SkyWatcher SupaTrak Mount Hacking

Wiring of Handset

Looking at the connector plug of the handset (6P6C RJ12 connector) with the copper connections uppermost and the lead going away from you, you should see the following colour wires inside the clear plastic shell, from left to right

1. White
2. Black/Brown
3. Red
4. Green
5. Yellow
6. Blue

The wires are used as follows

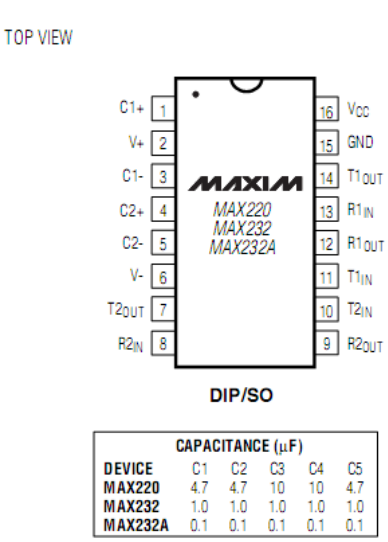
1. Flow Control
2. Ground
3. Data
4. +12V
5. Data (connected to 3 inside the mount)
6. Not Connected

Note that the data connection is duplicated on wires 3 and 5.

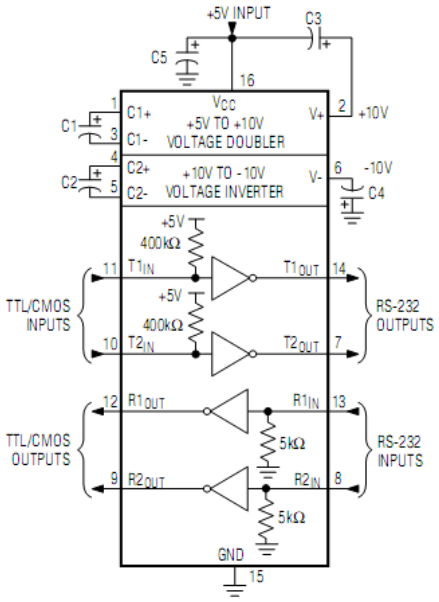
Electrical Signal Protocol

The data connection carries RS232 binary data at TTL levels in a half-duplex manner (i.e. the same line carries signals both from the hand controller to the mount and from the mount to the hand controller). The communcation is at 9600 baud with 8 data bits, 1 stop bit and no parity (9600 8N1). In order to communcate with the scope from a standard PC serial port, it is best to use a RS232 driver chip (for example MAX232 series) which requires just a 5V input.

The flow control line seems to be pulled low by the sending component just before it starts sending data, and released to a high value after the packet has been sent. Presumably the Hand Controller (HC) or mount will refrain from sending information if the line is already low. In practice it doesn't seem necessary to do this to control the scope from the PC.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



Connections for interface circuit to RS232:

- Ground (Pin 2 – black/brown on HC cable) to pin 15 of MAX232
- +5V (see below) to pin 16 of MAX232
- RS232 Ground (Pin 5 on DSub 9 socket) to pin 15 of MAX232
- RS232 Rxd (Pin 2 on Dsub 9 socket) to pin 14 of MAX 232
- Data (Pin 3/5 on HC cable – red & yellow) to pin 11 of MAX232

Do not forget to connect the 4 capacitors as shown in the diagram above – they are required by the charge pump circuitry that generates +/- 10V from the 5V supply
These connections should be enough to snoop on the HC<->Mount communcations.

To send commands from the PC to the mount, additionally connect

- RS232 Txd (Pin 3 on Dsub 9 socket) to pin 13 of MAX232
- Connect pins 11 & 12 of the MAX232 using a 1k resistor[1].

You can create a 5V supply for the MAX232 by simply using an L7805 voltage regulator driven by the +12V supply from the scope.

[Circuit Diagram to Follow].

[1] No idea if the resistor is really necessary – I used it as a safety approach to reduce the likelihood of breaking something and it worked. A direct connection may be fine.

Communication Protocol

Communication follows the pattern of a request from the HC to the mount followed by a response from the mount to the HC. I have not observed the mount sending any messages except in response to a HC request.

