ETX 60/70

HBX Motor Protocol Description

Version: 4

Date: 11 Nov 2017

Table of Contents

Introduction

This document describes the protocol used between the Autostar (#494, #497) controllers and the RA and DEC motors. The device investigated was the ETX60AT controlled by a #494. Some sample implementations an an arduino Mega2560 are included. The information is stored on the github site.

Test software was written to allow the EQMOD/ASCOM project to control the ETX. CdC was used as the astro software.

The test software includes a monitor mode as well as a test mode, selectable by jumpers.

Notices

Information in this document was summarised from multiple sources, especially the Weasner site. It is a collection of those documents and the use of or the reproduction of these documents implies and retains the original warnings/copyrights where they existed. Some work is additional to those documents and may be freely used taking care to carefully examine the caveat of the Patent notice below and its implications.

In particular, the discussions between Dick Seymour, Stefan Keller-Tuberg, Mike Weasner, ultima_gtr_v8, and Gene Chimahusky provided a lot of early information.

e.g. http://www.weasner.com/etx/autostar/2008/494cable.html

The main hardware interface software information comes from Yahoo RoboScope group files:

autostar_mtr_2.html and arduino autostar.zip

Some of this information is reproduced in the github repository.

Additional information on the motors is in the Microchip application notes:

AN893-00893a.pdf, AN905-00905a.pdf, and I2C.pdf.

As described in the sources of the information, ***"some of the information in the documents is derived from US patent #6,304,376 (held by Meade), and observation of the interactions between the Autostar controllers and the DH series motors. The information in this document was not derived via disassembly of the Autostar or motor processor firmware. Care must be taken to not violate the claims of the aforementioned patent in applying this information."***

Any consequences of the use of this information in any way are the sole responsibility of the reader.

ETX60/70 Physical Interfaces

Connector Details

Auxiliary Port

1. Ground
2. Aux Data
3. Aux Clock
4. 12V

HBX Port

1. 12V
2. Aux Clock
3. Aux Data
4. Alt Clock
5. Alt/Az Data
6. Az Clock
7. Alt/Az Data
8. Ground

Power

2.5mm jack, centre positive.

Signal Details

Description

Each motor is connected to the Autostar using two signals (plus a common ground). These clock and data signals allow the implementation of a bidirectional serial protocol.

The clock signal for each motor is always driven by the Autostar, which determines the transfer timing. The frequency of the clock signal is Autostar dependent with a frequency of around 5.5 kHz. On my #494 it is 220uS per data bit. Observations suggest that the AUX clock operates at higher speed. Each clock signal needs a pull up resistor of approximately *50KΩ.*

The data signal is driven by either the Autostar or a motor controller and appears to only be driven low. The signal is weakly pulled high by the Autostar. *Email comments on the original interface suggest that this is approximately 220KΩ.*

Although the wiring between the Autostar and the motors would allow the two sets of signals to be completely separated, it appears that the data lines are tied together in the Autostar controllers. In the 494, all three data lines (on pins 2, 4, 6) are connected together.

Protocol

The following information comes from autostar_mtr_2.html.

The data transfer is structured as a series of commands.

Each command has:

- A start sequence (which allows the Autostar to be sure there is a motor present)
- A command sequence during which the eight bit command is sent by the Autostar
- A data sequence during which some number of bits containing data associated with the command is sent by, or returned to, the Autostar.

The number of bits is a function of the command, but is always constant for a given command.

Start sequence

In response to the Autostar driving the clock line low, the motor drives the data line low. This is used by the #494 Autostar to determine the presence of a motor. The state of the data line appears to be ignored by the #497 Autostar software. The #494 then drives the clock high, and the motor releases the data line to float high. (It does not drive it high)

Command sequence

An eight bit command, MSB first, is sent by the Autostar. This tells the motor what to do and what bits will be transferred later in the transfer.

Data sequence

The most significant bit of the most significant byte (for multibyte data) is transferred first. Some commands use a number of data bits which is not a multiple of eight.

