected sine/cos pair to the value of the last one which was inserted into a circular data buffer. The new data pair will be inserted into the circular buffer only if it is sufficiently different from the last one. The reason for this is that when the telescope is stopped or moving slowly, consecutive data samples are at essentially taken at the same shaft position. Thousands of data samples could be generated during one 36.0 arcsecond cycle. To decide if the new data point is different enough from the last one, The DSP calculates the phase angle difference between the new data point and the last one that was inserted into the buffer.

To compute the phase angle the interrupt routine regards the 2 numbers as X, Y values describing a unit vector from the origin to the circumference of the unit circle. The phase difference is estimated by computing the dot product of the 2 vectors and computing the Z coordinate of the cross product of the 2 vectors. The dot product gives the cosine of the angle between them and the cross product gives the sine. The ratio of those is the tangent, which for small angles is equal to the angle itself. The acceptance threshold is presently set to $\pi/75$ which is 2.4 degrees. If the angle difference is $\geq \pi/75.0$, the new data pair is saved in the circular buffer. This is the same criterion used by Johan Schraml in

the Effelsberg control system. The 2.4 deg. represents $0.0067 = 2.4/360.0$ of a full circle. Since that full circle represents 36.00 arcseconds of shaft rotation of the encoder, that corresponds to $0.0067 * 36.0 = 0.24$ arcseconds of telescope movement. In this way, the 800 element circular buffer always contains at least a few full cycles worth of recent data. This is enough to perform a reliable least squares fit of an ellipse in order to re-estimate the ellipse parameters and hence the error correction factors. See the section on Ellipse Fit below.

At any moment, the position of the telescope is the current cycle count plus the fractional part of the current 36.0 arcsec wave. It always uses the most recently received X, Y encoder data corrected by the current correction factors to compute this, regardless of whether that data was inserted into the circular buffer or not. So, the position data is never older than 1ms.

When in the track mode (as opposed to the slew mode) every time the code detects a transition in the sine wave from quadrant 1 to 4, or from 4 to 1, AND the last such quadrant change was in the same direction AND there are at least CALIB_MIN data points in the circular buffer, then a new fit of the ellipse to all of the data in the circular buffer (about 800 data points) is performed. In general this combines data from at least the last several full sine/cosine cycles of the shaft encoder. So, it inherently tracks any slow drift in the encoder signals and any systematic change in the encoders as a function of angular position of the telescope. At typical tracking speed of about 10 -20 arcsec/sec. this causes an ellipse fit computation to be done every 2 or 3 seconds for each encoder.

As the telescope speed increases, the time period for a full 36 arcsec. cycle shrinks and ellipse fits are performed more frequently. At slew speeds (5400 arcsec/sec.), given that there are 3 encoders, this criterion would probably cause more ellipse fits per second than the DSP can perform. At the same time it is true that at slew speeds, high precision in the telescope position is not really required. So, in order to decrease the load on the DSP, the period between consecutive ellipse fit calculations is limited to no less than 100ms (see CALIBRATE_PERIOD in track.h). This corresponds to a speed of 360 arcsec/sec. So, at speeds between 360 and 720 arcsec/sec. ellipse fits are performed on every second full cycle and so on.

The Ellipse Fit

The ellipse fit is done using about the simplest method possible. It has been written up in the literature ("A Note on the Least Squares fitting of ellipses" by Paul Rosin, Pattern Recognition Letters v14 1993 p799). The use of the least squares approach is not very robust in this case, however. The typical least squares approach works by picking a functional form $Y = P(X)$, like a polynomial, for instance, and then forming an expression for the difference between the observed data points and that functional form. The coefficients of the best fit polynomial are found by expressing the sum of the squares of those differences and then differentiating that expression with respect to the coefficients of the polynomial and setting that equal to 0. That leads to a system of linear equations, whose solution is the coefficients of the polynomial. Implicit in this is that the function be single valued and that X be the independent variable.

