A Bluetooth Telescope Controller

V 2.0 – October 2010

Contents

1 Introduction

These days, any off-the-shelf telescope with some self-esteem has a goto controller. Any Amateur Telescope Maker with as much self esteem would like to build such a controller himself. To realize this, he can base his project on a worked-out possibility available on the web. One of these systems is the wideely used design of Mel Bartels, commercialized by Dan Gray of Sidereal Technology. A somewhat less used alternative is the system designed by Martin Cibulski, which has also been built by some SSA members.

The two designs are quite different. The Bartels system consists of a relatively simple driver circuit and relies on PC software for controller algorithms. In contrast, the Cibulski system is self-contained, and is more like the systems integrated in the off the shelf goto scopes.

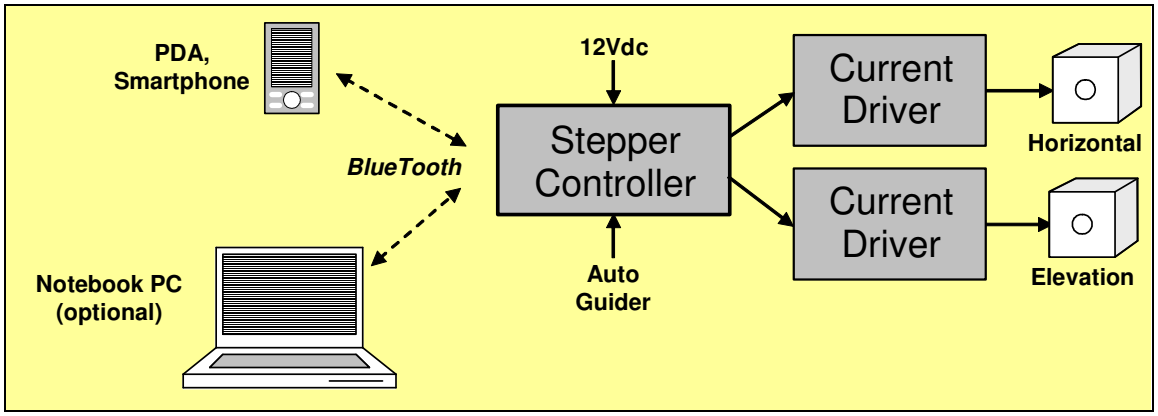
The Bartels design has the advantage that it is fairly simple to build, and anybody can install PC software. The main disadvantage is that you always need a PC, and some big battery to supply it in the field. The Cibulski system is a bit more complicated, it is based on an Atmel microcontroller which needs to be programmed, and has a more complicated circuit. It also requires a proprietary hand paddle for control. Main advantage is of course that no additional equipment is required, and that it is quite low in power consumption.

The goal I set when starting the design that follows, was to have a more or less self contained system that could be controlled by using a de-facto standard protocol like use don the Meade LX-200. This opens up the possibility to have the telescope controlled from a planetarium program, running on a PC or a PDA (more convenient for in-the-field use). On the business side it is designed to be able to control a variety of power drivers.

The following is written as an analysis of the background math but also to capture the whole reasoning behind the design. Maybe it is of use to someone attempting a similar project.

1.1 System Overview

As can be seen in the overview shown below, the system consists of three parts: a controller and two current drivers. The controller takes care of user interface, coordinate transformations and sending out pulses to advance the steppers. The current drivers convert the step pulses into the corresponding current to drive the actual stepper motors.



System Overview

The goal of separating controller and driver is to decouple the implementation of the controller form the actual AC needs of the stepper motors. The driver can be optimized for

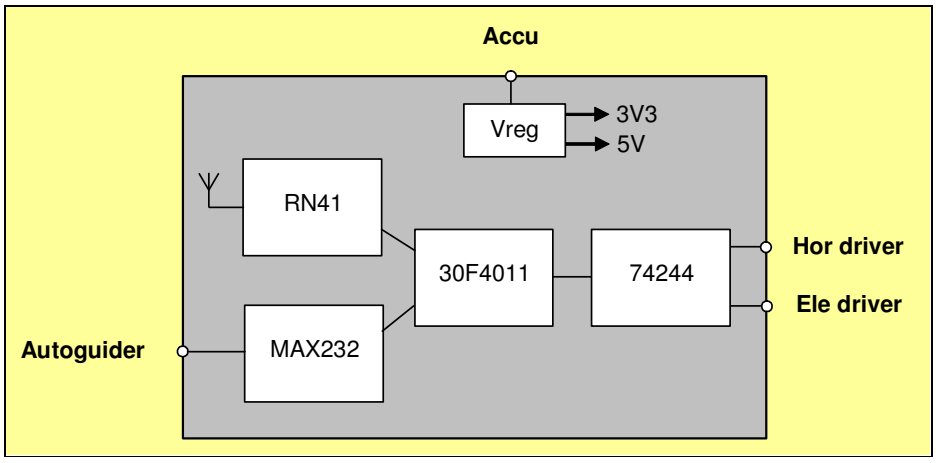
the steppers selected to move the telescope. Stepper drivers which have a step-pulse control interface are commercially available from various sources.

The controller has two serial interfaces. The first runs over a Bluetooth link, using the Serial Port Profile (SPP). This interface is used to link to external applications by means of the LX-200 control protocol. The second serial interface is wire based, and can be used for autoguider or possibly a hand paddle.

The controller is powered by for example a car battery. The current drivers will have their own supply which depends on the selected units (probably also car battery).

1.2 The Hardware

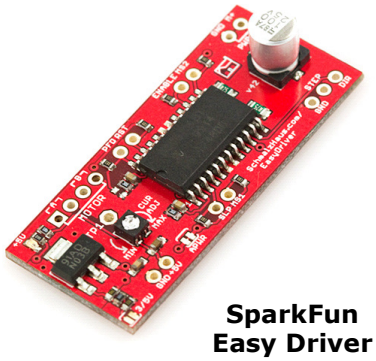
The controller is built around a Microchip dsPIC 30F4011 digital signal controller. This controller has built-in DSP functionality, such as a MAC (Multiply and ACcumulate). Such a MAC provides a single cycle fixed-point mutiply instruction, which is good for coordinate transformations.



Block diagram

The Bluetooth interface is implemented with an RN41 module by Roving Networks. This module provides a serial port to the PIC, and requires only a few simple controls. The wired serial interface is implemented with a MAX232 compatible level shifter, and the interface towards the current drivers is using a 74244 TTL transceiver.

The plan is to use a pair of SparkFun 'Easy Driver' modules, which are based on an Allegro A3967 driver chip. This chip can drive steppers between 7..30V with a max current of 750mA. An alternative for more demanding stepper motors could be an M415C, obtainable from www.stappenmotor.nl. Another possibility is to roll your own, for example base don the new L6470 by ST-Microelectronics.



SparkFun Easy Driver

1.3 The Software

The 30F4011 DSC contains the software that basically converts LX-200 commands into pulses for the steppers. This sounds simpler than it is in reality: the command "go to this RA-Dec coordinate" requires these equatorial coordinates to be transformed into telescope based Hor-Ele reference system, which continuously change with time. Then these target Hor-Ele coordinates need to be compared with the actual values, and a software control loop is running to close the gap. The control loop finally pulses the steppers to drive the telescope to the required direction.

Eventually the controller will reach equilibrium and end up in tracking mode. In this mode only the movement caused by the rotation of the earth has to be compensated in order to remain locked on the set target RA-Dec coordinate. This tracking mode can be supported by an autoguider, which will correct the direction by directly directly adding or subtracting steps from the current Hor-Ele setting. Every now and then this should also be processed in the coordinate transformation algorithm.

The transfomation uses a set of parameters that has to be initialized in a calibration procedure. This procedure couples known star positions to measured stepper settings (or Hor-Ele coordinates) to derive the parameters. Two fixes are enough, but more is better. Other parameters are telescope related, such as number of steps per revolution and max acceleration, and these can be configured once and stored persistently.

The external controlling application can be any planetarium program that speaks LX-200 commands. Examples of such programs are "Cartes de Ciel" freeware, or "TheSky" by Software Bisque. Alternatively a simple PDA or PC application can be made to have some kind of hand paddle function.

2 Algorithms

This section provides more detailed mathematical background of the algorithms used in the telescope controller. I have collected this background material in order to be able to write the controller software.

2.1 Dynamic range

There is an enormous difference in speed between just tracking an object and slewing to an other object as a consequence of a GoTo command. While tracking only the rotation of the earth has to be compensated, which goes at a maximum of 15”/sec (from west to east).

For slewing a much higher speed is required so it doesn’t take all night to for example sweep from Capella to Albireo. A more realistic slewing speed would be in the order of 1 or 2 º/sec.

The angular resolution of the scope drive should be below the resolving power of the telescope, or maybe more realistically, of the average seeing. A good value is about 0.2” and this value will determine the angular motion associated with a single (micro)step of the motor.

Taking these values into account, the stepping frequency will range from about 30 Hz for tracking to about 20kHz for slewing. The resulting dynamic range required to achieve these speed values is a factor 1000.

The maximum stepping speed a motor can handle without skipping steps, differs between the various types but also depends on how the motor is mechanically loaded. Practical values are around 1000 full steps per second, so with a factor 16 microstepping about 16kHz will be the maximum achievable angular speed. Microstepping in itself also makes motor operation more reliable, and might even allow higher frquencies.