Each bit of data ***after*** the start sequence is transferred as follows:

1. The Autostar drives the clock line high
2. The transmitter of the bit forces the data line low (to transfer a zero value), or allows it to float high
3. The Autostar drives the clock line low
4. The receiver of the bit samples the data line to determine the bit value

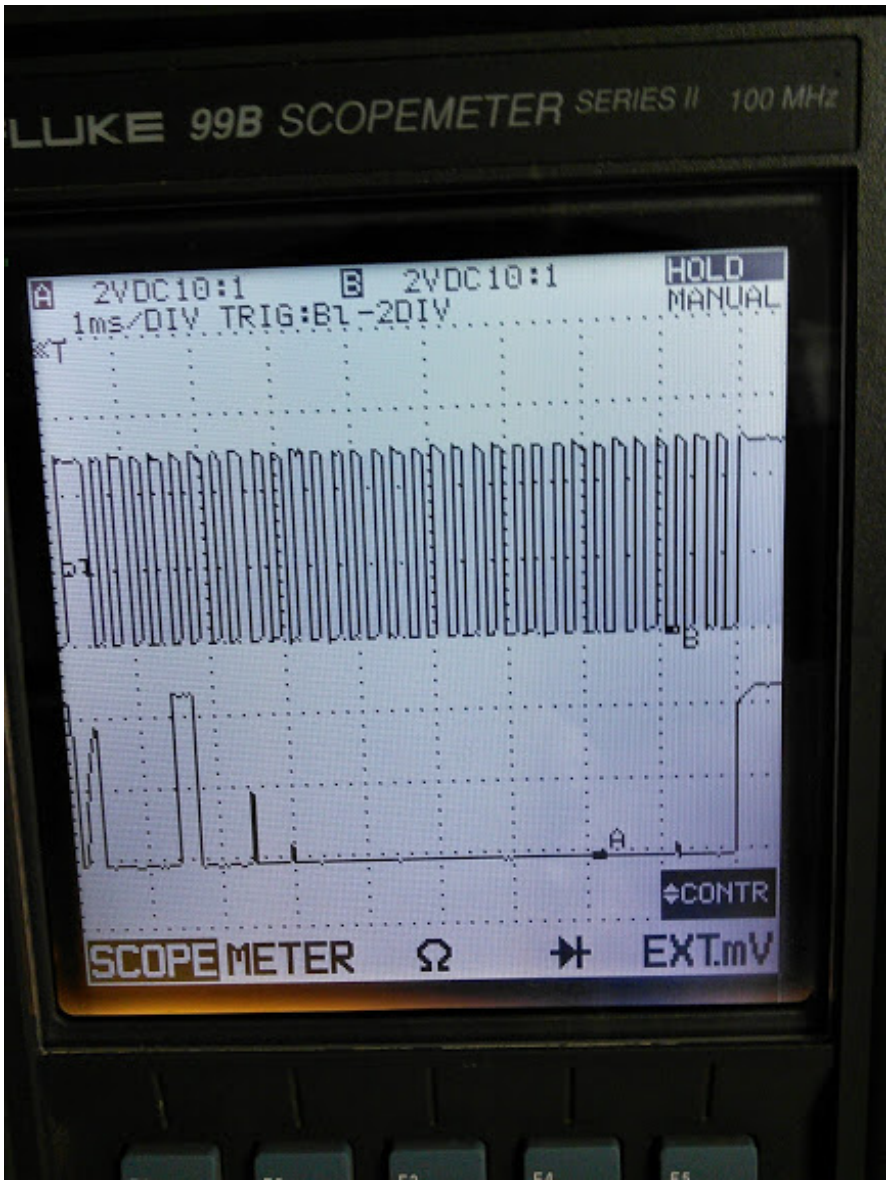


Illustration 1: Typical Status Clock/Data sequence

Command Summary

Command	Parameters	Description	Notes
0x00 (Rotate slow)	Speed [24 bits, 2's comp] (A->M)	Rotate the motor at the specified rate. The rate is a 24-bit 2's complement number of motor ticks per 6.55 ms period in an 8.16 fixed point fractional format.	The protocol assumes that the Autostar and motor act as a closed loop. The changes can be seen as the Autostar converges on a sidereal tracking speed. See Spreadsheet in Appendix A
0x01 (Rotate fast)	Speed [24 bits, 2's comp] (A->M)	Rotate the motor at the specified rate. The rate is unknown but is specified for rates greater than 2x sidereal.	
0x02 (Set offset count)	Error [16 bits] (A -> M)	Add the specified offset to the current position value. The servo loop will try to compensate	This is used by the pulse-guide commands to nudge the scope position. For the #497 it's also used to nudge the scope to the correct position after GOTO in some cases.
0x03 (Set LED current)	Led Value [8 bits] (A->M)	Sets the LED current used for the encoder LED.	This is stored in the Autostar and sent at initialisation.
0x04 (Calibrate LED)	None	Tells the motor to determine the optimal LED current for the optical encoder	Unknown whether it uses this value, or whether it needs to be read and written to take effect.
0x05 (Stop)	None	Stop the motor	The motor may continue to rotate for a short time after this command is issued
0x06 (Full speed reverse)	None	Rotate the motor at the highest possible speed in the negative direction	Speed = 0x0140FFFF
0x07 (Full speed forward)	None	Rotate the motor at the highest possible speed in the positive direction	Speed = 0xFEB80000
0x08 (Read status)	Relative position change [16 bits, 2's comp], PWM duty cycle [8 bits], encoder error [1 bit] (M -> A)	Read information about the position and activity of the motor. The position and flag bit is reset after this command.	If the PWM value is 0xff and the position change is zero then it is likely that the motor is stalled Flag bit - ?? Illegal encoder transition non gray code scope moved too quickly for encoder, or badly calibrated LED
0x09 (Get LED current)	led value [8 bits] (M->A)	Get the LED current.	Issued after the calibrate LED command
0x0b (Get motor type)	motor type [8 bits] (M->A)	Get the type of motor attached	The DH motors returns 0x10. Larger values cause the #494 Autostar to indicate the ETX Autostar should be used.
0xe4 (unknown)	None	Unknown. Used by the Autostar after the stop command, when performing LED calibration, parking the scope, or sleeping the scope	Doesn't seem to clear the status information. Not sure about “after parking”.