The expression for the general quadratic (any conic section) is not of the $Y = P(X)$ form, however.

$$Q(x,y) = A * X^2 + B * X * Y + C * Y^2 + D * X + E * Y + F = 0$$

Where $Q(x,y)$ is referred to as the Euclidean Distance.

Unfortunately, this is not a single valued function, like a polynomial, so it doesn't lend itself very directly to the usual least squares approach. It is difficult to come up with a computationally useful

method of expressing the distance from an (X,Y) data point to the circumference of the general ellipse. Any such expressions lead to a very complex system of nonlinear equations which are very hard to solve. So we minimize the square of the Q instead of the square of the distance of each point to the ellipse circumference. The sum of the squares of Q has the property that it is zero for a perfect solution and grows as the solution gets poorer, though it's not very clear what "poorer" means in this case. However this does lead to a system of linear equations which are easy to solve.

We can set F to 0 with little loss of generality, then minimize the sum of the squares of Q . This produces 5 linear equations in 5 unknowns (A , B , C , D and E). The drawbacks to this approach are that this form is not as numerically stable as a normal fit to a polynomial and we also have to avoid the degenerate solution of all constants = 0. The solution gets more inaccurate rapidly as the ellipse center departs from the origin. Because of this, the DSP code keeps a running offset which it subtracts from raw the data before it is even inserted into the circular buffer to try to keep it centered. Failure to do this leads to very inaccurate results.

Tests of this algorithm were performed with synthetic data to see how sensitive the solutions were to noise. It seems that it starts to degrade noticeably with one part in 1000 noise. It turns out this is close to what we see at the telescope in the worst case. (This may improve with a more permanent cabling and connecting installation. We saw mostly 60Hz pickup as the major source of noise.)

Numerically, the solution is difficult to do on the T.I. 320C44 DSP chip because it doesn't have 64-bit floating point arithmetic. The sums that have to be computed involve high powers of the data, so high accuracy (i.e., lots of significant digits) is desirable when doing this computation. The T.I. DSP has 40 bit arithmetic at most (although each 40 bit number uses 64 bits worth of memory locations). The solution was done with 32-bit arithmetic to minimize memory consumption and accesses. However, the sums are formed using the `binaryAdder()` function (see `calibrate.c`) which adds numbers in adjacent pairs in the data arrays. This tends to always be adding together two numbers which are nearly of the same order of magnitude, so it minimizes round off error. It does consume a bit more memory, but this extra precaution largely makes up for the lower precision.

The ellipse fit method was also tested with synthetic data against a simpler sine fitting method. The conclusion was that it was good as long as noise was low. Also it is being used successfully at the Effelsberg telescope.

Data Communication between Linux host and DSP

By volume, the communication is almost all from the DSP to the host. A position report record is output at regular intervals, nominally 10ms. In general a report consists of a type code plus status bits in the first word and (for most reports) the Universal Time when the report was sent in the second word. The contents of the remainder of the report varies with the type of the report.

In the description of a position report below each element is 32 bits long. Note that the raw data words are included as of the time the report is generated. This is only a subsample (every 10ms rather than every 1 ms) of the raw data, so that the host can look at it and see if anything is going wrong. It is not a complete transmission of all raw encoder data.

Type = 0 position report to Linux Host

Byte

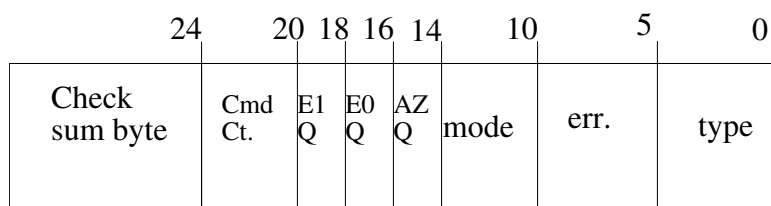
offset Contents

- 0 DSP status and type code word. (unsigned integer). See below. This record type has type field = 0;