A calculated example:

Stepper resolution:	1.8º (200 step/rev.)
Microsteps:	16
Telescope resolutie:	0.2”
Required reduction:	(3600 * 1.8) / (16 * 0.2) = 2025

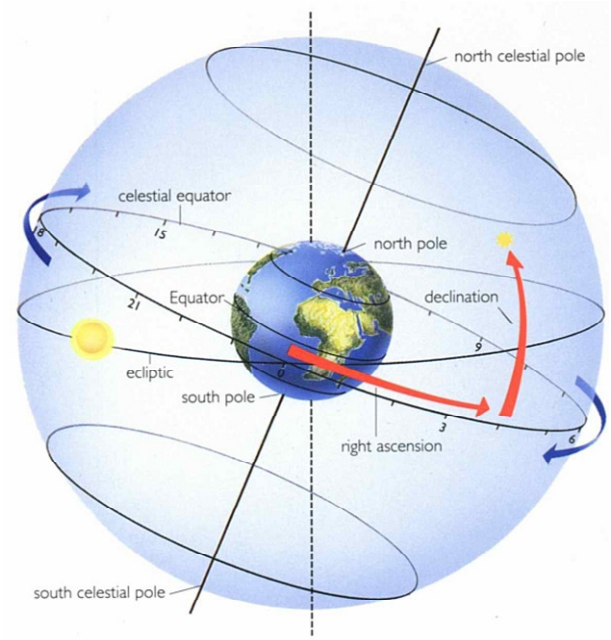
When instead 64 microsteps are used, the required mechanical reduction is only 506. There is however a downside of cranking up the number of microsteps: the accuracy of a single step will suffer from non-linearity inherent to the motor design, which will result in periodic error. This non-linearity can to certain extent be compensated in software, but it is better to use a current driver that is capable of correcting the driving current to compensate for the non-linearity. This will probably require some calibration…

2.2 Coordinate systems

The coordinates of a celestial body are given in Right-Ascension and Declination. These coordinates fix a certain point on the celestial globe, and for remote stars (their yearly parallax is zero) the RA-Dec coordinates are constant. This system is called “Equatorial”, since the line of zero Declination is defined by the projection of the earth equator on the celestial sphere. In other words, it is referenced on the Earth itself.

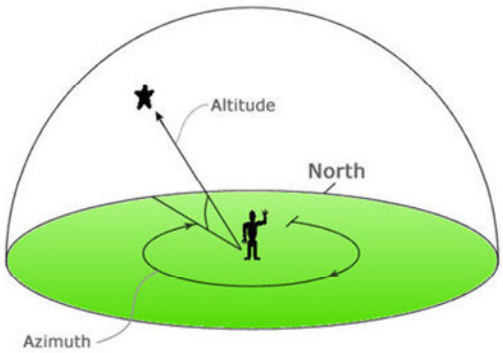
The Right Ascension is the angle along the celestial equator measured from the crossing of

equator and ecliptic. This is the RA the Sun has at the beginning of spring, the vernal equinox. Declination is the angle measured perpendicular to the celestial equator.



Equatorial coordinate system

In reality we are located somewher on the Earth, which rotates about its axis once every 24 hours. Furthermore, the Earth revolves around the Sun in about 365.25 days. This yearly revolution around the Sun silently adds another Earth rotation to the number of days, when not the Sun but some remote star is taken as reference. The result of this is that we see the yearly motion of constellations when we watch the heavens the same time every evening.



Horizontal coordinate system

So, if we want to refer e certain celestial object to our viewing location, we need to know exactly where we are, and what time and date it is. The coordinate system that is centered on the observers' location is called Horizontal (or Alt-Azimuth). This system is much easier

to comprehend than the Equatorial system: the angles that are similar to the equatorial RA and Dec are called Azimuth (Az, angle along the horizon) and Altitude (Alt, vertical angle perpendicular to the horizon) respectively.

So a solar day is defined by successive culminations of the Sun. If we reference the Earth rotation to the celestial sphere instead of the Sun, we call it a Sidereal (star). A sidereal day is the time between culminations of a fixed celestial object. A sidereal day is somewhat shorter than a solar day, because the daily bit of Earth orbital motion (i.e. 360º/365.25) must be subtracted from a solar day to obtain a sidereal rotation. The ratio between solar and sidereal time is approximately 1.002737909350795, and hence a sidereal day lasts only 23:56:04.

This difference in length of day causes the sky to apparently move westward through the year.

A third coordinate system that should be considered is that centered on the telescope. This resembles the Horizontal system, but it takes into account that the telescope may not be level and may also have some mechanical alignment errors. The transformations that will be used in the controller are between sky based Equatorial (RA-Dec) and the telescope based Horizontal-Elevation system (Hor-Ele).

2.3 Transformations

The relation between Equatorial and Horizontal coordinate systems is analytic, and has as parameters the observers' latitude and the local sidereal time (LST). The LST is directly derived from the Greenwich Mean Sidereal Time (GMST) by adding an offset determined by the observers' longitude. GMST can be directly derived from UTC.

The transformation f between the coordinate systems can be given as:

$$(Az \quad Alt) = f_{LAT,LST}(RA \quad Dec)$$

And the reverse:

$$(RA \quad Dec) = f^{-1}_{LAT,LST}(Az \quad Alt)$$

In practise, we're not usually not interested in our real LST and Latitude. Assuming that the telescope has no errors, misalignment to the Horizontal coordinate system could be ignored by defining 'apparent LST' and 'apparent Latitude'. Tilt in east-west direction is compensated by a an offset LST, while tilt in north-south direction is compensated by an offset Latitude.

Disadvantages of this simplification are a less accurate correction of refraction near the horizon, and also compensation of location-dependent parallax effect noticable when observing relatively nearby objects (satellites, moon, planets).

A more precise method of making the transformation is based on transformation matrices. In this method telescope mount construction errors can be also be accounted for. The method is very well described by Toshimi Taki on his website, and is for example used in the software of Mel Bartels' system. A disadvantage of this method is the increased complexity that requires quite some processing power. For a microcontroller it may be too demanding.

In practical systems a combination of both methods can be useful. If the system is started up, pointing at approximately Alt=0 and Az=0 and knowing LST and Latitude, a first order approximation can be used to more easily find the calibration stars.

2.3.1 Horizontal to Equatorial and reverse

First define a set of parameters:

δ: Declination (-90º .. 90º)
α : Right Ascension (0 .. 24h)
h : Altitude above horizon (0 .. 90º)
A : Azimuth eastward from north (0 .. 360º)
μ : Hour angle (0 .. 24h)
φ : Local Latitude (-90º .. 90º)
Ψ : Local Longitude (-180º .. 180º, East of Greenwich is positive)
LST: Local Siderel Time
GMST: Greenwich Mean Sidereal Time
DUTC: Decimal days since 1 Jan 2000, 12:00 (UTC)

The following formulae define the transformations:

$$\mu = \left\{ \frac{1}{15} * \arctan\left(\frac{\sin(A) * \cos(h)}{\cos(A) * \cos(h) * \sin(\varphi) - \sin(h) * \cos(\varphi)} \right) \right\} \bmod 24$$

$$\delta = \arcsin\left(\sin(h) * \sin(\varphi) + \cos(A) * \cos(h) * \cos(\varphi)\right)$$

Right Ascension and Hour angle have a simple relation:

$$\alpha = LST - \mu$$

Note that the Azimuthal coordinate system has a fixed relation with the Hour angle, whereas the relation with Right Ascention changes with time.
The Local Sidereal Time (*LST*) can be calculated as follows:

$$LST = \left(GMST + {}^{\Psi}\!/_{15} \right) \bmod 24$$

The Greenwich Mean Sidereal Time is accurately related to UTC as follows:

$$GMST = \left(18.697374558 + 24.06570982441908 * D_{UTC}\right)$$

Calibration of the controller software now boils down to relating step counts and Alt-Az coordinates. To make this more linear, count values of 0 could be shifted to align with coordinate values of 0.

For the reverse transformations similar formulae are derived:

$$h = \arcsin\left(\cos(15 * \mu) * \cos(\delta) * \cos(\varphi) - \sin(\delta) * \cos(\varphi)\right)$$

$$x = \arccos\left(\frac{\sin(\delta) - \sin(h) * \sin(\varphi)}{\cos(h) * \cos(\varphi)} \right)$$

$$(LST - \alpha) \geq 0 \Rightarrow A = 360 - x \qquad (LST - \alpha) < 0 \Rightarrow A = x$$

2.3.2 The matrix method

The matrix method can be used for any linear transformation, of which Alt-Az to RA-Dec as described in teh previous section is just a special case. To calibrate the transformation matrix, at least two reference points are needed, i.e. two sets of measurements relating a reference star to telescope coordinates.

The matrix works between cartesian or orthogonal spaces. Because both celestial and telescope coordinates are spherical, these first have to be converted to so-called direction vectors. These are 3 dimensional vectors with unit length, marking the direction of a star or the telescope.

First define some variables:

δ: Declination (-90º .. +90º)

α : Right Ascension (0 .. 24h)

μ : Hour angle (0 .. 24h)

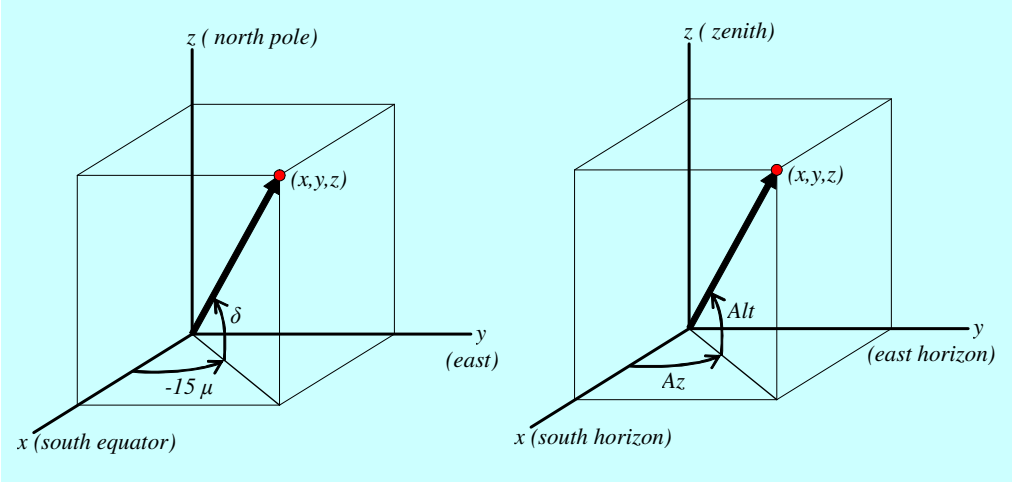
ε: Elevation (0 .. 90º)

θ: Horizon angle (0 .. 360º, eastward from south)

k: 1.002737908, Conversion factor from sidereal to solar time

t, t0: time and reference time (e.g. start of session)

Again, Right Ascension and Hour angle are related by: $\alpha = LST - \mu$



Equatorial and Horizontal direction vectors

The cartesian direction vectors are given by:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos(\delta)\cdot\cos(-15\cdot\mu) \\ \cos(\delta)\cdot\sin(-15\cdot\mu) \\ \sin(\delta) \end{pmatrix} \quad \text{en} \quad \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \cos(\varepsilon)\cdot\cos(\theta) \\ \cos(\varepsilon)\cdot\sin(\theta) \\ \sin(\varepsilon) \end{pmatrix} \qquad \text{(1a, 1b)}$$

Reverse operation for equatorial coordinates:

$$-15\cdot\mu = \arctan\left(\frac{y}{x}\right) \quad \text{en} \quad \delta = \arcsin(z) \qquad (2)$$

Note that the time of observation is a parameter through the hour angle, but instead any time reference could be taken.

