DistoX Advanced Information

Firmware Version 1.3 & 1.4

2010/12/12

Introduction

This document describes some additional functions of the DistoX as well as the binary protocol used for the communication between the Disto and a PDA or PC connected over Bluetooth.

A description of the basic functions of the DistoX and the calibration procedure can be found in the documents “DistoX User Manual” and “DistoX Calibration Manual”.

Advanced Functions

Silent Mode

In Silent Mode the device does not transfer any measured data to the PDA. The data is still written to the internal memory but is immediately marked as ‘already transferred’. The same holds for any pending data present when Silent Mode is entered.
When in Silent Mode the device shows a zero in the top middle of the screen (memory number). Normally either the number of pending measurements or nothing is visible there.
Silent Mode may be switched on or off manually by pressing the following key sequence on the Disto key pad:

AREA AREA REF REF CLR.

Test Displays

In Area mode (after pressing the AREA key), the screen shows the actual azimuth and inclination values as during normal (DIST key triggered) measurements. There are, however, two differences to a normal measurement:
- The compass keeps running even if the Laser switches off after some time.
- If the reference is set to Front (by pressing the REF key), the display changes from azimuth/inclination to the actual roll angle in the upper line and the dip angle (inclination of the magnetic field) in the lower line.
Do not take measurements in this mode! They will neither be stored nor transferred and the memory display may be become inconsistent.

In Volume mode (after pressing the AREA key twice), the screen shows the firmware version in the upper line and the serial umber of the extension board in the lower line. In this mode, switching to Front reference and back with the REF key toggles the Silent Mode on and off (see above).

The DistoX Wire(less) Protocol

The DistoX uses the Bluetooth Serial Port Profile (SPP) to communicate to a PDA or PC. The Bluetooth connection is always set up by the PDA. SPP offers a simple bidirectional byte oriented channel. Communication takes place in the form of individual transactions. There are two kinds of transactions:
Disto initiated and PDA initiated.

Disto initiated transactions consist of a data packet sent from the Disto to the PDA and an acknowledge packet sent from the PDA to the Disto. Data is resent continuously in intervals of 5 seconds until a valid acknowledge is received.

PDA initiated transactions consist of a command packet sent from the PDA to the Disto followed by a reply packet sent from the Disto to the PDA in most cases. The PDA side should repeat the command if no reply is received.

Data Packets

All data packets have a length of 8 bytes.

Measurement Data Packet:
Byte 0: bit 7: sequence bit, bit 6: bit 16 of distance, bits 0-5: 000001 (packet type)
Byte 1: low byte of distance
Byte 2: high byte of distance
Byte 3: low byte of declination
Byte 4: high byte of declination
Byte 5: low byte of inclination
Byte 6: high byte of inclination
Byte 7: high byte of roll angle

The sequence bit is inverted for each new transaction. A packet with the same contents and the same sequence bit as the preceding packet is a wrongly repeated packet. It should be acknowledged and discarded. For an illustrative code sample see Appendix A.

The 17 bit distance is in mm.
The 16 bit angles are in radiant * 2^15 / Pi (full circle = 2^16).
For declination: 0x0000 = north, 0x4000 = east, 0x8000 = south, 0xC000 = west.
For inclination: 0x0000 = horizontal, 0x4000 = up, 0xC000 = down.
The roll angle is in radiant * 2^7 / Pi (full circle = 2^8).
Display orientation: 0x00 = up, 0x40 = left, 0x80 = down, 0xC0 = right.
The units used in the data packets are independent of the units selected for the Disto display.

Calibration Data Packet 1 (Acceleration Sensor Reading):
Byte 0: bit 7: sequence bit, bits 0-6: 0000010 (packet type)
Byte 1: low byte of Gx sensor
Byte 2: high byte of Gx sensor
Byte 3: low byte of Gy sensor
Byte 4: high byte of Gy sensor
Byte 5: low byte of Gz sensor
Byte 6: high byte of Gz sensor
Byte 7: always 0

Calibration Data Packet 2 (Magnetic Field Sensor Reading):
Byte 0: bit 7: sequence bit, bits 0-6: 0000011 (packet type)
Byte 1: low byte of Mx sensor
Byte 2: high byte of Mx sensor
Byte 3: low byte of My sensor
Byte 4: high byte of My sensor
Byte 5: low byte of Mz sensor
Byte 6: high byte of Mz sensor
Byte 7: always 0

For each calibration measurement an acceleration packet is sent followed by a magnetic field packet. The two packets must be acknowledged separately.