Table 1 HBX Command Summary

Motor Gearing

Detailed Information is in the etx60at-gears.ods file.

Az/RA Motor

The following table summarises the AzRA gear train. The encoder on the motor shaft appears to generate 144 pulses per rotation. This generates a 'count' of 1,233,750 for a full rotation (24HR).

Az/RA Motor	Gear1	Gear2	Ratio	Hexadecimal
Encoder	1	144		
G1	18	30	144	
G2	12	30	240	
G3	12	30	600	
G4	12	30	1,500	
G5	12	42	3,750	
Worm	1	94	13,125	
Total		1,233,750	1,233,750	0x12D356
Solar Count/Hour			51,406	0xC8CE
Sidereal Count/Hour			51,266	0xC842
Counts/Worm Tooth			13,125	0x3345

Table 2 RA/AZ Gear Train

Alt/DEC Motor

The following table summarises the AltDEC gear train. The encoder on the motor shaft appears to generate 144 pulses per rotation. This generates a 'count' of 1,315,440 for a full rotation (360˚).

Alt/DEC Motor	Gear1	Gear2	Ratio	Hexadecimal
Encoder	1	144	144	
G1	18	30	240	
G2	12	36	720	
G3	12	36	2,160	
G4	12	36	6,480	
G5	12	42	22,680	
Worm	1	58	1,315,440	
Total		1,315,440	1,315,440	0x141270
Counts/Degree			3,654	0x0E46
Counts/Worm Tooth			22,680	0x5898

Table 3 DEC/ALT Gear Train

Az/RA Motor Experimental Data

The following information was collected by sending the appropriate speed command (Rotate Slow Forward at Sidereal Speed) via the HBX interface and noting the responses to 'read status' query, which returned the current position. The time noted was elapsed time from the start. The calculated value per hour (51,266) corresponds well to the measured value (51,292).

Rotation (arcmin))	Measured Counts	Counts/Minute
0	0	0
6	312,192	52,032
7.5	385,147	51,353
9	459,399	51,044
9.5	486,622	51,223
11	563,099	51,191
12	616,215	51,351
Avg		51,292

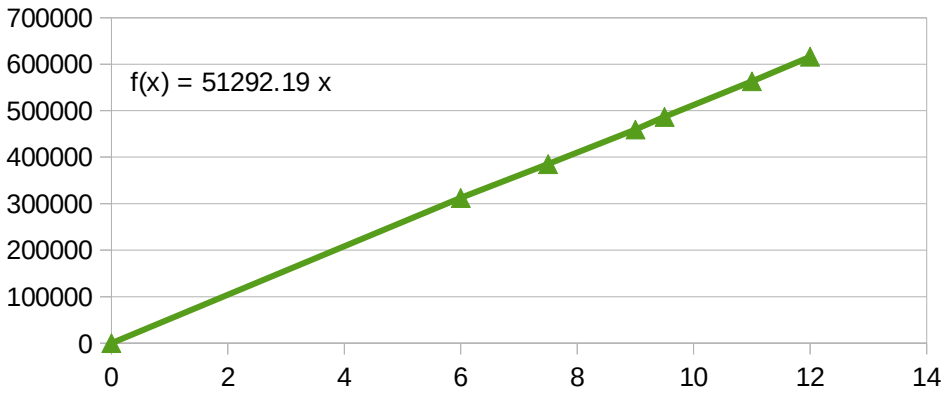


Table 4 Experimental Data – Sidereal Counts per Arc Minute

Motor Speed

The following table summarises the speed measurements. The number of counts was measured experimentally for varying times at increasing speeds. The speed was set using the “Rotate Slow” (0x00) command, using the 8.16 twos complement format. The number of counts was then recorded by sending a “Read status” command and recording the cumulative values. Finally, the value was divided by the measurement time to give the rate at which the counts are returned.

