

The Mote-Enabled GPS Server

V2.0 – August 1, 2005

Craig Pataky

For over ten years, GPS has been the most desirable source of outdoor positioning information. It follows that we want to utilize the GPS in our urban navigation project.

For some time, we've been casually experimenting with the GPS receiver of Crossbow's MTS420 sensor platform. Our results have been mixed and frustrated. It so happens that the MTS420 was designed in such a way that the energizing the GPS receiver would compromise the mote's ability to receive radio messages. This limitation is unacceptable in our application, as we require accurate positioning information to be broadcast on demand.

We overcame the limitation by modifying the MTS420 sensor board such that the step-up voltage regular U11 was removed, and pads 7 and 8 were shorted. The regulator was the source of the radio interference, and its removal allowed us to energize the GPS receiver and receive radio messages simultaneously. We have a good source of regulated 3.3V power, so U11 is superfluous to our application anyway.

Still, working hardware is only half the battle. Our next challenge was to create a well-defined mote-enabled package so that our projects could take advantage of our GPS solution both now and in the future. This document and the module it describes are the answer to that challenge.

The Mote/GPS is nothing more—or less—than a Crossbow Mica2 coupled with a modified MTS420 board. The module can communicate with other equipment through either USART0 or the radio, and additional status information is indicated on the LEDs.

The Basic Radio Packet Interface

Radio communications typically consist of a query to be generated by an interrogating mote, and a response to be generated by the MoteGPS to that query. Both messages are in compliance with the TinyOS message structure.

The key fields of a query are type and length. The type must be 4, the length must be 2. All other fields, including the CRC, are ignored.

The response to the query is a TOSMsg with the following important fields:

<u>Field</u>	<u>Value</u>
TosMsg.addr	FFFFh
TosMsg.type	5
TosMsg.length	10h
TosMsg.crc	Not valid.
TosMsg.data	The following structure comprises the data:

<u>Field</u>	<u>Description</u>	<u>Size</u>
src;	Always FFFFh.	WORD
lat;	Latitude, in degrees.	WORD
lat_man;	1/100000 of degrees.	DWORD
lng;	Longitude, in degrees.	WORD
lng_man;	1/100000 of degrees.	DWORD
north;	1=north, 0=south.	BYTE
east;	1=east, 0=west.	BYTE

As an example, suppose we were at 44 and 4609/100000ths degrees North, 123 and 7056/100000ths degrees West. The structure would be filled in as follows:

<u>Field</u>	<u>Description</u>	<u>Value</u>
src;	Always FFFFh.	FFFFh
lat;	Latitude, in degrees.	44
lat_man;	1/100000 of degrees.	4609
lng;	Longitude, in degrees.	123
lng_man;	1/100000 of degrees.	7056
north;	1=north, 0=south.	1
east;	1=east, 0=west.	0

This structure allows any device to easily calculate highly accurate positioning information. Consider that in Eugene, one degree longitude is 263100 feet. It follows that 1/100000 of a degree at this longitude is 2.631 feet. Similarly, one degree latitude in Eugene is 364500 feet, resulting in 1/100000 of a degree being 3.645 feet.

If the GPS receiver can not determine geographic position for any reason, such as a building obstruction or during initial power-on warm-up, all data fields except source are loaded with zero. The source field remains FFFFh at all times.

The Advanced Radio Packet Interface

New to Version 1.2 are APIs designed to work hand-in-hand with modes 3 & 4 of the Mote Modem. This makes it easy for people to interact with the GPS without requiring specific knowledge of TinyOS messages.

Using a Mote Modem in mode 3 or 4, you can query the GPS position by sending the string "GPS_POSITION" and the Mote enabled GPS will respond with position information (if available) in the following manner:

```
$#44#4589#123#7052#1#0
GPS_LAT=44.04589N
GPS_LNG=123.07052W
```

The textual responses GPS_LAT and GPS_LNG represent degrees and fractional degrees, along with N/S or E/W indicators. The '#' delimited response is described as follows:

<u>Field</u>	<u>Description</u>	<u>Example</u>
Attention	Always \$	\$
lat	Latitude, in degrees.	44
lat_man	1/100000 of degrees.	4589
lng	Longitude, in degrees.	123
lng_man	1/100000 of degrees.	7052
north	1=north, 0=south.	1
east	1=east, 0=west.	0
[cr/lf]	Every message ends with carriage return, linefeed.	