The request takes the general format:

:[C][data]<CR>

The ':' character is literal, [C] represents a single letter which identifies the command, [data] represents optional command data (always observed to be hex encoded as ASCII), <CR> is the carriage return character (ASCII 13).

The reply takes the general format:

=[data]<CR>

Where the '=' character is literal and [data] is optional response data, again ASCII encoded hex, the meaning of which depends on the command being replied to.

Note that where a hex value needs more than one byte to send, it is send least significant byte first (LSB) – thus the value 256 is sent 0001, 257 as 0101, 258 as 0201 etc.

Command Parameter/Response Abbreviations

The following abbreviations are used below when describing command parameters and responses.

[M] - a single character detailing which motor to address – 1=Azimuth, 2=Altitude
[D] - a single character specifying direction 0 = positive (right/up), 1 = negative (left/down)
[V] - a three byte hex value (i.e. 6 characters), specifying a value between 0 and 16777215 (000000 to FFFFFF). Remember that these are LSB first, but the nibbles of each byte are in order, so 1,000,000 which is hex 0F4240 would be seen as 40420F.
[R] - a single character specifying a speed ration – 1 = normal, 3 = fast, 4 = slew(?)

[Ack] - indicates that the only response from the mount is acknowledgement, ie the response is =<CR>.

Command Set

Note that for all commands the ':' prefix and the trailing <CR> is assumed. Similarly for all responses, the '=' prefix and trailing <CR> is assumed.

Command	Syntax	Response	Comments
Motor Present	F[M]	[Ack]	Assumed to be a simple test to check the response of each motor controller
Get Precision	a[M]	[V]	Get the number of divisions in a full circuit of the axis specified. This value seems to be '20501A' or 2117658 for both axes. The identification of this value has been confirmed by fetching position counts (see 'j' below) 10 degrees apart. Note that this count for a full circle implies (oddly enough)

			170 degrees equals exactly almost 1,000,000 counts.
Get Sidereal Rate	D[M]	[V]	Get the speed value required to track at the siderial rate. This value seems to be '1B0300' or 795 for both axes. See 'speed calculations' below for a description of the meaning of this value. This value has been verified by setting the latitude to zero degrees, pointing the scope east and observing the tracking speed sent to the mount.
Stop Motor	L[M]	[Ack]	Immediately stop the motor specified.
Set Motor Direction	G[M][R][D]	[Ack]	Sets the speed ratio and direction for the specified motor. Note that 'high' speed ratio is used for the two highest movement speed on the HC.
Set Motor Speed	I[M][V]	[Ack]	Sets the speed of drive for the specified motor. Details on speed calculations given below.
Run Motor	J[M]	[Ack]	Starts the specified motor at the previously set speed, direction and speed ratio.
Get Status	f[M]	See Below	Get the status of the specified motor
Get Position	j[M]	[V]	Get the position of the specified axis relative to the power on position (which is 0x800000 or 8388608).
Goto Position	S[M][V]	[Ack]	Slew the specified axis to the position specified by [V]. Note that the slewing only works with the speed ratio set to '4' and is inaccurate (+/- 100 counts or so typically).

Decoding the Response to 'Get Status'

The response to get status consists of 3 significant characters, which have the following meanings:

1st Char: Bit 1 set if last move was manual, clear if last move was a goto (S command)
Bit 2 set if current direction is –ve,
Bit 3 set if last movement was at speed ratio 'high'.

i.e. values are :

0 +ve direction set, last move at 'normal', last move 'goto'
1 +ve direction set, last move at 'normal'
3 -ve direction set, last move at 'normal'
5 +ve direction set, last move at 'high'
7 -ve direction set, last move at 'high'

2nd Char: 1 if motor running, 0 if stationary
3rd Char: Seems to be always '3'.

Error Responses

The mount ignores any commands not beginning with the ':' character and will not process lines until a terminating <CR> is received.

Commands addressed to an invalid motor number seem to be largely ignored.

Invalid parameters seem to be flagged with a '!1' response.