Hex Speed 16.8	Count Value	Time (S)	Command Speed	Rate (Counts/s)
0	0	0		0
000001	1070	1,800	.004	0.6
000020	9136	480	.125	19.0
000040	50213	1,320	.25	38.0
000080	68340	900	.5	75.9
000100	127768	840	1	152.1
000200	475870	1,560	32	305.0
000400	73694	120	64	614.1
000800	72003	60	128	1,200.1
001000	146215	60	256	2,436.9
002000	148719	30	512	4,957.3
004000	287023	30	1024	9,567.4

Table 5 Experimental Rate

The calculated rate (152.59 counts/s) corresponded to the measured value (152.34 ticks/s) within experimental error. This calculated rate is based on 6.55mS (from the patent) and other texts.

Handset Speed	Speed Command	Speed Value - ALT				Speed Value - AZ			
		Forward		Reverse		Forward		Reverse	
		HEX	Decimal	HEX	Decimal	HEX	Decimal	HEX	Decimal
Max	5, 6	NA	NA	NA	NA	NA	NA	NA	NA
2	1	384D61	3689825	C7B29F	13087391	3C073B	3934011	C3F8C5	12843205
1	1	16855A	1475930	E97AA6	15301286	180317	1573655	E7FCE9	15203561
.5	1	0C02DA	787162	F3FD26	15990054	0CCE53	839251	F331AD	15937965
64	1	06016D	393581	F9FE93	16383635	066729	419625	F998D7	16357591
32	1	0300B6	196790	FCFF4A	16580426	033394	209812	FCCC6C	16567404
16	1	01805B	98395	FE7FA5	16678821	0199CA	104906	FE6636	16672310
8	1	00C02D	49197	FF3FD3	16728019	00CCE5	52453	FF331B	16724763
2	1	00170E	5902	FFE8F2	16771314	001894	6292	FFE76C	16770924

Table 6 Handset Speed Values to Command Values

Captured Sequences

Original Weasner Site

The following example commands sequences may help understand the interaction of the Autostar and Motor Controller (MC). Note that these are the #494 sequences, but they seem to be the same as the #497 ones. (Values returned by status command often differ due to differences in motor speed and command timing)

Startup

Command		Data from MC	Data to MC
0x05	Stop Motor		
0x08	Read Status	0x00 0x00 0x00 0	
0x0b	Get Motor Type	0x10	
0x03	Set LED current		0xe7

Initial alignment

Command		Data from MC	Data to MC
0x05	Stop Motor		
0x08	Read Status	0x00 0x00 0x00 0	
0x07	Full Speed Forward		
0x08	Read Status	0x00 0x2a 0x00 0	
0x08	Read Status	0x00 0x2b 0x00 0	

The value returned depends on the sampling (status read) time.

Move scope

Command		Data from MC	Data to MC
0x08	Read Status	0x00 0x00 0x00 0	
0x07	Full Speed Forward		
0x08	Read Status	0x00 0x2a 0x00 0	
0x00	Set Speed Slow		0x47 0x1c 0x6f

Sleep scope

Command		Data from MC	Data to MC
0x05	Stop Motor		
0xE4	Unknown		

Park scope

Command		Data from MC	Data to MC
0x00	Set Speed Slow		0xFF 0xA1 0x57
0x08	Read Status	0xFF 0xFF 0x00 0	

0x05	Stop Motor		

Calibrate motors

Command		Data from MC	Data to MC
0x05	Stop Motor		
0xE4	Unknown (Sleep?)		
Assumed hard reset (using CLK)	Motors move (1-2 sec)		
0x05	Stop Motor		
Ox04	Calibrate LED current		
Ox09	Get LED current	0xE7	
0x08	Read Status	0x25 0x6c 0x00 0	