For telescope coordinates the reverse operation is:

$$\theta = \arctan\left(\frac{m}{l}\right) \quad \text{and} \quad \varepsilon = \arcsin(n) \tag{3}$$

For both $\arctan()$ functions, when x (or l) < 0, a shift to the 2nd or 3rd quadrant must be done by means of an 180º rotation

The conversions of direction vectors from one coordinate system to the other are defined by the following transformations:

$$\begin{pmatrix} l \\ m \\ n \end{pmatrix} = T \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} = T^{-1} \cdot \begin{pmatrix} l \\ m \\ n \end{pmatrix} \tag{4a, 4b}$$

Ideally, the transformation T is described by a rotation matrix. In that case the inverse T^{-1} is equal to its transposed. This means that it can be found by mirroring in the t_{ii} diagonal, or in other words: t_{ij} -> t_{ji}. In practise the inverse must be determined by full calculation, because the transformation is likely not ideal due to misalignment errors.

The matrix T can be deduced from two or three calibration sets, stored as:

Name	Time	RA	Dec	Hor	Ele
Star 1	t_1	α_1	δ_1	θ_1	ε_1
Star 2	t_2	α_2	δ_2	θ_2	ε_2
Star 3	t_3	α_3	δ_3	θ_3	ε_3

From the three calibration measurements first the 6 direction vectors must be calculated: 3x for the equatorial and 3x for the telescope system. For each of the three sets the transformation should work, and hence the matrix can be deduced:

$$\begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{pmatrix} = T \cdot \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \quad \text{and:} \quad T = \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix}^{-1} \tag{5}$$

The required matrix operations are standard linear algebra...

The link between (x,y,z) and (l,m,n) vectors is the time of observation. The (l,m,n) vectors are derived from the step counters in the controller, maximum stepcount equals 360º.

In practise, a calibration will go through the following steps:

- Measure the positions of two or three known stars
- Calculate direction vectors with (1)
- Derive transformation matrix with (5)
- Determine the inverse transformation matrix

Then the matrices can be used to do the actual transformations:

- Determine the direction vector with (1a) or (1b)
- Perform the transformation with (4a) or (4b)
- Determine the resulting coordinates with (3) or (2)

2.4 Simulation

In order to try out the transformation algorithms a PC application was made to simulate the behaviour of an LX-200. This simulator is used in combination with 'TheSky PE' running on a PDA (HTC P3300, Windoze Mobile 5). The PDA application sends LX-200 commands to the simulator, which should react as a real LX-200 compatible telescope. The behaviour that is reverse-engineered with this simulator will finally be used to implement the software in the controller.

One of the first things that is noted, is that 'TheSky PE' continuously requests the current RA and Dec values of the telescope. The returned values are made visible as a cross-hair on the PDA screen. As long as the telescope is tracking, these coordinates will not change. When the simulated telescope is moved to another coordinate, you can also see the crosshair move over the sky background.

The simulator always starts on the equator (Dec = 0) and the local meridian (RA = LST). Using the simulators N, E, S and W buttons, the telescope direction can be changed. The speed that is used is determined by pressing one of the Slew, Find, Center and Guide buttons. Speeds are in accordance with LX-200, respectively 2º/s , 1º/s , 8'/s and 15"/s.

The actual direction of the simulated telescope does not have to be what is reported to TheSky. To calibrate, the telescope can be moved using the direction buttons to simulate pointing to a known star. When this star is selected in TheSky, and the 'sync' command is selected, the right RA/Dec coordinates are passed to the simulator, which can use these to do a calibration (or at least store a measurement for later calibration).

The idea is to implement and test all algorithms in the simulator, before actual implementation in the controller. This allows for much easier debugging.

LX-200 Simulation

The following interactions were linked to various operations in TheSky-PE:

Menu->Telescope->Initialize

Command	Reply	Comment
:Sg 354*50#	1	Set site longitude (ex: 5*9.1' east)
:St +52*13#	1	Set site latitude (ex: 52*13' north)
:SL 22:08:05#	1	Set local time
:SG -01#	1	Set hours to UTC
:SC 02/28/10#	1Updating##	Set local date (ex. feb 28, 2010), 30sec for update

Point on chart->Sync button

Command	Reply	Comment
:Sr 11:14:14#	1	Set RA of object
:Sd -00*55:43#	1	Set Dec of object
:CM#	Object#	Synchronize telescope with target object

Arrow key or equivalent. (rate depends on Menu->Telescope->Set Slew Rate)

Command	Reply	Comment
:RS# *or* :RM# *or* :RC# *or* :RG#		Set slew rate to Max *or* Set Slew rate to Find *or* Set Slew rate to Centering *or* Set Slew rate to Guiding
:Mn# *or* :Me# *or* :Ms# *or* :Mw#		Move North (up) *or* Move East (left) *or* Move South (down) *or* Move West (right)
:Q# :Q#		Stop slewing STOP! (just in case…)

Select object (or point on chart)->Slew button

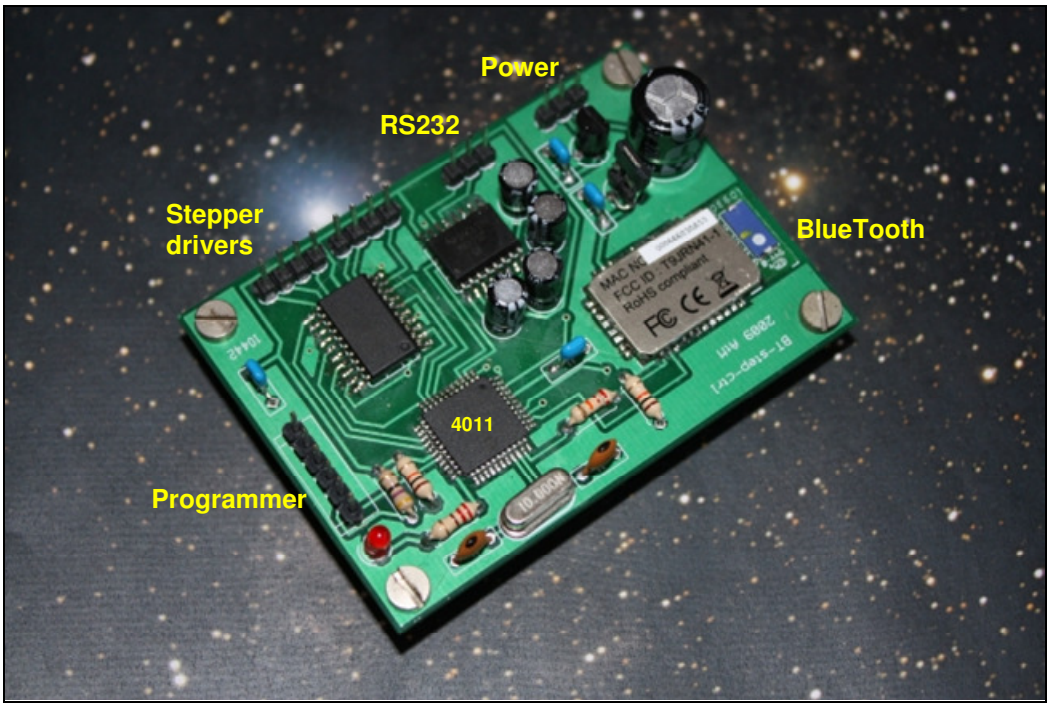
Command	Reply	Comment	
:Sr 18:21:53#	1	Set RA of object	
:Sd -02*53:30#	1	Set Dec of object	
:MS#	0 or 1 or 2	Slew to target object	OK Object too low (below horizon) Object too high to reach

Menu->Telescope->Abort slew

Command	Reply	Comment	
:Q# :Q# :Q# :Q#		Stop slewing Stop! STOP! S T O P !	 (in case you've missed it) (just in case…) (to be absolutely sure)

3 The prototype

A prototype of the controller has been built according to the the plan described in section 1. The image below shows this prototype and indicates the interfaces and the PIC DSC.

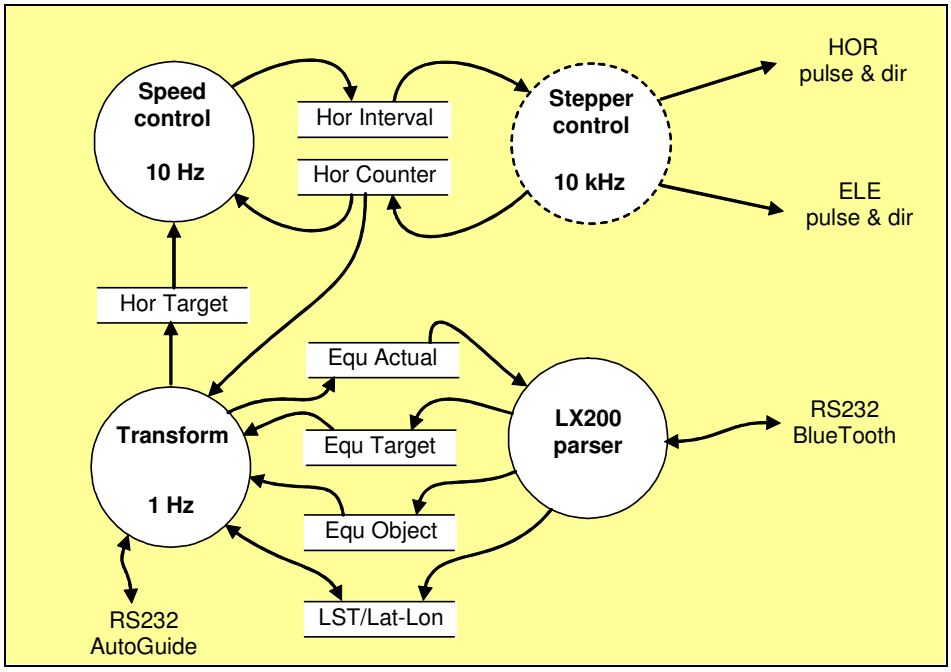


Controller Prototype

The total cost of this module is about €50, the RN41 Bluetooth module being the cost driver. As can be seen, with a little effort it would be possible to squeeze on som more components or to shrink the PCB. The final design could also increase the use of SMT for passive components. All in all, the current prototype is foremost meant as a platform for experimentation and software development. The main challenge will be to squeeze it all in the tiny memory, measuring 48kB of Flash and 2kB of RAM. Furthermore, the clock speed is quite low, only 40MHz with a non-pipelined instruction rate of 10MHz.

