SOUNDNINE INC

Helping build successful monitoring systems

XT Sensors Version 2

2020-06-10

Table of Contents

1 XT Sensors 3

1.1 About Version 2 XT Sensors 3

1.2 Differences from Version 1 Sensors 3

1.3 Ultimodem Setup for Version 2 XT Sensors 3

1.4 Sample Ultimodem Configuration 3

2 Sampling and Retrieving Data 4

2.1 Data Retrieval Example 4

3 Data Messages 5

3.1 Message Types 6
3.1.1 Table of Message Types 6
3.1.2 X0 Format 6
3.1.3 X1 Format 7
3.1.4 C1 Format 7

3.2 CRC16 Codes 8
3.2.1 Preceding Spaces 8
3.2.2 CRC16 Example Code 8

1 XT Sensors

1.1 About Version 2 XT Sensors

Version 2 XT sensors evolved from the original XT and XTP sensors produced from 2015 to 2019. These devices are intended for low-cost high-accuracy real time data applications. They have no internal memory, operate over a simple inductive modem interface and generally do not have user-replaceable batteries.

There are now several different types of XT sensors including combinations of temperature, conductivity, pressure, acceleration, tilt and dissolved oxygen. There are therefore quite a few different data message types returned by these sensors. All of these data messages have certain common elements to facilitate efficient automated parsing.

1.2 Differences from Version 1 Sensors

Version 1 sensors maintained compatibility with the Sea-Bird Electronics inductive modem module (IMM). Version 2 abandons this compatibility – the Ultimodem is required to communicate with version 2 devices.

Version 1 sensors reported data at 1200 baud. Version 2 sensors respond at 4800 baud.

Version 1 device ID's and calibration coefficients could not be changed through the inductive interface. In version 2 devices these parameters can be changed.

Version 1 sensors required the FCL command to start sending a carrier signal before sending the XTP sampling command. With version 2 sensors the IM line must be idle for at least 5 seconds before sending the XT2 command.

1.3 Ultimodem Setup for Version 2 XT Sensors

The ultimodem must be configured to MOD 4 and TELMODE 3 to communicate properly with XT sensors. Usually WAKEUPSRC 1 is appropriate – this prevents the modem from waking in response to signals or noise on the inductive line.

1.4 Sample Ultimodem Configuration

```
<Config type='Ultimodem' sn='U0RG' v='0'>
<Hardware>
  <Assembly>50106.1</Assembly>
  <Firmware>ULTIMODEM V0.99F</Firmware>
</Hardware>
<Settings>
  baud=19200
  mod=4800 baud@19.2k (4)
  id=RG
  group=0
```

```
  wakeupsrc=RS232 Only (1)
  hostPrompt=S>
  modemPrompt=S9>
  cmdTO=60
  asyncRx=1
  telMode=3
  hostWakeup=2
  termFromHost=13
  termToHost=254
  thost0=0
  thost1=5
  thost2=3000
  thost3=12000
  thost4=500
  thost5=5
  tmodem2=500
  tmodem3=18000
  icd=1
  echo=1
  sflow=1
  debuglevel=0
  imfConfig=0
  setagc=32
</Settings></Config>
```

2 Sampling and Retrieving Data

The process of sampling and retrieving data is extremely simple:

1. Wait up to 1500ms if power to the modem was just enabled.
2. Send a character (usually \r) to wake the ultimodem
3. Wait 100mS for an S9> prompt from the modem
4. Send \r to clear the modem receive buffer.
5. Wait 100mS
6. Send XT2\r
7. Wait until either S9> prompt or the last expected data arrives. The modem waits five seconds from the end of the last sensor response before ending receive mode and giving the S9> prompt. The maximum duration of the XT2 command is usually 45 seconds – unless custom configured for systems with large numbers of sensors.