Invalid commands seem to be flagged with a '!0' response. Note that some commands not listed here seem to be valid (for instance K and E – this may lead to an enhanced command set or may indicate duplicates as K seems to stop motors like L).

Other error conditions (unexpected parameter data?) is flagged with a '!3' response.

Speed Calculations

Observation shows that lower values of the parameter to the 'I' command imply higher speeds. This makes the speed vary inversly with the value, which is sensible as it allows for high precision specification at low speeds.

For 'normal' speed ratio, the speed value seems to be related to the rotational speed as follows:

Rotation Rate (deg/s) = (counts per second) * 360 / 2117658

(counts per second) = 19531.25 / (speed value)

For our sidereal rate of 795, this gives a counts per second of 24.568, and therefore a rotation rate of 0.0004176 deg/s. Multiply this by 86164.1s per sidereal day and we get 359.86 degrees. Not exact, but close enough (using 794 gives 360.31 degrees, slightly further out).

Where does the magic number 19531.25 come from? Well taking 2117658 counts in a complete circle and dividing by 86164.1s in a sidereal day, we get a requirement for 24.577 counts per second, or 1 per 40.688 milliseconds. Assuming that the rotational speed is inversely proportional to the speed value, we arrive at the conculsion that the speed value multiplied by some timing constant gives the amount of time taken for one count. Dividing 40.688ms by 795 gives 51.18 microseconds as the timing constant. This seems suspiciously close to being 51.2 microseconds (i.e. 512 * 0.1 microseconds) which would be significantly easier to deal with in calculations on a simple microprocessor. 19531.25 is the number of 51.2 microsecond periods in 1 second.

The following table shows the speed values and multiples of sidereal rate for the 4 speeds at speed ratio 'normal'

Speed	Speed Value	Calculated Speed	Measured Speed[2]	Documented Speed (in Manual)
Track 1	397	2x		1x
Track 2	99	8x		4x
Track 3	24	32x	~35x	8x
Non-Track 1	12	64x	64x	32x

For the 'high' speed ratio, the value of 'counts per second' appears to be given roughly by:

'counts per second' = 666666 / (speed value)

This corresponds to a timing constant of 1.5 microseconds (rather than 51.2 microseconds) and gives the following table for the two HC speeds that use 'high' ratio.

Speed	Speed Value	Calculated Speed	Measured Speed	Documented Speed (in Manual)
Non-Track 2	145	187x	184x	64x
Non-Track 3	34	798x	~600x[3]	800x

[2] Measured by timing a movement over 10 degrees in the altitude direction.
[3] Measured as ~12s over 30 degrees. This value is low as no allowance was made for the time taken for the mount to get up to speed.

The handset issues speeds considerably lower than those shown above when tracking, and it seems that any speed below the maximum can be used. No experiments have been carried out outside the ranges sent by the handset for the 'normal' and 'high' speed ratios, but a speed of ~400x sidereal (speed value = 68) worked as expected.

Command Sequences

Startup Sequence

The startup communication when the handset is connected to the mount consists of the following sequence:

```
:F1          Motor Present Check (Azimuth)
=
:F2          Motor Present Check (Altitude)
=
:a1          Get Full Circle Count (Azimuth)
=1A5020
:D1          Get Sidereal Rate (Azimuth)
=1B0300
:a2          Get Full Circle Count (Altitude)
=1A5020
:D2          Get Sidereal Rate (Altitude)
=1B0300
:D2          Get Sidereal Rate (Altitude) (repeated)
=1B0300
```

Move Sequence

This details how the handset requests the mount to move in response to pressing and holding one of the direction buttons – in this case to go left at speed 1 in non-tracking mode.

```
:L1          Stop Motor 1 (Azimuth)
=
:G111        Set ratio & direction for motor 1 – negative direction, 'normal' ratio
=
:I10C0000    Set speed for motor 1 to speed value 0x0C (12)
=
:J1          Run motor 1
=
```

Stop Sequence

This is typical of the sequence sent when a direction button is released

```
:L1          Stop Motor 1
=
:f1          Get Status Motor 1
=303
```

Store Location

This is a typical sequence when a location is stored into the handset by pressing the 'Set' button and a number button

:j1	Get Position (Azimuth)
=E2B27F	Result is 0x7FB2E2
:j2	GetPosition (Altitude)
=464C7F	Result is 0x7F4C46

The returned values are presumably stored in semi-permanent memory in the hand controller.