3.1 Controller Software

The most demanding task for the controller will be the generation of stepper pulses. This happens in an interrupt servioce routine (ISR) triggered by a hardware timer. Considering the instruction clock of 10MHz and a 10kHz step pulse frequency, there are about 500 instructions between timer interrupts. The timer interval is 50 µsec (instead of 100) because each pulse has a rising and a falling edge. This is not a lot, but luckily the ISR task is very simple, counter update a conditional toggle of the pulse bit.



Software overview

The software structure is as shown in the above diagram. There are three operational tasks that run concurrently, and optionally (not drawn) a fourth monitoring the RS232 guide-port for configuration commands or autoguider input.

Operational tasks:

- LX200 interpreter: Interprets the LX-200 control protocol over the Bluetooth interface to PC or PDA. This task also executes the calibration functions, which are in fact initiated by some of the LX-200 commands. It is the least time critical task, so it runs at lowest priority.
- Transform: This task calculates every (sidereal) second what the Target Horizontal coordinates on the next tick should be. This calculation relies on the availability of a valid transformation matrix. It converts the Target Equatorial coordinate input from the LX200 task and the current LST into Target Horizontal coordinate.
- Speed Control: This task calculates 10 times per second what the stepper speeds should be, based on the difference between Actual and Target Horizontal coordinates and the current speed. The calculation takes into account the acceleration and speed limits. It has a higher priority than the Transform task.

Finally, the Stepper Control task is implemented as an Interrupt Service Routine (ISR) because it has a very strict timing requirement. Since it is an ISR it automatically gets higher priority than any regular task.

3.1.1 LX200 parser

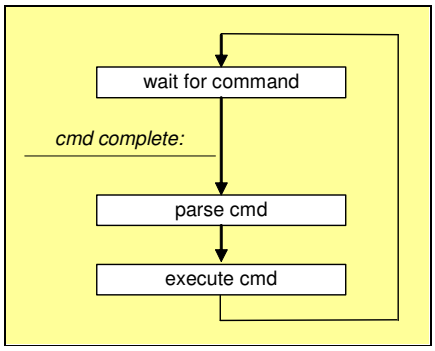
The main function of this task is to maintain the communication link with the controlling PC or PDA application. This communication uses the Meade LX-200 commandset, which can be considered as a de-facto standard.

This protocol mainly uses Equatorial coordinates, since this is the frame of reference for

most planetarium programs and astronomic starcatalogs. Three coordinates are used, indicating the Target and Object from commands and the Actual coordinate for replies. The first two are used to direct the telescope to a certain Target direction and to store an Object for later referral. The third coordinate is used to report the current Actual direction, which can be represented by a crosshair on the planetarium sky map.

Other parameters that can be set by the protocol are a set of predefined speeds, observing site location and time, and celestial coordinates for calibration purposes.

Communication with the Xform task happens through shared memory locations, as shown in the Software overview diagram.



LX200 task

The LX200 task blocks on incoming data received from the Bluetooth link. When a command is complete it is interpreted and executed. For some commands the result is reported back. As long as no commands are received, the task is in idle state and uses no processing resources.

An additional function of the LX200 task is to supervise the initialisation and calibration of the transformation matrices. This is in fact driven by the commands from the PC or PDA.

Initially, the controller assumes to be pointing at the south horizon. As soon as it receives site location and time information a rough initialisation is done. From this point on, the controller can be managed with the planetarium program.

To achieve better accuracy, the controller also needs to be calibrated. Three synchronisation commands are required to be able to do a proper calibration. The sync command in fact means that the previously set Equ Object coordinates (in hour angle format) are the transformation of the Hor Actual coordinate, and the set is stored as a calibration measurement. Three of such measurements are used to derive the transformation matrix.

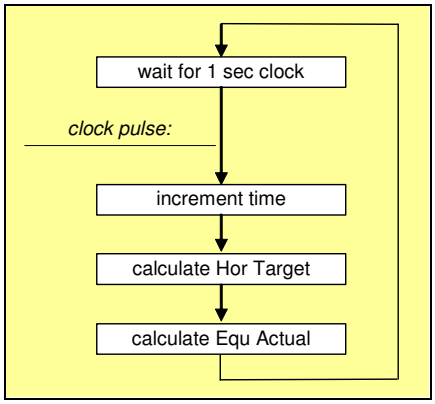
The various command sequences have been reverse engineered with the simulator, as described in section 2.4.

The effect of various commands on the processing is summarized in the table below:

Command	Reply	Action
:Sg 354*50#	1	Store site longitude in *Lat-Lon*
:St +52*13#	1	Store site latitude in *Lat-Lon*
:SL 22:08:05#	1	Store site time in *LocTime*
:SG -01#	1	Store offset to UTC in *LocTime*
:SC 02/28/10#	1Updating##	Store site date in *LocTime*, Calculate *LST*, Initialize xforms, based on defined Horizontal and site
:Sr 18:21:53#	1	Store object RA in *RA-Dec Obj*
:Sd -02*53:30#	1	Store object Dec in *RA-Dec Obj*
:MS#	0	Copy *RA-Dec Obj* to *RA-Dec Tar*, This will start slewing
:CM#	Object#	Save calibration set: {*RA-Dec Obj*, *LST*, *Hor-Ele Act*} If sufficient data, calibrate xforms
:RS#, :RM#, :RC#, :RG#		Set *SlewRate* to Max, Find, Centering, Guiding
:Mn#, :Me# :Ms#, :Mw#		Copy *SlewRate* to *DeltaDec*, *DeltaRA* Copy *-SlewRate* to *DeltaDec*, *DeltaRA*
:Q#		Copy 0 to *DeltaDec* and *DeltaRA*

3.1.2 Xform

Each (sidereal) second this task calculates the new Hor-Ele coordinates from the Target RA-Dec, as they are valid on the next second. Note that the Target may be different from the Actual RA-Dec, in which case the telescope is in fact slewing to a new direction. This implies that every second a new transformation must be done, using the transformation matrix.



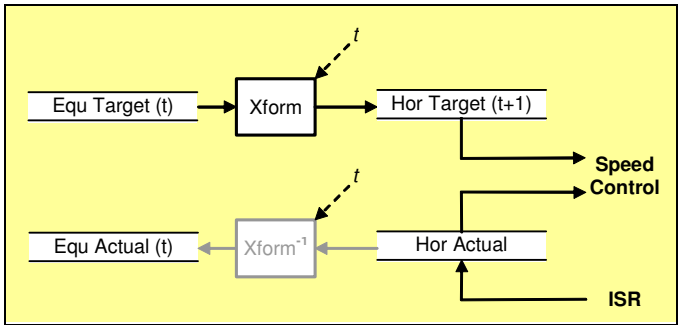
Xform task

The Xform task also takes care of updating the local solar and sidereal time, and feeds the Actual Horizontal coordinate back into the Actual Equatorial. This implies that every time an inverse coordinate transformation must be done as well! This step could be skipped

however, if the Target and Actual Equatorial coordinates have the same value, since in that case the telescope is only tracking the object.

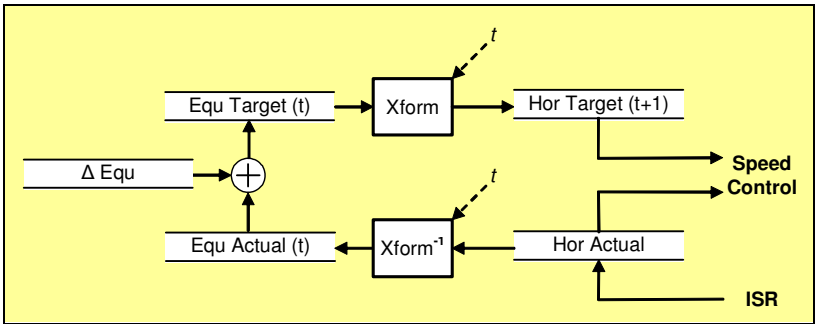
So the Xform task can be in three different modes:

- 'tracking', following the Equ Actual, should be equal to Equ Target
- 'moving', making corrections to the current direction, i.e. Equ Actual
- 'slewing', changing the Equ Actual to Equ Target, as fast as possible



Xform task in 'tracking' mode

So in 'tracking' mode all that needs to be done is convert the Equ Actual (=Equ Target) into the Hor Target, one second in the future. This process and all involved variables are shown in above diagram. In this mode the inverse transform is optional, since the Equ Actual should not change.

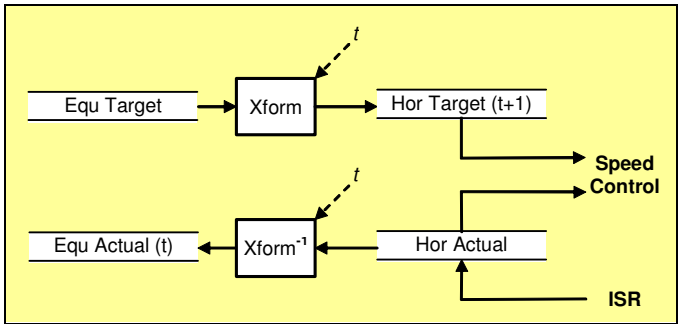


Xform task in 'moving' mode

When the Xform task is in 'moving' mode, an offset needs to be added to the Equ Actual each second. The magnitude of this offset is usually set by an LX-200 command before the actual move command is issued.

Another source of offset is the input from an autoguider. The magnitude in this case is probably better configured for the telescope, default guide speed is 15"/sec, but this may give too large deviations for photographic use. Alternatively the guide commands could be implemented to directly change the SpeedControl task settings.

An inverse transformation is required, since the Equ Actual is expected to change.

