Delta-T Communication Protocol

Introduction

Communication with the Delta-T can take place over one of the following channels:

- A virtual serial port provided by a USB connection to the Delta-T
- A serial connection via the RS232 port on the Delta-T (via a hardware serial port or a USB-to-Serial adapter)

The 2nd revision of the Delta-T is also equipped with an Ethernet port. However, as of this writing Ethernet communication is not recommended for general use.

If connecting via the RS232 port, a baud rate of 19,200 bps should be used. If connecting via the USB port, the baud rate generally does not matter.

Packet structure

Each packet sent to or received from the Delta-T is structured as follows:

SOM	1 byte	Start Of Message = 59 = 0x3B
NUM	1 byte	Number of bytes that follow, excluding CHK. Must be at least 3 to account for SRC, RCV, and CMD. If (for example) the DATA payload includes 3 bytes, then NUM = 6.
SRC	1 byte	Source Address. See below for a list.
RCV	1 byte	Receiver Address. See below for a list.
CMD	1 byte	Command. See below for a list.
DATA	0-N bytes	Data payload for the command or response. Can be 0 or more bytes.
CHK	1 byte	Checksum. Calculated by summing the bytes of the packet, excluding the SOM and CHK, and taking the Least Significant Byte of the two's complement.

Addresses are as follows:

- PC: 0x20
- Delta-T: 0x32

Sample exchange

The following shows a typical exchange for the GET_VERSION command:

SEND:

```
3B      // SOM
03      // 3 bytes coming up before checksum (SRC, RCV, and CMD)
20      // Message is being sent by the PC
32      // Message is being sent to the Delta-T
FE      // 0xFE = GET_VERSION command
AD      // Checksum = (-(0x03 + 0x20 + 0x32 + 0xFE)) & 0xFF
```

RECEIVE:

```
3B    // SOM
07    // 7 bytes coming up before checksum (SRC, RCV, CMD, and 4 DATA bytes)
32    // Message is being sent by the Delta-T
20    // Message is being sent to the PC
FE    // Message is in response to a GET_VERSION command
01    // First byte of DATA payload
00    // Second byte of DATA payload
33    // Third byte of DATA payload
A3    // Fourth byte of DATA payload
D2    // Checksum = (-(0x07 + 0x32 + 0x20 + 0xFE + 0x01 + 0x00 + 0x33 + 0xA3)) & 0XFF
```

Commands

(0x80) CMD_FORCE_RESET

Description: Force a software reset.

Parameters: <none>

Returns: <none>

(0x81) CMD_FORCE_BOOT

Description: Force a reset into bootloader for firmware update.

Parameters: <none>

Returns: <none>

(0xB0) COH_NUMHEATERS

Description: Returns the number of available heaters that can be indexed.

Parameters: <none>

Returns: **1 Byte**

0	Number of heater channels

(0xB1) COH_ON_MANUAL

Description: Enables a heater channel for manual control with specified duty cycle and period.

Parameters: **4 Bytes**

0	Index of heater channel <0 … NUMHEATERS-1>
1,2	Period in 10ths of a second (LSB, MSB)
3	The duty cycle in percent 1 – 100 (%)

Returns: **1 Byte**

0	Result of operation – see appendix

(0xB4) COH_OFF

Description: Turns off the specified heater.

Parameters: **1 Byte**

0	Index of heater channel <0 … NUMHEATERS-1>

Returns: **1 Byte**

0	Result of operation – see appendix

(0xB5) COH_REPORT

Description: Returns a packed structure with data on the current operational state for the specified heater.

Parameters: **1 Byte**

0	Index of heater channel <0 … NUMHEATERS-1>

Returns: **12 Bytes**

0-11	<Packed Structure – see appendix>

(0xBF) COH_RESCAN

Description: Rescans 1-Wire bus and updates temp sensors

Parameters: <none>

Returns: **1 Byte**

0	number of sensors found after re-scan

(0xFE) CMD_GET_VERSION

Description: Gets the MAJ.MIN.BLD format version number from the device

Parameters: <none>

Returns: **4 Byte**

0	Version MAJOR
1	Version MINOR
2, 3	Version BLD (YYDDD) YY=Year, DDD=day of year !(MSB, LSB)

APPENDIX: AUX Data Structures, and Response Values

Note that some of the fields and values below are provisioned in the protocol but not currently used.

```
typedef __packed struct HtrReportTg { // A report structure for AUX packet data
  UBYTE                               //
    StateUB;                          // 0    State of the heater channel
  UBYTE                               //
    ModeUB;                           // 1    Mode heater channel is in
  UWORD                               //
    SetPointUW;                       // 2-3  setpoint (12-bit format) for heater channel
  UBYTE                               //
    TempHtrIdUB;                      // 4    temp sensor ID associated with channel
  UWORD                               //
    TempHtrUW;                        // 5-6  temperature of the associated sensor
  UWORD                               //
    TempAmbUW;                        // 7-8  ambient temperature
  UWORD                               //
    PeriodUW;                         // 9-10 period of PWM control (10ths of a second)
  UBYTE                               //
    DutyCycleUB;                      // 11   duty cycle of PWM control
} HtrReportTy;                        //

#define HTR_STATE_OFF          0      // heater is off
#define HTR_STATE_ON           1      // heater is on
#define HTR_STATE_USERON       2      // heater is on by user (switch) (NOT IMPLEMENTED)

#define HTR_MODE_MANUAL        0x01   // manual mode PWM control
#define HTR_MODE_RELATIVE      0x02   // control channel relative to ambient
#define HTR_MODE_ABSOLUTE      0x03   // control channel to absolute temperature
#define HTR_MODE_OVERRIDE      0x04   // override is active (switch)

#define HTR_ERR_NONE           0x80   // error: none
#define HTR_ERR_USERMODEACTIVE 0x81   // error: "user mode" is active
#define HTR_ERR_INVALIDHEATER  0x82   // error: invalid heater number
#define HTR_ERR_SETPOINTRANGE  0x83   // error: setpoint is out of range
#define HTR_ERR_PWMPERIOD      0x84   // error: pwm period is invalid
#define HTR_ERR_PWMDUTYCYCLE   0x85   // error: duty cycle is invalid
```