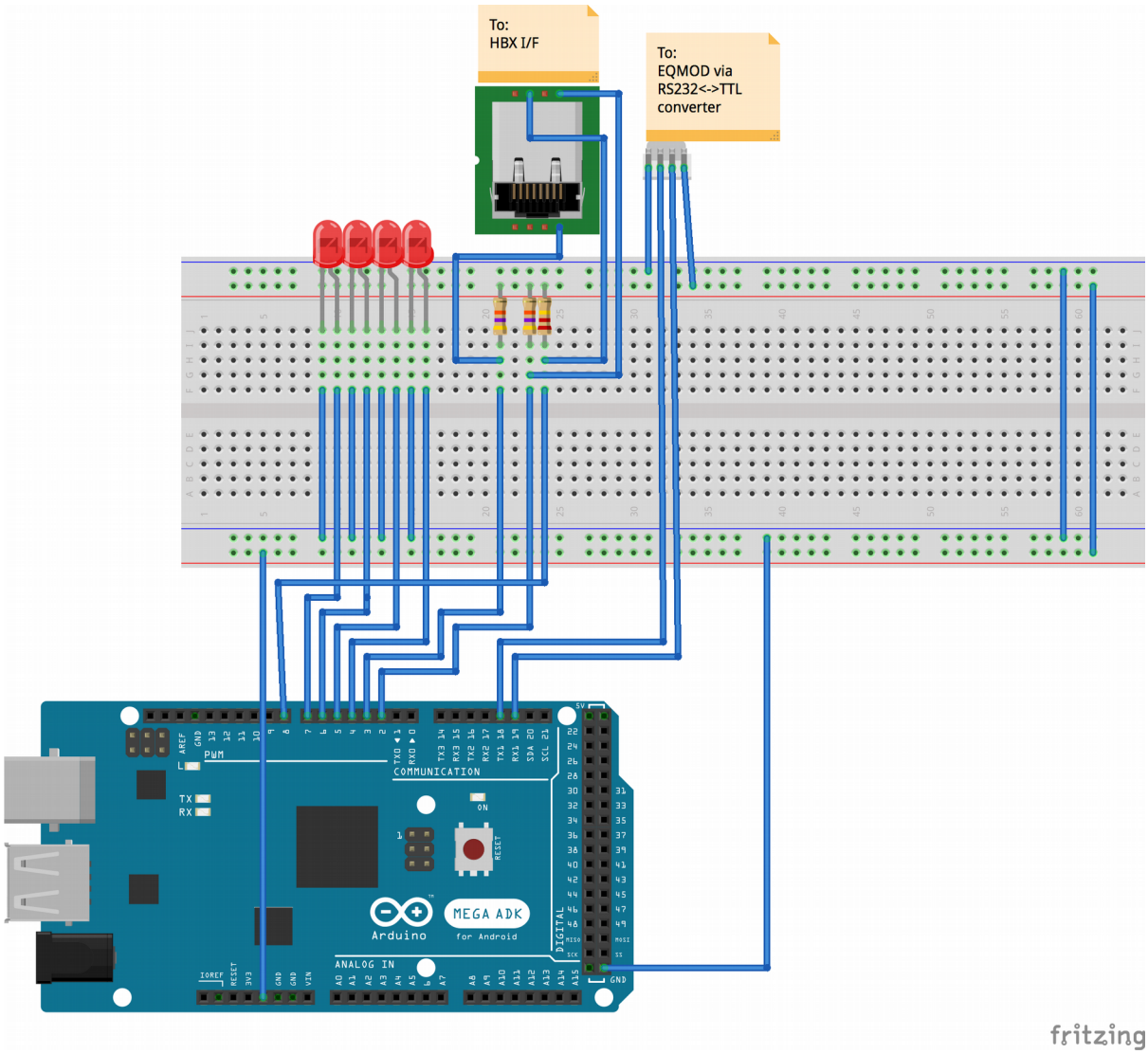
Just for info,
when using the x00 or x01 motor commands
to command a constant motor speed,
the 24 bit no sent appears to be derived from
a base value (that is scope agnostic) multiplied by the ratio.
This base no is constant across all the Autostars (incl ASII)
and hence a simplified drive could be made using this rate
as a hardcoded var
ie
BaseRate for 1deg/sec := 1,546,188.4
sooo for an LX200GPS
Motor command value for 1deg/sec := 1,546,188.4 * 7.1111111
:= 10,995,117

Andrew Johansen Melbourne Australia

The LX200s can be run up to 8deg per second using the x00 and x01 commands.
In the 497s, there is an internal speed limit
this is based on the fact that the rate sent to the motor
must be less that x007FFFFE (8,388,606),
ie the max "controllable" speed is inversely proportional to the ratio.

Above that rate, the full Fwds or Full Reverse commands must be used.

Typical Hardware Setup



Current Arduino Capture

alignment to achernar

The graphs below are derived from the monitored startup values captured from a #494 doing an EASY alignment. They show the position in counts and speed values set on successive status calls and speed setting.

Values in #494 were set at nov16, 2017, 8.00 pm. See ETXStartup.ods