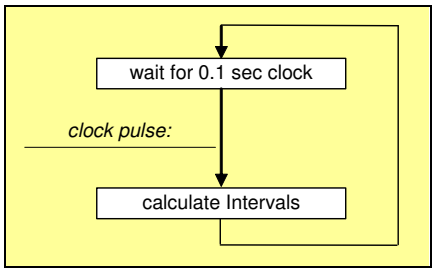


Xform task in 'tracking' mode

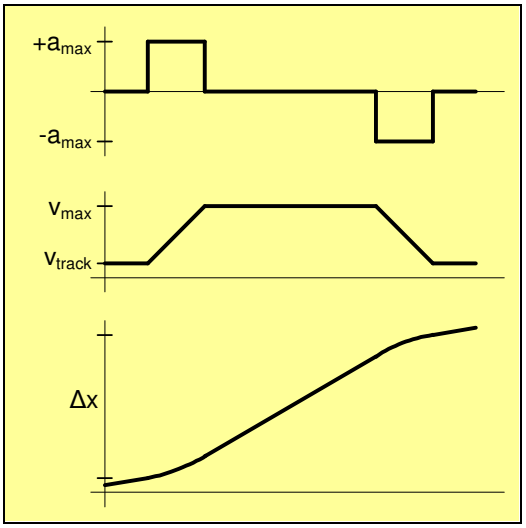
In 'slewing' mode, the telescope needs to be moved from the Equ Actual to the set Equ Target with the highest possible speed. The new Hor Target at t+1 is bluntly calculated from the Equ Target, as in tracking mode, but the difference with the Hor Actual will in this case be rather large. The SpeedControl task takes care of all the dynamics involved.

3.1.3 SpeedControl

The SpeedContol task controls the speed of the stepper motors. This task in fact determines the complete dynamic behaviour of the controller, based on telescope parameters and the Actual and Target Horizontal coordinates. The evaluation is done about 10 times per second. All calculations take place in the telescope coordinate system, so no time consuming transformations are required. The output of the evaluation are Interval settings for the Stepper Control ISR, i.e. the number of interrupts to wait until pulsing a stepper.



The SpeedControl task needs to ensure that acceleration and speed remain within the limits that are defined for the driven telescope/mounting system. When a slew is started it is tempting to go to maximum speed directly, but the forces required to start the telescope mass moving would be enormous. Also this would result in missing steps, which has a bad effect on accuracy. For this reason a gradual acceleration must be ensured by configuring a limiting parameter. The same applies to maximum speed, although this will probably be determined by the driving controller and the steppers rather than the telescope.



The diagram above shows the relation between acceleration (a), speed (v) and displacement (x). As can be seen, the acceleration behaviour is a straightforward on/off process: we're either accelerating (+/-A_{max}) or we're not. As long as acceleration is on, the speed changes, but up to a maximum (v_{max}). When acceleration is off, speed remains constant. When the speed is constant, the rate of displacement is also constant, i.e. a straight (usually slanting) line.

The target speed is determined by the difference between Actual and Target Hor-Ele coordinates. As long as the target (or maximum) speed (in either direction) is not reached, there is an acceleration.

When the target position is nearing, deceleration should kick in. The way this is done is carefully designed, so that the Target will be reached smoothly and overshoot is prevented. To achieve this, usually a PID control algorithm is applied (PID: Proportional-Integral-Derivative). For most stepper systems no independent position feedback is used and control is done by means of an open loop algorithm. Still this algorithm uses the actual proportional (speed), integral (position) and derivative (acceleration) information to derive the control settings.

Each Xform interval (1 sidereal second) is divided in a fixed number of slots. Each slot the SpeedControl task runs once and hence this is considered the unit of time. Speed is defined as number of steps per slot, an acceleration the speed variation per slot. Every slot, the SpeedControl task determines the speed for the coming slot, and then sets the ISR counter intervals accordingly. It is the determination of the new speed setting that is the actual control algorithm.

For the prototype, the following parameters apply:

Xform interval		1000	ticks
SpeedControl slot		100	ticks
ISR rate		10000	per second
		1000	per slot
Max speed		5000	steps/second
	$\boldsymbol{V_{max}}$	500	steps/slot
Max acceleration	$\boldsymbol{A_{max}}$	50	steps/slot^2

Each slot the algorithm calculates whether the steppers need to accelerate or decelerate. If the controller is in slewing mode, this can be derived analytically from current speed, Max acceleration and number of steps to go.

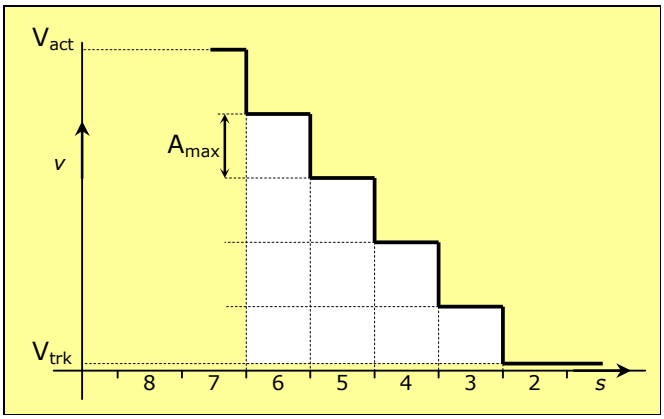
First off, the required speed to reach the target in the remainder of the Xform interval is derived from the number of steps left to reach target, ***Q***:

$$V_{req} = Q / s \quad ,$$

where ***s*** is the number of remaining slots in the current interval.

If the required speed is less than A_{max}, it can be assumed we are either tracking or at most moving or centering. The required speed is then used directly to control the steppers, so it will become the actual speed. This way, corrections are smeared out over an Xform interval.

If $V_{req} > A_{max}$, it can be assumed the controller is in slewing mode and thus different rules apply. First off, it needs to be evaluated whether a deceleration should be initiated in order to reach the target position at the same time as being back to tracking speed.



Steps during deceleration

In the graph above the speed ***v*** is shown as a function of slot counter ***s***, during a deceleration phase. The tracking speed V_{trk} is derived from the increment number of steps in target position, between successive Xform ticks. It is close to 0, when compared to A_{max}, a fact which can be used in the approximations.

The number of deceleration stages is:

$$n = V_{act} / A_{\max}$$

The number of steps to be made in the deceleration phase equals the shaded area:

$$N = A_{\max} \cdot \sum_{i=1}^{n-1} i = \tfrac{1}{2}\left(n^2 - n\right) \cdot A_{\max}$$

If the number of steps left to target position ***Q*** is approximately equal to or smaller than ***N***, the deceleration should be started. If ***Q*** is larger than ***N***, an acceleration should be done.

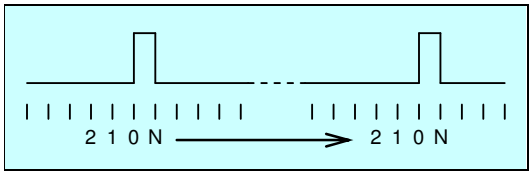
Of course, acceleration can only happen up to the speedlimit, V_{max} , is reached.

The newly calculated speed is passed to the StepperControl function by means of interval

counter settings (Hor Interval). Lower speeds will have higher counter values. The SpeedControl task also takes care of the direction, which is usually a bit value sent to the current driver. The StepperControl ISR then only has to deal with positive counters.

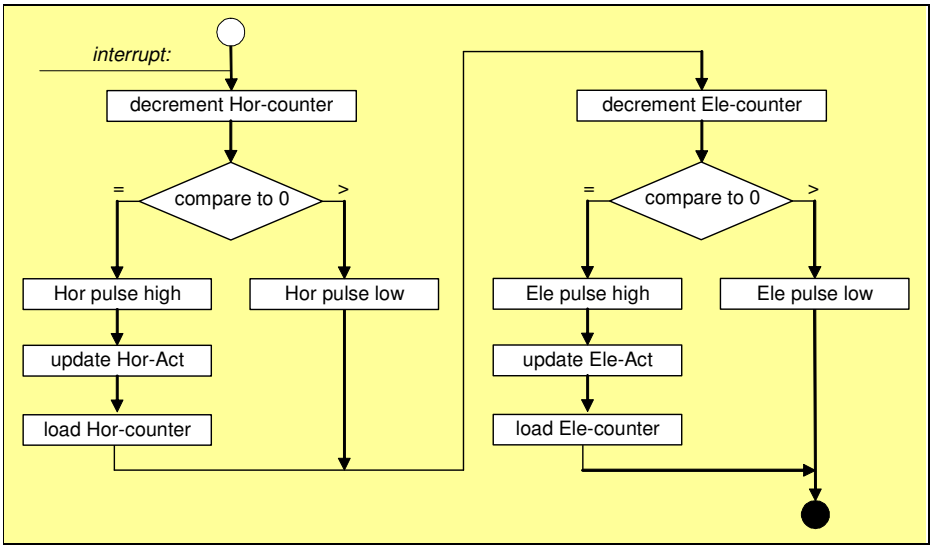
3.1.4 StepperControl

The stepper speed calculated by the SpeedControl task determines the delay that has to be inserted between two successive stepper pulses. Higher speed means lower delay, until half the timer frequency is reached, which defines the maximum speed.



So the delay counters are filled in by the SpeedControl task, and all the high speed ISR needs to do is decrement this counter. When 0 is reached the bit is raised and the counter reloaded, on the next interrupt the bit is lowered again. In fact the low condition is defined by count being any other value than 0.

From this it is clear that the maximum step frequency can be only half the interrupt frequency. In other words, a 100µsec interrupt timer can deliver a stepper frequency of 5000Hz.



StepperControl ISR algorithm

The ISR logic is shown in the diagram above. For each stepper there are only a few instructions to be executed: decrement counter, compare to 0, set/reset pulse bit, set/reset direction bit, reload counter if necessary. Such an ISR does not require any context switching and can be implemented as a fast ISR.

3.1.5 Monitor

In the background, lowest priority, a Monitor task is running. This task listens to the wired RS232 port, for either terminal commands or an autoguider. The port can be used to set the configuration bits and for debugging.

3.1.6 Fixed Point operation

In order to make the various transformations doable on a small controller, floating point operations must be avoided and hence a fixed point library is implemented. A float consists of a mantissa and an exponent portion, comparable with the so-called scientific notation on a calculator.