2.1 Data Retrieval Example

```
S9>
S9>xt2
...

#X1:06Q:1:25.6554,1982.512,10.376,8.536,1.766,2,-88.533,1.022,-
3.0,1FD3,1262,0,7:CF47*41262,32*186
#X0:099:2:25.8029,2171.071,1.765,0,-88.687,1.009,-
5.6,9FB5,3553,0,A:1C9C*41178,32*154
```

```
#X0:09C:3:25.9170,2036.946,1.767,1,-88.961,1.024,-
3.3,9CB7,3554,0,5:E83F*41528,32*154
#X0:06W:4:25.9977,1698.092,1.737,1,-88.530,1.023,-
6.7,7751,3552,0,A:AD86*41352,32*157
#C1:09N:5:23.3467,2083.432,0.0001,287981.00,10.439,8.356,4.737,38,-
89.274,0.988,-5.6,2B57,147D,0,35:29E2*41261,32*225
#C1:09P:7:23.5146,1839.387,0.0000,288802.75,10.356,6.114,4.726,49,89.552,0.97
6,-1.2,1F2E,979,0,1B:6A0D*41512,32*220
#C1:09W:9:23.3316,1913.233,0.0001,287427.88,10.370,8.044,4.854,93,89.653,0.99
7,-0.0,529A,2DD5,0,20: 5CA*41006,32*223

OK; 0 Events
S9>
```

The above transaction took about 4.5 seconds from the XT2 command to the end of the last data message and about 9.5 seconds from the XT2 command to the S9> prompt.

3 Data Messages

There are many types of data messages and more data message types will be added in the future.

All messages types have the following common format:

```
#[Msg type]:[Device MID]:[Device ID]:[Data payload]:[CRC16]
```

The Ultimodem appends data to each string to make a combined format:

```
#[Msg type]:[Device MID]:[Device ID]:[Data payload]:[CRC16]*[Sig Strength],[SSNS]*[timer]\r\n
```

Note the parameters added by the Ultimodem are likely to change as firmware evolves, but they will always be started with a '*' character and ended with a '\r'. We will be adding code to automatically verify received CRC codes and flag messages as verified or failed.

Data	Format	Description
MSG Type	Alphanumeric; 2-4 digits	
Device MID	Alphanumeric; 3 digits	Immutable identifier derived from the serial number of the sensor minus the initial X. if the MID is 09P then the sensor serial number is X09P
Device ID	Numeric; 1-99; 1 or two digits	Mutable device identifier. This specifies the order in which sensors send data messages. **Systems must not use more than one sensor with the same Device ID!** **In some cases Device ID's must be intentionally skipped to allow longer data messages.**

Data Payload		Completely dependent on the MSG type. The payload will almost certainly be comma separated values and must not contain the characters ':','\r','#', or '*'
CRC16	Hex, 4 digits, preceding space or zeros	CRC16 checksum value.
Sig Strength	Decimal	Signal strength as measured by the modem. This is a minimum signal strength. We expect to update the scale or this parameter.
SSNS	Decimal	Number of samples in the signal strength measurement
Timer	Decimal	Receive time in mS of the message. This may be used to help estimate power consumption and timing in systems with many sensors.

3.1 Message Types

The message type specifies the format and meaning of the data payload section of the message. Some message types require more time to transmit, so Device ID numbers must be skipped after these message types in systems using them.

3.1.1 Table of Message Types

Message Type	Description
X0	Temperature and acceleration (XT format)
X1	Temperature, acceleration and pressure (XTP format)
C0	Conductivity, Temperature and Acceleration
C1	Conductivity, Temperature, Acceleration and Pressure
T0	Test format
T1	Test format
T2	Test format
T3	Test format with conductivity
T4	Test format with DO

3.1.2 X0 Format

Sample message:

```
#X0:09C:3:25.9170,2036.946,1.767,1,-88.961,1.024,-3.3,9CB7,3554,0,5:E83F*41528,32*154
```

The data payload in this message is:

```
25.9170,2036.946,1.767,1,-88.961,1.024,-3.3,9CB7,3554,0,5
```

Parameter	Format	Value in example	Description
Temperature	%.4f	25.9170	Calibrated temperature

Thermistor	%.3f	2036.946	Raw thermistor resistance (ohms)
Battery voltage	%.3f	1.767	Battery Voltage
VB Drop	%d	1	Change in battery voltage under load in mV
Tilt	%.3f	-88.961	Tilt in degrees
Acceleration magnitude	%.3f	1.024	Magnitude of accelerometer reading in g
Uncal Temp	%.1f	-3.3	Uncalibrated instrument internal temperature
Wakeups	%X	9CB7	Num wakeups
Samples	%X	3554	Num samples
Comms	%X	0	Num Comms
Resets	%X	5	Num resets