- 4 time in units of milliseconds. This is the DSP's tick counter value at the time the position was computed. The tick timer comes from the data clock which in turn comes from the IRIG interface, so it is very accurate. (Treat as an unsigned integer.)
- 8 Azimuth cycle count. signed 32-bit cycle number.
- 12 Azimuth phase. Phase within the current 36.00 arcsecond wave in units of milliarcseconds. A 32-bit integer which is always ≥ 0 .
- 16 Elevation 0 cycle count.
- 20 Elevation 0 phase.
- 24 Elevation 1 cycle count.
- 28 Elevation 1 phase.
- 32 This word contains 3 RMS values as an accuracy check. This is the RMS value of $1.0 - \sqrt{\sin^2 + \cos^2}$ for all the data currently in the circular buffer. It should ideally be 0.0. When we use a signal generator to supply sine and cosine to the program (i.e., a very low noise source) the value of this is about 0.0006. In the lab test setup with a real encoder it runs from 0.01 to 0.006. Values much larger than 0.01 are indicative of a poor ellipse fit solution, and/or noisy data. The 3 RMS values are multiplied by 3000.0 and converted to integers. The 3 integers are then packed into this long word as 3 10-bit numbers. AZ is shifted 20 bits left, EL0 is shifted 10 bits left and el1 is shifted 0 bits left. Servo unpacks this in positionServo().
- 36 Two 16 bit raw encoder data words hi 16 bits is AZ cos, low 16-bits is AZ sine. (At least that's how it's packed in on the DSP. The servo computer is little endian. See the positionServo function.)
- 40 two 16-bit raw encoder data words. Hi 16-bits is AZ ref. data, lo 16 is elev0 cos.
- 44 two 16-bit raw encoder data words. Hi 16-bits = elev0 sine. Low 16 is elev0 ref.
- 48 two 16-bit raw encoder data words. Hi 16-bits is elev1 cos. Low 16 is elev1 sin.
- 52 The 16-bit raw encoder data for elev1 ref, expressed as a 32-bit signed number.

The DSP Status Word

The DSP status word has continuously updated information. It is the first word of every report that comes from the DSP to the host. The error events are reported in the 5 bit err field. As error events are detected by the DSP, the error codes are stuffed into a software FIFO. Whenever the next message is being sent from the DSP to the host, the next error code is removed from that FIFO and OR'ed into the status word of that message. So the error code doesn't necessarily have anything to do with the message that carries it. If the error codes are inserted into the FIFO faster than the rate of reports (normally about 120 per second, total) then some error codes will be missed. In practice, errors are rare.



- 1 Type code determines the type and length of the record which of it is part. 5 bits.
- 2- Error code. See below. 5 bits
- 3- Mode of program. 4 bits
 - 1= normal tracking mode. Valid fiducial, corrections being applied.
 - 2= in the search for fiducial mode. Index word above is 0.
 - 3= correction coefficients not determined yet. Still waiting for at least one full wave so that coefficients can be computed.
- 4 AZQ is an azimuth quality code: Indicates the quality of the Azimuth information which the DSP is sending to the Linux system. It equals 3 when the DSP thinks it knows the telescope position to maximum precision.
 - 0 = DSP has no absolute position info. so, pos'n is meaningless.
 - 1 = An ellipse solution has been done.
 - 2 = Ellipse solution done and coarse position set.
 - 3 = Ellipse Solution, coarse position and ref. pulse used.
- 5 E0Q is elevation encoder 0 quality code. Same meaning as for AZQ.
- 6 E1Q is elevation encoder 1 quality code. Same meaning as for AZQ.
- 7 Cmd. Ct. is the low 4 bits of the count of the number of reports sent by the DSP since it started. This is largely diagnostic.
- 8 Check Sum Byte is computed so that it equals the complement of the sum of the 3 low order bytes. This is used so that the linux system can detect garbage or fragmentary reports in the input FIFO.

Error Codes from the DSP software

These error codes appear in the DSP error code field in the status word which the DSP puts in every report it sends to the Linux host.