Acknowledge Packet

An acknowledge packet consists of a single byte:
Byte 0: bit 7: sequence bit, bits 0-6: 1010101

The sequence bit of the acknowledge byte must match the sequence bit of the corresponding data packet.

Command Packets

The following commands may be sent from the PDA to the Disto:

Start Calibration Mode (1 byte):
Byte 0: 00110001

Stop Calibration Mode (1 byte):
Byte 0: 00110000

Start Silent Mode (1 byte):
Byte 0: 00110011

Stop Silent Mode (1 byte):
Byte 0: 00110010

Read Memory (3 bytes):
Byte 0: 00111000
Byte 1: low byte of address
Byte 2: high byte of address

Write Memory (7 bytes):
Byte 0: 00111001
Byte 1: low byte of address
Byte 2: high byte of address
Byte 3: data byte 0
Byte 4: data byte 1
Byte 5: data byte 2
Byte 6: data byte 3

Read and Write commands read or write 4 bytes of memory starting at the given address. Depending on the address, the memory accessed is RAM, configuration data (PIC EEPROM), or data store (external EEPROM). A memory read reply is sent by the Disto for each read and write command. The read reply following a write command should be used to check the correctness of the written data. There is no reply for the Start and Stop commands.

Reply Packets

The format of read reply packets is similar to that of the data packets.

Memory Read Reply (8 bytes):
Byte 0: 00111000 (packet type)
Byte 1: low byte of address
Byte 2: high byte of address
Byte 3: data byte 0
Byte 4: data byte 1
Byte 5: data byte 2
Byte 6: data byte 3
Byte 7: always 0

Address Space

The 16 bit address space used for read and write commands is divided into the following ranges:

0x0000 – 0x7FFF: Data store (External EEPROM)
0x8000 – 0x80FF: Configuration data (PIC EEPROM)
0x8100 – 0xBFFF: Reserved
0xC000 – 0xC0FF: RAM
0xC100 – 0xDFFF: Reserved
0xE000: Firmware Version (Major/Minor/0/0, read only)
0xE001 – 0xFFFF: Reserved

The data store in the external EEPROM contains a circular buffer to store the measured data. The 32Kbyte are divided into 4096 blocks of 8 bytes each. The layout of a single block is the same as that of the corresponding data packet. There is a single difference: bit 7 of byte 0 is interpreted as a ‘hot’ bit, it marks the data packets stored but not yet transmitted. Unused parts of the memory have a type byte (byte 0) of 0 or 0xFF. After writing the last block in memory, the address wraps around to the first block. The oldest data blocks are overwritten when no unused memory is available. There is always at least one unused block to mark the start and end of the queue.

The configuration store in the PIC EEPROM contains additional persistent information. The following locations are currently used:

0x00: User set operation mode
    bit 0: grad, bit 1: Bluetooth on, bit 2: compass on, bit 3: calibration mode, bit 4: silent mode
0x01: Calibration counter (used during calibration only)
0x08/0x09: Serial number (low/high)
0x10 – 0x27: G calibration coefficients
0x28 – 0x3F: M calibration coefficients

Appendix A: Basic Communication Code

```
unsigned byte input[8]; // input buffer
int oldType, oldDist, oldAzi, oldIncl;  // previous packet

ReadBytes(input, 0, 8); // receive 8 bytes
Byte type = input[0];
if ((type & 0x3F) == 1) { // measurement data
    int distance = input[1] + (input[2] << 8) + ((type & 0x40) << 10);
    short azimuth = (short)(input[3] + (input[4] << 8));
    short inclination = (short)(input[5] + (input[6] << 8));
    WriteByte(type & 0x80 | 0x55);  // send acknowledge byte
    if (type != oldType
      || distance != oldDist
      || azimuth != oldAzi
      || inclination != oldIncl) { // valid data
        Store(distance, azimuth, inclination); // store new data
        oldType = type;
        oldDist = distance;
        oldAzi = azimuth;
        oldIncl = inclination;
    }
} else {
    // handle other data packets
}
```