3.1.3 X1 Format

Sample Message:

```
#X1:06Q:1:25.6554,1982.512,10.376,8.536,1.766,2,-88.533,1.022,-3.0,1FD3,1262,0,7:CF47*41262,32*186
```

In this message the data payload is:

```
25.6554,1982.512,10.376,8.536,1.766,2,-88.533,1.022,-3.0,1FD3,1262,0,7
```

Parameter	Format	Value in example	Description
Temperature	%.4f	25.6554	Calibrated temperature
Thermistor	%.3f	1982.512	Raw thermistor resistance (ohms)
Pressure	%.3f	10.376	Calibrated pressure (dbar)
Pressure Voltage	%.3f	8.536	Raw pressure sensor voltage (mV)
Battery voltage	%.3f	1.766	Battery Voltage
VB Drop	%d	2	Change in battery voltage under load in mV
Tilt	%.3f	-88.533	Tilt in degrees
Acceleration magnitude	%.3f	1.022	Magnitude of accelerometer reading in g
Uncal Temp	%.1f	-3.0	Uncalibrated instrument internal temperature
Wakeups	%X	1FD3	Num wakeups
Samples	%X	1262	Num samples
Comms	%X	0	Num Comms
Resets	%X	7	Num resets

3.1.4 C1 Format

Sample message:

```
#C1:09W:9:23.3316,1913.233,0.0001,287427.88,10.370,8.044,4.854,93,89.653,0.99
7,-0.0,529A,2DD5,0,20: 5CA*41006,32*223
```

In this message the data payload is:

```
23.3316,1913.233,0.0001,287427.88,10.370,8.044,4.854,93,89.653,0.997,-0.0,529A,2DD5,0,20
```

Parameter	Format	Value in example	Description
Temperature	%.4f	23.3316	Calibrated temperature
Thermistor	%.3f	1913.233	Raw thermistor resistance (ohms)
Conductivity	%.4f	0.0001	Conductivity (S/M)
Period	%.2f	287427.88	Raw conductivity period
Pressure	%.3f	10.370	Calibrated pressure (dbar)
Pressure Voltage	%.3f	8.044	Raw pressure sensor voltage (mV)
Battery voltage	%.3f	4.854	Battery Voltage
VB Drop	%d	93	Change in battery voltage under load in mV
Tilt	%.3f	89.653	Tilt in degrees
Acceleration magnitude	%.3f	0.997	Magnitude of accelerometer reading in g
Uncal Temp	%.1f	-0.0	Uncalibrated instrument internal temperature
Wakeups	%X	529A	Num wakeups
Samples	%X	2DD5	Num samples
Comms	%X	0	Num Comms
Resets	%X	20	Num resets

3.2 CRC16 Codes

The CRC calculation includes all bytes after the '#' and before the last ':'. Neither the '#' nor the ':' are included in the CRC.

3.2.1 Preceding Spaces

Note some sensors allow preceding spaces in the hex output of the CRC code. See the #C1:09W example above – the CRC value is " 5CA". Processing code should allow either preceding spaces or preceding zeros, as in " 5CA" or "05CA". The CRC will always be four digits.