Name	Number	Description
BAD_CMD_RECVD	1	DSP received a command from the host with a bad command code.
BAD_CMD_LENGTH	4	DSP received a command from host with a bad length. Length doesn't match that expected for the command code.
ALLOC_ERROR	5	DSP tried to malloc memory space for a circular buffer and malloc failed. This is a fatal error. The DSP can't do anything useful without a circular buffer..
MALLOC_FAILED	7	malloc routine failed (some malloc call other than the one that gets space for the circular buffers).
BAD_INPUT_STATE_VARIABLE	8	The state driven process which decodes commands from the host got a bad state variable.
NO_DOWNCOUNTERS	9	A call to changeDownCounters() to specify a new position report period returned error.
NO_DECREMENTERS	10	A call to registerDownCounter() in order to set up a raw data recording session, returned an error. There may be no unused downcounters left.
SYNC_ERROR1	11	The comport 1 data was not sync'ed. That is, when the comport was read, the first A/D data word was not from channel 0.
CANT_RESYNC1	12	Attempt to re-sync comport 1 failed. Reading up to 8 consecutive data words from it did not turn up one with the expected channel number.
SYNC_ERROR5	14	The comport 5 data was not sync'ed. That is, when the comport was read, the first A/D data word was not from channel 0.
CANT_RESYNC5	13	Attempt to re-sync comport 5 failed. Reading up to 8 consecutive data words from it did not turn up one with the expected channel number.
BAD_SET_POS_CMD	15	The parameters in a SET_POSITION command received from the host do not make sense. They have a bad encoder ID or a bad cycle count.
AZ_AMPLITUDE (sine saturation)	16	Amplitude of sine signal from Azimuth encoder is too large. MAX = 32760
AZ_AMPLITUDE (sine too low)	17	Amplitude of sine signal from Azimuth encoder is too small. Below 16384 in absolute value.
AZ_AMPLITUDE (cosine saturating)	18	Amplitude of cosine signal from Azimuth encoder is too saturating at 32760 or more.

continued on next page

Error Codes Continued

AZ_AMPLITUDE (cosine low)	19	Amplitude of cosine signal is below 16384 absolute value.
ELO_AMPLITUDE (sine saturation)	20	Amplitude of sine signal from Elevation 0 encoder is too large. max = 32760
ELO_AMPLITUDE (sine too low)	21	Amplitude of sine signal from Elevation 0 encoder is too small. Below 16384 absolute value.
ELO_AMPLITUDE (cosine saturating)	22	Cosine amplitude is 32760 or more (absolute value)
ELO_AMPLITUDE (cosine too low)	23	Cosine amplitude of Elev. 0 encoder is below 16384 absolute value.
EL1_AMPLITUDE (sine saturating)	24	Amplitude of sine signal from Elevation 1 encoder is too large. Above 32760.
EL1_AMPLITUDE (sine too low)	25	Amplitude of sine signal from Elevation 1 encoder is too too small. Below 16384 absolute value.
EL1_AMPLITUDE (cosine saturation)	26	Amplitude of cosine signal from Elevation 1 encoder is too large. MAX = 32760.
EL1_AMPLITUDE (cosine too low)	27	Amplitude of cosine signal from ELEV. 1 encoder is too low. Below 16384.
MISSED_REPORT	28	Means that the DSP was late for sending a position report. It is supposed to send them every 10ms. This means it was at least 1ms late. Not necessarily a serious error. Depends on how late and how often.
AZ_ELLIPSE_BAD	29	The computed ellipse fit for the azimuth encoder data failed the sanity check. The parameters were outside reasonable limits. The solution was ignored and the DSP continued using the old solution.
ELO_ELLIPSE_BAD	30	The computed ellipse fit for the Elev. 0 encoder data failed the sanity check. The parameters were outside reasonable limits. The solution was ignored and the DSP continued using the old solution
EL1_ELLIPSE_BAD	31	The computed ellipse fit for the Elev. 1 encoder data failed the sanity check. The parameters were outside reasonable limits. The solution was ignored and the DSP continued using the old solution