As an example, the float <0.12345678e04> stands for <1234.5678>. The great advantage of using floats is that they scale very well; the range is in fact determined by the exponent. A big disadvantage is that the accuracy of a float is not so good, so in most cases a double is used instead. This has a double precision, but also double the aount of bits. Also the floating point operations require much more performance than do fixed point variables. For this reason floating point operations are usually done in a co-processor, named floatin pint unit (FPU).

So, for a micro contorller fixed pint operations are the best choice, but these need a bit of planning. An FP variable in fact is an integer with a predefined structure, and this means that FP operations can be don with integer instructions.

In FP variables the word length (e.g. 32 bits) is split in an integer part and a fraction part. The number of fraction bits determines precision, and the number of integer bits determines the range. The split can be tuned for the specific use case.

For the controller a precision of 20 fraction bits is chosen, which boils down to better than 1 part in a million (0.000001). The integer part is 11 bits an dthe remaining bit is used as a sign, as in 2's-complement fashion. The range therefore is -2048 to +2047. The notation for such an FP format is F12.20. For angles this is sufficient, and the precision also suffices since we need about a tenth of an arc second.

Multiplication and division works about like it was teached on primary school. Multiply as integers, and then scale back to the right precision. When two F12.20 numbers are multiplied, it results in an F24.40 number, which should be right shifted by 20 bits and masked off to get back to F12.20. Care must be taken to prevent overflows.

Multiplication of an F12.20 with a straight integer (in fact an F32.0) does not require any shifting. Division has similar operation, but here a pre-shift left is utilized instead of a post-shift right.

The fixed point library contains the operations needed for the transformations:

```
fixmul(y,x);        /* y*x                  */
fixdiv(y,x);        /* y/x                  */
fixsin(y);          /* sin(y)               */
fixcos(y);          /* cos(y)               */
fixatan2(y,x);      /* arctan(y/x)          */
fixasin(y);         /* arcsin(y)            */
fixsqrt(y);         /* √(y)                 */
```

In this set op operations all parameters and results are in a 32bit fixed point format (fp32), where the precision is definable. Furthermore there are various constants defined which can

be used for transformations.

Another optimization is to used "Grads" instead of degrees for indicating angles. It makes table lookup more efficient, and that is exactly what the trig functions use. There are 256 Grads in a right angle (90 degrees).

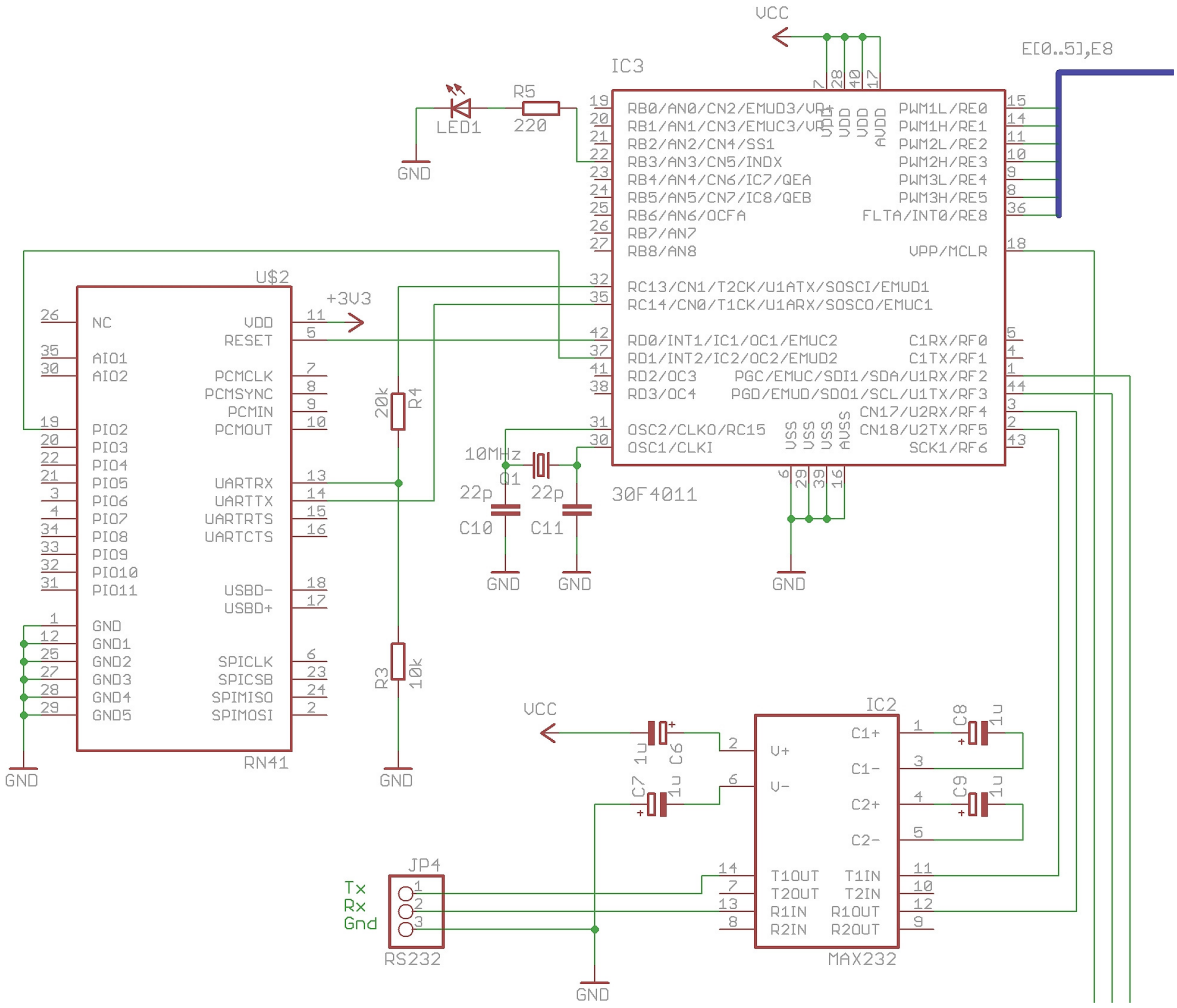
The required linear algebra is implemented in an additional xform library, which uses the above FP functions.

3.1.7 Operating system: "*Juggler*"

Especially for this purpose a real time operating system (RTOS) has been implemented, named ***Juggler***. It is a minimalistic RTOS, which just takes care of multi-tasking, inter-task synchronisation and timing. It has the concept of timeslices, enabling pre-emption of long runing tasks. However, in general at each interrupt or system call a reschedule is done that will result in a task switch when a higher prio task is ready to run. The ***Juggler*** implementation has very low overhead and is made in such a way that it could easily be ported to other processors. For more information see elsewhere on the website.

3.2 Controller Hardware

As already shown in the block diagram in section 1.2, the heart of the controller hardware is formed by a Microchip 30F4011 digital signal controller.



Prototype schematic

The schematic shows the parts that form the business logic of the prototype:

- the 30F4011 processor,
- the RN41 Bluetooth module and
- the RS232 interface.

Of both communications interfaces (RN41 and MAX232) only the dataleads are used. The RN41 by default has a bitrate of 115k2, which is a tad high for the 4011. For that reason the prototype has a modification to force the bitrate to 9600 by pulling PIO7 HIGH (i.e. pin4 to 3V3).

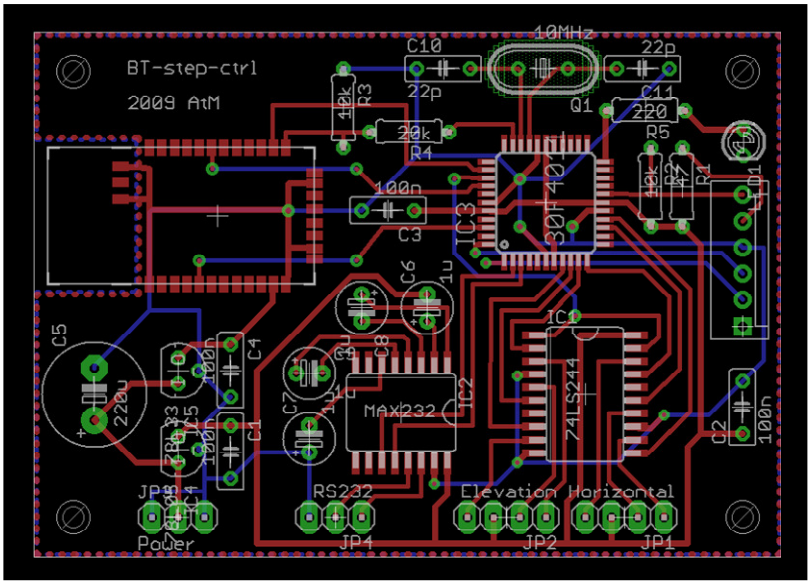
Apart from the dataleads, two control lines are passed from RN41 to 4011:

- PIO2, for indication of 'connected' state, HIGH when connected, LOW otherwise
- RESET, to enable forced reset of the RN41 from SW.

The B3 (port B, bit3) pin of the 4011 has been connected to a LED, which is used by the SW to indicate whether it is alive. For this purpose it is flashed by the Xform task, every time it runs (once a second).

Port E of the 4011 is connected to the current driver interface, for stepper pulses and direction. The transceivers are not shown here.

Finally, a programming interface is connected (PGC, PGD, Vpp), but also not shown in the partial schematic.