3.2.2 CRC16 Example Code

The following code calculates a CRC16 value from a string.

```
const uint16_t crc16table[256] = {
0x0000, 0x1189, 0x2312, 0x329B, 0x4624, 0x57AD, 0x6536, 0x74BF,
0x8C48, 0x9DC1, 0xAF5A, 0xBED3, 0xCA6C, 0xDBE5, 0xE97E, 0xF8F7,
0x1081, 0x0108, 0x3393, 0x221A, 0x56A5, 0x472C, 0x75B7, 0x643E,
0x9CC9, 0x8D40, 0xBFDB, 0xAE52, 0xDAED, 0xCB64, 0xF9FF, 0xE876,
```

```
0x2102, 0x308B, 0x0210, 0x1399, 0x6726, 0x76AF, 0x4434, 0x55BD,
0xAD4A, 0xBCC3, 0x8E58, 0x9FD1, 0xEB6E, 0xFAE7, 0xC87C, 0xD9F5,
0x3183, 0x200A, 0x1291, 0x0318, 0x77A7, 0x662E, 0x54B5, 0x453C,
0xBDCB, 0xAC42, 0x9ED9, 0x8F50, 0xFBEF, 0xEA66, 0xD8FD, 0xC974,
0x4204, 0x538D, 0x6116, 0x709F, 0x0420, 0x15A9, 0x2732, 0x36BB,
0xCE4C, 0xDFC5, 0xED5E, 0xFCD7, 0x8868, 0x99E1, 0xAB7A, 0xBAF3,
0x5285, 0x430C, 0x7197, 0x601E, 0x14A1, 0x0528, 0x37B3, 0x263A,
0xDECD, 0xCF44, 0xFDDF, 0xEC56, 0x98E9, 0x8960, 0xBBFB, 0xAA72,
0x6306, 0x728F, 0x4014, 0x519D, 0x2522, 0x34AB, 0x0630, 0x17B9,
0xEF4E, 0xFEC7, 0xCC5C, 0xDDD5, 0xA96A, 0xB8E3, 0x8A78, 0x9BF1,
0x7387, 0x620E, 0x5095, 0x411C, 0x35A3, 0x242A, 0x16B1, 0x0738,
0xFFCF, 0xEE46, 0xDCDD, 0xCD54, 0xB9EB, 0xA862, 0x9AF9, 0x8B70,
0x8408, 0x9581, 0xA71A, 0xB693, 0xC22C, 0xD3A5, 0xE13E, 0xF0B7,
0x0840, 0x19C9, 0x2B52, 0x3ADB, 0x4E64, 0x5FED, 0x6D76, 0x7CFF,
0x9489, 0x8500, 0xB79B, 0xA612, 0xD2AD, 0xC324, 0xF1BF, 0xE036,
0x18C1, 0x0948, 0x3BD3, 0x2A5A, 0x5EE5, 0x4F6C, 0x7DF7, 0x6C7E,
0xA50A, 0xB483, 0x8618, 0x9791, 0xE32E, 0xF2A7, 0xC03C, 0xD1B5,
0x2942, 0x38CB, 0x0A50, 0x1BD9, 0x6F66, 0x7EEF, 0x4C74, 0x5DFD,
0xB58B, 0xA402, 0x9699, 0x8710, 0xF3AF, 0xE226, 0xD0BD, 0xC134,
0x39C3, 0x284A, 0x1AD1, 0x0B58, 0x7FE7, 0x6E6E, 0x5CF5, 0x4D7C,
0xC60C, 0xD785, 0xE51E, 0xF497, 0x8028, 0x91A1, 0xA33A, 0xB2B3,
0x4A44, 0x5BCD, 0x6956, 0x78DF, 0x0C60, 0x1DE9, 0x2F72, 0x3EFB,
0xD68D, 0xC704, 0xF59F, 0xE416, 0x90A9, 0x8120, 0xB3BB, 0xA232,
0x5AC5, 0x4B4C, 0x79D7, 0x685E, 0x1CE1, 0x0D68, 0x3FF3, 0x2E7A,
0xE70E, 0xF687, 0xC41C, 0xD595, 0xA12A, 0xB0A3, 0x8238, 0x93B1,
0x6B46, 0x7ACF, 0x4854, 0x59DD, 0x2D62, 0x3CEB, 0x0E70, 0x1FF9,
0xF78F, 0xE606, 0xD49D, 0xC514, 0xB1AB, 0xA022, 0x92B9, 0x8330,
0x7BC7, 0x6A4E, 0x58D5, 0x495C, 0x3DE3, 0x2C6A, 0x1EF1, 0x0F78
};

uint16_t crc16Calc(char *s){
      uint16_t crc = 0;
      while (*s>0){
            crc = (crc >> 8) ^ crc16table[(crc ^ *s++)&0xFF];
      }
      return crc;
}
```