Host To DSP communication

Below are the codes for the commands sent from the host to the DSP.

```
#define AZ_INDEX_PULSE_SEARCH 0x10 /*enter Azimuth index pulse search mode.*/  
/* Length = 1. No other words in the command are used. */  
  
#define ELEV_INDEX_PULSE_SEARCH 0x11 /*enter elev index pulse search mode.*/  
/* Length = 1. No other words in the command are used. */  
  
#define SET_TIME      0x12 /* Set local sidereal time or UT, etc.. */  
/* Length = 2. The second word is the time in units of 1/CLOCK_RATE sec. */  
  
#define RESET      0x13 /* Flush circular buffers & re-do buf. init. */  
/* Length = 1. No other words are used. */  
  
#define SET_NREPORT      0x14 /* Specify normal report period. */  
/* Length = 2. The second word is the report period in units of  
1/CLOCK_RATE. This is stored in nReport. Every nReport interrupts, the  
foreground computes the telescope position and sends it to the host.  
CLK_RATE = 1000. So, nReport is expected to be in the range of 6 to 10 (6ms to 10ms).  
Setting nReport to 0x7FFF_FFFF effectively disables reporting.  
*/  
  
#define TERMINATE      0x16 /* Exit the main while loop */  
/* Length = 1. No other words used. */  
  
#define SET_CIRC_BUF_SIZE 0x17 /* Specify new size for circ buffers. */  
/* Length = 2. 2nd word is new circ buffer size. */  
  
#define SEND_CORR_REPORT 0x18 /* Host wants correction report sent. length = 1 */  
  
#define SEND_TIME      0x19 /* Host wants report of time (=tickCounter).*/  
/* Length = 1. Used for synchronization of clocks. */  
  
#define RAW_DATA      0x1A /* Command DSP to start sending raw data.*/  
/* Length = 3. Second word is the length of time (an integer number of  
milliseconds) during which to record data. Third word indicates from which  
encoders data is wanted. Low 3 bits are set to indicate which of the 3  
encoders is desired. 1 -> azimuth, 2 -> elev0, 4 -> elev1. The next bit  
bit 3 (=0x8), indicates capture cosine and sine (if it's 0) OR cosine  
and ref. pulse (if it's 1).  
*/  
  
#define MODE_0      0x1b /* Set mode back to 0. */  
/* UNdoes ref. pulse search. Is only 1 word long. */
```

DSP to Host Communication

Below are command codes sent from the DSP to the Host:

```
/* ----- D S P   T Y P E   C O D E S ----- */
/* These are the type codes which go in the low 4 bits of the status word.
   They identify the records that the DSP can send to the Unix host.
*/
#define POSITION_REPORT 0 /* This is a normal position report. See struct posRpt */

#define INIT_REPORT 1 /*Sent once after initialization is done or whenever "GO" is received.*/
                      /* See struct initRpt */

#define CORRECTION_REPORT 2 /* Coefficients computed by ellipse fit. See struct corrRpt */

#define DIAGNOSTIC4 3 /* Sends 4 specified integers to host. See struct diag8 */
#define DIAGNOSTIC8 4 /* Sends 8 specified integers to host. See struct diag4 */
#define DIAGNOSTIC4F 5 /* Sends 4 floating point numbers to host. See struct diag4f */
#define DIAGNOSTIC4H 6 /* Same as DIAGNOSTIC4 but signals Linux to print in hex.*/

#define RAW_DATA_REPORT 7 /* Used when recording raw data to file. See struct rawDataRpt */
#define TIME_REPORT 8 /* A no-op. Just a keep alive report. See struct timeRpt */
#define RAW_DATA_END_REPORT 9 /* raw data recording is ending. See struct rawDataEnd */

#define REF_PULSE_REPORT 10 /* A ref. pulse was detected. See struct refPulseRpt */
```

See the track.h and servoTrack.h for details of the structure of each command.