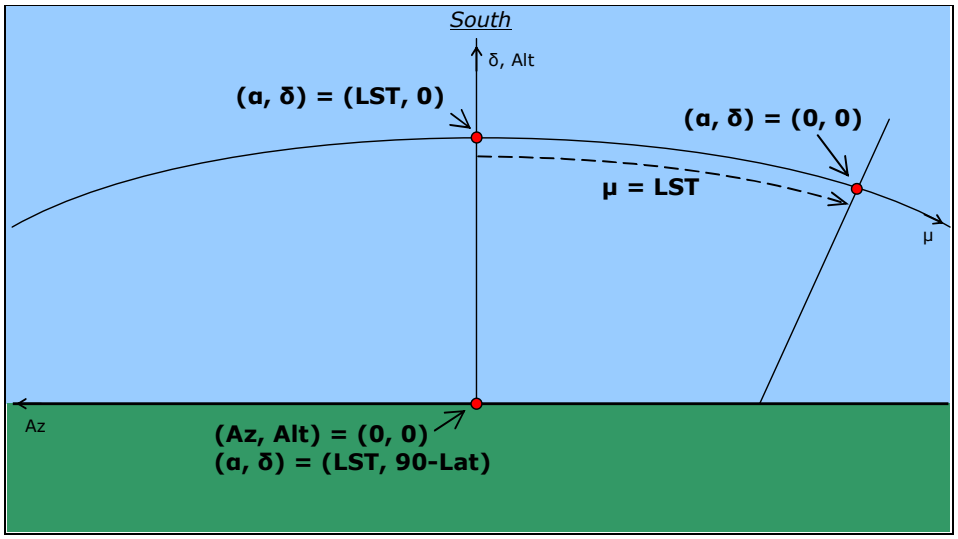
Prototype PCB layout

The board layout shows the placement of the various components. All user interfaces (except the ICD2 program/debug interface) are on one side, which should ease the integration in a box. Notice the absence of ground plane around the antenna area of the Bluetooth module. The PCB should for the same reason be put in a non-conducting enclosure…

4 The transformations applied

4.1 Initialisation

Point telescope south, and to the horizon. For best accuracy the telescope should be level with the azimuthal plane. Switch on the controller, make connection with planetarium program, initialize the controller with data from planetarium (i.e. location and time).



Coordinates near South horizon

On the local meridian, viewing South, the hour angle (μ) by definition is 0. This means that the RA (α) equals the local sidereal time (LST). Conversely, where the RA is 0, the hour angle equals LST. Direction of μ and α are opposite: α increases in eastward direction.

Where the meridian crosses the horizon, Azimuth and Altitude are defined as 0. The Declination in this point by definition is 90º-Latitude. Azimuth increases eastwards.

The LST can be deduced from location and time. Further, the controller will assume that the starting direction of the telescope is on (Az,Alt)=(0,0). An initial transformation matrix can now be deduced based on the local latitude alone.

From the above follows that the matrix to go from equatorial (HA-Dec) to horizontal (Az-Alt) direction vector should describe a rotation about the y-axis (from scope to the east) over an angle of (90º - Lat). This matrix then becomes:

$$T = \begin{pmatrix} \cos(90-lat) & 0 & -\sin(90-lat) \\ 0 & 1 & 0 \\ \sin(90-lat) & 0 & \cos(90-lat) \end{pmatrix} = \begin{pmatrix} \sin(lat) & 0 & -\cos(lat) \\ 0 & 1 & 0 \\ \cos(lat) & 0 & \sin(lat) \end{pmatrix}$$

Note that, contrary to most other systems, the Azimut angle is taken east from the south (instead of east from the north).

The inverse matrix is easily found by adjugation:

$$T^{-1} = \begin{pmatrix} \sin(lat) & 0 & \cos(lat) \\ 0 & 1 & 0 \\ -\cos(lat) & 0 & \sin(lat) \end{pmatrix}$$

Hence all that needs to be done is calculate the sine and cosine of the latitude, and fill in the initial matrix values.

4.2 Calibration

A calibration in practise takes the following steps:

- Measure Hor-Ele directions of three known stars
- Calculate direction vectors with (1)
- Calculate the transformation matrix with (5)
- Calculate the inverse transformation matrix

From now on the calibrated transformation can be used. Calibration can be made more accurate by adding calibration measurements. The oldest set will then be discarded in favor of the new measurement.

To get from one to the other system, the following steps are taken:

- Determine direction vector with (1a) or (1b):
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos(\delta)\cdot\cos(-15\cdot\mu) \\ \cos(\delta)\cdot\sin(-15\cdot\mu) \\ \sin(\delta) \end{pmatrix} \text{ or } \begin{pmatrix} l \\ m \\ n \end{pmatrix} = \begin{pmatrix} \cos(\varepsilon)\cdot\cos(\theta) \\ \cos(\varepsilon)\cdot\sin(\theta) \\ \sin(\varepsilon) \end{pmatrix}$$
- Execute the transformation with (4a) or (4b):
$$\begin{cases} l = t_{11}\cdot x + t_{12}\cdot y + t_{13}\cdot z \\ m = t_{21}\cdot x + t_{22}\cdot y + t_{23}\cdot z \\ n = t_{31}\cdot x + t_{32}\cdot y + t_{33}\cdot z \end{cases} \text{ or } \begin{cases} x = t'_{11}\cdot l + t'_{21}\cdot m + t'_{31}\cdot n \\ y = t'_{12}\cdot l + t'_{22}\cdot m + t'_{32}\cdot n \\ z = t'_{13}\cdot l + t'_{23}\cdot m + t'_{33}\cdot n \end{cases}$$
- Determine coordinates with (2) or (3):
$$\begin{cases} \varepsilon = \arcsin(n) \\ \theta = \arctan\left(\frac{m}{l}\right) \end{cases} \text{ of } \begin{cases} \delta = \arcsin(z) \\ -15\cdot\mu = \arctan\left(\frac{y}{x}\right) \end{cases}$$

4.3 Determination of an inverse matrix

$$T = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \qquad T^{-1} = \frac{adj(T)}{\det(T)}$$

The adjugated of T is determined as follows:

$$adj(t_{ji}) = (-1)^{i+j}\cdot\det(M_{ij})$$

where M_{ij} is minor of element t_{ij} van T. The minor matrix can be obtained by omitting row i and column j. So, for element tt_{12} the adjugated value is:

$$tt_{12} = -\left(t_{12} \cdot t_{33} - t_{13} \cdot t_{32}\right)$$

Finally, the determinant of T is calculated as follows:

$$\det(T) = t_{11} \cdot t_{22} \cdot t_{33} + t_{12} \cdot t_{23} \cdot t_{31} + t_{13} \cdot t_{21} \cdot t_{32} - t_{13} \cdot t_{22} \cdot t_{31} - t_{12} \cdot t_{21} \cdot t_{33} - t_{11} \cdot t_{23} \cdot t_{32}$$

If the transformation T is a pure rotation, then det(T) = 1. In practise the value will probably be close.

5 LX200 commandset

5.1 Command Format

All commands are strings starting with ':' and terminated with '#'.
<CR> and <LF> may be ignored.
Return strings are usually also terminated with a '#'.

Parameter format:

Format	Example	Range	Description
HH:MM.T	05:47.4	00:00.0 - 23:59.9	Hours, minutes, and tenths of minutes.
sDD*MM	+45*59	-90*00 - +90*00	Signed degrees and minutes (the '*' represents ASCII 223 which appears on the handbox as a degree symbol).
DDD*MM	254*09	000*00 - 359*59	Unsigned degrees and minutes.
HH:MM:SS	13:15:36	00:00:00 - 23:59:59	Hours, minutes, and seconds.
MM/DD/YY	02/06/92	01/01/00 - 12/31/99	Month, day, and year. The two digit year indicates the following: 92-99 = 1992-1999 00-91 = 2000-2091
sHH	-5	-24 - +24	Signed hour offset.
NNNN	3456	0000 - 9999	Four digit object number.
sMM.M	02.4	05.5 - 20.0	Signed magnitude value.
NNN	134	000 - 200	Three digit object size (minutes).
DD*	56*	00* - 90*	"Higher" parameter (degrees).
TT.T	59.2	56.4 - 60.1	Tracking "frequency."
<obj> info	CNGC1976 SU DNEBMAG 3.9 SZ 66.0'	n/a	Object information.
Ok	1	0 or 1	Status value returned after setting values. If the valueis legal 1 is returned, otherwise 0 is returned.

5.2 General Telescope Information

Command	Returns	Description
:GR#	+HH:MM.T#	Gets the current Right Ascension.
:GD#	sDD*MM#	Gets the current Declination.
:GA#	sDD*MM#	Gets the current Altitude.
:GZ#	DDD*MM#	Gets the current Azimuth.
:GS#	HH:MM:SS#	Gets the current sidereal time.
:SS HH:MM:SS#	Ok	Sets the sidereal time.
:GL# :Ga#	HH:MM:SS#	Gets the local time either in 24 hour (GL) or 12 hour (Ga) format.
:SL	Ok	Sets the local time.

HH:MM:SS#		NOTE: The parameter should always be in 24 hour format.
:GC#	MM/DD/YY#	Gets the calendar date.
:SC MM/DD/YY#	Ok	Sets the calendar date. NOTE: After the Ok, if the date is valid, two strings will be sent. The first will contain the message "Updating planetary data," the second (sent after the planetary calculations) will contain only blanks. Both strings will be terminated by the "#" symbol.
:Gt#	sDD*MM#	Gets the latitude of the currently selected site.
:St sDD*MM#	Ok	Sets the latitude of the currently selected site.
:Gg#	DDD*MM#	Gets the longitude of the currently selected site.
:Sg DDD*MM#	Ok	Sets the longitude of the currently selected site.
:GG#	sHH#	Gets the offset from Greenwich Mean Time.
:SG sHH#	Ok	Sets the offset from Greenwich Mean Time.
:W1# :W2# :W3# :W4#	Nothing	Sets the current site number.

5.3 Telescope Motion

Undocumented command:

:Mg{e,s,w,n}nnnn# , where *nnnn* is nr of msec to move in e, s, w, or n direction

Command	Returns	Description
:Mn# :Ms# :Me# :Mw#	Nothing	Starts motion in the specified direction at the current rate.
:MS#	0, 1, 2, or 4	Slews telescope to current object coordinates. 0 is returned if the telescope can complete the slew, 1 is returned if the object is below the horizon, 2 is returned if the object is below the "higher" limit, and 4 is returned if the object is above the lower limit. If 1, 2, or 4 is returned, a string containing an appropriate message is also returned.
:MA#	0	Slews telescope to object alt-az coordinates (set with the Sa and Sz Commands). This Command only works in the LAND and ALTAZ modes.
:Qn# :Qs# :Qe# :Qw#	Nothing	Stops motion in the specified direction. Also stops the telescope if a slew to an object is in progress.
:Q#	Nothing	Stops a slew to an object.
:RG# :RC# :RM#	Nothing	Sets the motion rate to guide (RG), center (RC), find (RM), or slew (RS).

:RS#		
:Sw N#	Ok	Sets the maximum slew rate to "N" degrees per second where N is 2 through 4.

