

## Alltrax AXE Motor Controller Serial Port Communication

The Alltrax AXE has a serial port that can be used to program the device and read real time data. This file is concerned with reading real time data only. Writing program and/or EEPROM could easily cause problems with the controller, motor, batteries, etc. This is information I have found and verified by testing on an AXE controller. I have not verified this information with Alltrax. Therefore, use at your own risk, and do not hold Alltrax or myself responsible for any damage done by using this information.

Note that the AXE may send out some bogus data on power up. Good practice would be to clear the host input buffer before sending a command.

### Serial Port Parameters:

The AXE is set to 9600 baud, 8 data bits, 1 stop bit, no parity. No Handshaking.

### Data Coding:

Byte values (not ASCII) are used. All 8 bits are required.

### Protocol:

A command string is sent from the host. The AXE responds with a reply.

### Packet Format:

A packet consists of seven bytes; all seven must be transmitted.

| byte | function                |  |
|------|-------------------------|--|
| 0    | source/target addresses | the high order hex digit is the source, low order is the target. Host is hex 5, AXE is hex B |
| 1    | command                 | the command byte (see below)   |
| 2    | address                 | RAM address to start reading from  |
| 3-5  | data                    | don't cares in command (but value will affect the checksum), RAM data in reply               |
| 6    | checksum                | overflowed checksum (sum of all 7 bytes = 0x00 when truncated to one byte)                   |

### Commands:

Again, this file only deals with command 0x04, the others are shown for reference.

- 0x00 - read byte from program memory
- 0x01 - write byte to program memory
- 0x02 - read byte from EEPROM
- 0x03 - write byte to EEPROM
- 0x04 - Read three sequential bytes from RAM starting at the requested address
- 0x05 - Reset the AXE program and start up again

## Addresses:

Most data is encoded into two bytes (low order byte first)

The following data is available:

| adr  | value                              |   |
|------|------------------------------------|---|
| ---  | -----                              |   |
| 0x20 | throttle position - one byte       | Value is 0 to 255 = 0% to 100%              |
| 0x2C | controller temperature - two bytes | Value is in degrees K 2.048 bits/deg        |
| 0x39 | battery voltage - two bytes        | Value is Volts 0.1025V per bit              |
| 0x3B | Error status - one byte            | (***)I need to fill in the bit values here) |
| 0x60 | Motor Current - two bytes          | Value is Amps                               |

## Examples:

To read battery voltage, send:

0x5B 0x04 0x39 0x00 0x00 0x00 0x68

Where:

0x5B means we are sending from the Host (hex 5) to the AXE (hex B)  
0x04 is the read command  
0x39 is the start address of the battery voltage data  
0x00 three bytes of don't care data  
0x00  
0x00  
0x68 checksum of the previous six bytes :  
 $0x5B + 0x04 + 0x39 + 0x00 + 0x00 + 0x00 = 0x98$   
 $0x98 \text{ XOR } 0xFF = 0x67$   
 $0x67 + 1 = 0x68$

The reply is:

0xB5 0x04 0x39 0x66 0x01 0x00 0xA7

Where:

0xB5 sending from AXE (hex B) to Host (hex 5)  
0x04 reply for read command  
0x39 starting at address 0x39  
0x0166 the battery voltage = 358 decimal \* 0.1025V/bit = 36.7 Volts  
0x00 third byte, not needed (is actually the Error byte here)  
0xA7 checksum calculated as above

To read temperature, send:

0x5B 0x04 0x2C 0x00 0x00 0x00 0x75

The reply is:

0xB5 0x04 0x2C 0x4A 0x02 0x00 0xCF

Where:

data is 0x024a = 586 decimal / 2.048 bits/deg = 286 deg K - 273 = 13 deg C