Goto Location

The rather long sequence below is a typical goto sequence.

:L1	Stop motor 1
=	
:G140	Set motor 1 to 'goto' mode
=	
:j1	Get motor 1 position (to see if we need to go past target point)
=4B4280	
:S100F87F	Set motor 1 target[4]
=	
:J1	Run motor 1
=	
:L2	Stop motor 2
=	
:G240	Set motor 2 to 'goto' mode
=	
:j2	Get motor 2 position
=AF967F	
:S2000080	Set motor 2 target (approaching from below, so this is the target value)
=	
:J2	Run motor 2
=	
:f1	Get status motor 1
=613	Running, High Speed, -ve Direction
:f2	Get status motor 2
=413	Running, High Speed, +ve direction
<last 4 lines repeat while both motors running>	
:f1	Get status motor 1
=203	Stopped, last move –ve direction
:G140	Put motor 1 back in 'goto' mode
=	
:S1000080	Set real target for motor 1
=	
:J1	Run motor 1
=	
:f2	
=413	Motor 2 still running
:f1	
=003	Motor 1 now stopped, last move +ve
:f2	
=003	Motor 2 now stopped, last move +ve

[4] This value set to less than the actual target value because we are approaching in the –ve direction – the HC aims to overshoot and then approach from below to limit backlash.

Note the values of the counts after this goto sequence :

```
:j1
=970080          Azimuth overshot by 151 counts (1.5 arc minutes)
:j2
=E50180          Altitude overshot by 485 counts! (nearly 5 arc minutes!)
```

It is possible to get much closer to the desired values by controlling the motors manually, although there will still be backlash issues and possibly issues with the repeatability of the counts.

Tracking

The sequence below shows how the hand controller gives the mount precise speed instructions in tracking mode and updates them every few seconds based on the new orientation of the mount. This tracking was performed just after powering on the mount, so the mount and hand controller assume that the orientation is horizontal and pointing north, hence the very small altitude speed set.

```
:D2              Get full circle count in altitude direction
=1B0300
:f1              Get azimuth status
=103
:f2              Get altitude status
=103
:j1              Get azimuth location
=500280
:j2              Get altitude location
=FFFF7F
:I2D2B20B        Set altitude speed value to 0xBB2D2. ~0.025 counts/sec, 0.1% sidereal.
=
:G210            Set altitude to run at speed ratio normal in +ve direction
=
:J2              Run altitude motor
=
:I1EE0300        Set azimuth speed value to 0x03EE (1006).  ~19 counts/sec, 79% of sidereal.
=
:G110            Set azimuth motor to normal speed ratio, +ve direction
=
:J1              Run azimuth motor
=
<~4-5 second with no activity>
:f1              Re-check motor statuses (now running, normal ratio, +ve direction)
=113
:f2
=113
:j1
=110380          Re-check positions
:j2
=5E0080
:I290A508        Note updated altitude speed
=
:I1EE0300        Azimuth speed unchanged
=
```

<previous block repeats every 4-5 seconds with new values as appropriate>
<The next sequence shows how tracking is cancelled>

:D1	Get full circle count for Azimuth (why???)
=1B0300	
:L1	Stop azimuth motor
=	
:L2	Stop altitude motor
=	

Setting Latitude

Storing the observation latitude is performed simply by the HC requesting the current position of the altitude axis from the mount (assuming that the movement of the mount since last power on represents the latitude as described in the instruction manual).

:j2	Get altitude count
=E29384	(0x8493E2 – 0x80000) * 360 / 2117658 = 51 degrees

Unknowns

- The GOTO functionality built into the mount is inaccurate – you can get much closer with slow movement commands to tweak the position. This depends on the relative position being accurate. Is it?
- How much backlash is there in the mount mechanism? If we knew how much we could correct for it in software rather than always approaching a point from the same direction.
- What are the additional commands that the mount responds to that the hand controller never sends? Are they useful?