5.4 Home Position

Command	Returns	Description
:hS#	Nothing	Starts a home position search and saves the telescope position. NOTE: All Commands except ":Q#" and ":h?#" are disabled during the search.
:hF#	Nothing	Starts a home position search and sets the telescope position according to the saved values. NOTE: All Commands except ":Q#" and ":h?#" are disabled during the search.
:hP#	Nothing	Slews the telescope to the home position.
:h?#	0, 1, or 2	Returns the home status: 0 if home search failed or not yet attempted, 1 if home position found, or 2 if a home search is in progress.

5.5 Library/Objects

Command	Returns	Description
:Gr#	HH:MM.T#	Gets object right ascension.
:Sr HH:MM.T#	Ok	Sets object right ascension.
:Gd#	sDD*MM#	Gets object declination.
:Sd sDD*MM#	Ok	Sets object declination.
:Sa sDD*MM#	Ok	Sets object altitude (for MA Command).
:Sz DDD*MM#	Ok	Sets object azimuth (for MA Command).
:CM#	(see description)	Sync. Matches current telescope coordinates to the object coordinates and sends a string indicating which object's coordinates were used.
:Gy#	GPDCO#	Gets the "type" string for the FIND operation. A capital letter means that the corresponding type is selected while a lower case letter indicates it is not.
:Sy GPDCO#	Ok	Sets the "type" string for the FIND operation.
:Gq#	SU#, EX#, VG#, GD#, FR#, PR#, or VP#	Gets the current minimum quality for the FIND operation.
:Sq#	Nothing	Steps to the next minimum quality for the FIND operation.
:Gh#	DD*#	Gets the current "higher" limit.\
:Sh DD#	Ok	Sets the current "higher" limit.
:Go#	DD*#	Gets the current "lower" limit.
:So DD*#	Ok	Sets the current "lower" limit.
:Gb# :Gf#	sMM.M#	Gets the brighter (Gb) or fainter (Gf) magnitude limit for the FIND

		operation.
:Sb sMM.M# :Sf sMM.M#	Ok	Sets the brighter (Sb) or fainter (Sf) magnitude limit for the FIND operation.
:Gl# :Gs#	NNN'#	Gets the larger (Gl) or smaller (Gs) size limit for the FIND operation.
:Sl NNN# :Ss NNN#	Ok	Sets the larger (Sl) or smaller (Ss) size limit for the FIND operation.
:GF#	NNN'#	Gets the field radius of the FIELD operation.
:SF NNN#	Ok	Sets the field radius of the FIELD operation.
:LF#	Nothing	Starts a FIND operation.
:LN#	Nothing	Finds the next object in a FIND sequence.
:LB#	Nothing	Finds the previous object in a FIND sequence.
:Lf#	(see description)	Performs a FIELD operation returning a string containing the number of objects in the field and the object that is closest to the center of the field.
:LC NNNN# :LM NNNN# :LS NNNN#	Nothing	Sets the object to the NGC (LC), Messier (LM), or Star (LS) specified by the number. Planets are "stars" 901- 909. The object type returned for LC and LS Commands depends on which object type has been selected with the Lo and Ls Commands (see below).
:LI#	<obj> info#	Gets the current object information.
:Lo N#	Ok	Sets the NGC object library type. 0 is the NGC library, 1 is the IC library, and 2 is the UGC library. This operation is successful only if the user has a version of the software that includes the desired library.
:Ls N#	Ok	Sets the STAR object library type. 0 is the STAR library, 1 is the SAO library, and 2 is the GCVS library. This operation is successful only if the user has a version of the software that includes the desired library.

5.6 Miscellaneous

Command	Returns	Description
:B+# :B-# :B0# :B1# :B2# :B3#	Nothing	Increases (B+) or decreases (B-) reticle brightness, or sets to one of the flashing modes (B0, B1, B2, or B3).
:F+# :F-# :FQ# :FF# :FS#	Nothing	Starts focus out (F+), starts focus in (F-), stops focus change (FQ), sets focus fast (FF), or sets focus slow (FS).
:GM# :GN# :GO# :GP#	XYZ#	Gets SITE name (XYZ). M through N correspond to 1 through 4.

:SM XYZ# :SN XYZ# :SO XYZ# :SP XYZ#	Ok	Sets SITE name.
:GT#	TT.T#	Gets the current track "frequency."
:ST TT.T#	Ok	Sets the current track "frequency."
:TM# :TQ# :T+# :T-#	Nothing	Switch to manual (TM) or quartz (TM). Increment (T+) or decrement (T-) manual frequency by one tenth.
:D#	(see description)	Gets the distance "bars" string.
:AL# :AP# :AA#	Nothing	Sets the telescopes alignment type to LAND, POLAR, or ALTAZ.
:r+# :r-#	Nothing	Turns the field de-rotator on (:r+#) and off (:r-#).
:f+# :f-#	Nothing	Turns the fan on (:f+#) and off (:f-#).

5.7 Reference stars

Name	Right Ascension			Declination			Mag
	h	m	s	d	m	s	
Acamar	02	58	15.6468	- 40	18	17.045	3.22
Achernar	01	37	42.8435	- 57	14	12.300	0.54
Acrux	12	26	35.8949	- 63	05	56.570	1.28
Adara	06	58	37.5458	- 28	58	19.517	1.53
Albireo	19	30	43.2879	+ 27	57	34.817	3.08
Alcor	13	25	13.5371	+ 54	59	16.614	4.00
Alcyone	03	47	29.0755	+ 24	06	18.503	2.88
Aldebaran	04	35	55.2417	+ 16	30	33.444	0.99
Alderamin	21	18	34.7737	+ 62	35	08.067	2.47
Algenib	00	13	14.1516	+ 15	11	00.933	2.84
Algieba	10	19	58.1346	+ 19	50	30.925	2.23
Algol	03	08	10.1123	+ 40	57	20.501	2.11
Alhena	06	37	42.7283	+ 16	23	57.381	2.02
Alioth	12	54	01.7469	+ 55	57	35.370	1.76
Alkaid	13	47	32.4385	+ 49	18	47.708	1.86
Almaak	02	03	53.9551	+ 42	19	47.033	2.17
Alnair	22	08	13.9881	- 46	57	39.784	1.77
Alnath	05	26	17.5162	+ 28	36	26.838	1.68
Alnilam	05	36	12.8117	- 01	12	06.924	1.72

Alnitak	05	40	45.5243	-	01	56	33.277	1.90
Alphard	09	27	35.2433	-	08	39	30.970	1.99
Alphekka	15	34	41.2728	+	26	42	52.872	2.22
Alpheratz	00	08	23.2562	+	29	05	25.541	2.06
Alshain	19	55	18.7967	+	06	24	24.390	3.72
Altair	19	50	46.9994	+	08	52	05.980	0.93
Ankaa	00	26	17.0192	-	42	18	21.969	2.40
Antares	16	29	24.4586	-	26	25	55.213	1.07
Arcturus	14	15	39.6698	+	19	10	56.706	0.16
Arneb	05	32	43.8168	-	17	49	20.219	2.59
Bellatrix	05	25	07.8613	+	06	20	58.929	1.66
Betelgeuse	05	55	10.3009	+	07	24	25.420	0.57
Canopus	06	23	57.0985	-	52	41	44.190	-0.63
Capella	05	16	41.3591	+	45	59	52.768	0.08
Castor	07	34	36.1470	+	31	53	18.825	1.58
Cor-Caroli	12	56	01.6658	+	38	19	06.182	2.89
Deneb	20	41	25.9137	+	45	16	49.218	1.33
Denebola	11	49	03.5961	+	14	34	19.352	2.13
Diphda	00	43	35.3699	-	17	59	11.679	2.05
Dubhe	11	03	43.6659	+	61	45	03.693	1.82
Enif	21	44	11.1539	+	09	52	30.044	2.39
Etamin	17	56	36.3702	+	51	29	19.998	2.23
Fomalhaut	22	57	39.0459	-	29	37	20.046	1.23
Hadar	14	03	49.3987	-	60	22	23.006	0.64
Hamal	02	07	10.4026	+	23	27	44.709	2.02
Izar	14	44	59.2219	+	27	04	27.170	2.50
Kaus-Australis	18	24	10.3154	-	34	23	04.604	1.81
Kocab	14	50	42.3281	+	74	09	19.798	2.06
Markab	23	04	45.6531	+	15	12	18.947	2.49
Megrez	12	15	25.5571	+	57	01	57.432	3.30
Menkar	03	02	16.7733	+	04	05	23.093	2.55
Merak	11	01	50.4799	+	56	22	56.730	2.35
Mintaka	05	32	00.4000	-	00	17	56.738	2.23
Mira	02	19	20.7872	-	02	58	39.534	6.54
Mirach	01	09	43.9244	+	35	37	13.876	2.08
Mirphak	03	24	19.3733	+	49	51	40.260	1.81
Mizar	13	23	55.5367	+	54	55	31.271	2.22
Nihal	05	28	14.7235	-	20	45	34.001	2.84
Nunki	18	55	15.9288	-	26	17	48.280	2.07
Phad	11	53	49.8289	+	53	41	40.942	2.43
Polaris	02	31	49.1452	+	89	15	50.772	2.00
Pollux	07	45	18.9433	+	28	01	34.423	1.22
Procyon	07	39	18.1183	+	05	13	29.976	0.40
Rasalgethi	17	14	38.8604	+	14	23	24.880	3.37

Rasalhague	17	34	56.0724	+	12	33	36.099	2.09
Regulus	10	08	22.3127	+	11	58	01.955	1.41
Rigel	05	14	32.2700	-	08	12	05.916	0.28
Sadalmelik	22	05	47.0386	-	00	19	11.465	2.94
Saiph	05	47	45.3760	-	09	40	10.779	2.06
Scheat	23	03	46.4589	+	28	04	58.041	2.47
Shaula	17	33	36.5192	-	37	06	13.782	1.63
Shedir	00	40	30.4403	+	56	32	14.382	2.25
Sirius	06	45	08.9433	-	16	42	57.712	-1.44
Spica	13	25	11.5765	-	11	09	40.754	1.06
Tarazed	19	46	15.5807	+	10	36	47.757	2.71
Thuban	14	04	23.3531	+	64	22	33.086	3.65
Unukalhai	15	44	16.0742	+	06	25	32.261	2.63
Vega	18	36	56.3364	+	38	47	01.290	0.03
Vindemiatrix	13	02	10.5987	+	10	57	32.876	2.84