This response assumes that valid GPS information is available. If the GPS unit does not have a position lock, then the response look more like this:

```
$#0#0#0#0#0#0#0
GPS_INVALID
```

At the time of this writing, the Mote GPS can be commanded to pause automatic Mode 1 transmissions. The device can also be pinged and queried for time, power. The commands and responses are shown below.

<u>Query:</u>	<u>Response:</u>	<u>Use:</u>
GPS_PING	GPS_PING	Determine if GPS is in area.
GPS_POWER	GPS_MV=nnnn	Determine GPS battery voltage, 3300 typical.
GPS_TIME	GPS_TIME=nnnnnnn	Get the UTC time, hhmmss, according to GPS.
GPS_PAUSE	GPS_PAUSING	Suspends automatic Model transmission for about 30 seconds. See <i>Command and Control</i> .

The Serial Interface on USART0

By default, the mote's USART0 has been programmed to operate at 9600 baud, N81. You can connect the device to a PC by using the Crossbow MIB510 programming board and a program such as HyperTerminal.

Alternatively, a TTL connection is available on J21 or J22. Attach to pin 28 as transmit data (data from the mote to an external device). This connection is meant for embedding the unit into a larger system, as you cannot connect directly to a PC using this method—a TTL to RS-232 adapter is required. If you really want to connect to a PC and not use the MIB510, adapters are readily available from HVW Technologies (www.HVWtech.com) for only \$10.

When the Mote/GPS first powers-up, the following sign-on message is displayed:

```
APP: GPS MODULE
Jul  5 2005
14:36:51
```

The sign-on message is the best opportunity to verify that the serial connection has been made correctly, and that the Mote/GPS module has been programmed with the correct version of software.

The Mote/GPS will then automatically send position messages over USART0 every two seconds. Every message begins with a dollar sign [\$] and ends with a carriage return-linefeed pair [cr/lf].

If the GPS receiver can not determine geographic position for any reason, such as a building obstruction or during initial power-on warm-up, the following message will be sent: \$GPS_INVALID[cr/lf]

On the other hand, if valid position information is being received by the GPS receiver, the information is sent out USART0 in three different formats. First in NMEA-0183 format, then the '#' delimited format mentioned under the Advanced Radio Packet Interface, and finally by an easily readable text format.

The NMEA-0183 Format

An example NMEA-0183 message, along with a format table is provided. Note how each field is separated with a comma.

Example:

```
$GPGGA,234422.544,4402.7535,N,12304.2316,W,1,04,1.7,138.0,M,,,,0000*15
```

<u>Name</u>	<u>Example</u>	<u>Description</u>
Attention	\$	Message Preamble
Message ID	GPGGA	Message Header
UTC Time	234422.544	ddmmss.sss
Latitude	4402.7535	ddmm.mmmm
N/S Indicator	N	N=North or S=South
Longitude	12304.2316	dddmm.mmmm
E/W Indicator	W	E=East or W=West
Position Fixed	1	0=GPS Position not valid, 1-3=Valid
Satellites Used	4	Range 0-12 satellites in view
HDOP	1.7	Horizontal Dilution of Precision
MSL Altitude	138.0	Altitude, in meters.
Altitude Units	M	M=Meters
Unused		Unused
Unused		Unused
Unused		Unused
Unused	0000	Unused
Checksum	*15	
[cr/lf]		Every message ends with carriage return, linefeed.

The '#' Delimited Alternative Format

Imagining a situation where a developer more familiar with our radio message format might want to connect directly to USART0 as an option, we provide this alternative message so that the maximum amount of code can be reused when switching between wired and radio connections.

An example of our alternative format, along with a format table, is provided. Note how each field is separated with the # sign.

Example:

```
$#44#4589#123#7052#1#0
```

<u>Field</u>	<u>Description</u>	<u>Example</u>
Attention	Always \$	\$
lat	Latitude, in degrees.	44
lat_man	1/100000 of degrees.	4589
lng	Longitude, in degrees.	123
lng_man	1/100000 of degrees.	7052
north	1=north, 0=south.	1
east	1=east, 0=west.	0
[cr/lf]	Every message ends with carriage return, linefeed.	

The Text Format

In strictly human readable terms, the text format provides the easiest way to get a handle on the GPS position and status. Along with the NMEA and '#' delimited messages, the following messages sent out USART0 every two seconds.

\$GPS_MV=nnnn	States the GPS battery voltage, 3300 typical.
\$GPS_TIME=nnnnnnnn	Get the UTC time, hhmmss, according to GPS.
\$GPS_LAT=nn.nnnnn	States the GPS latitude in degrees and fractionals.
\$GPS_LNG=nnn.nnnnn	States the GPS longitude in degrees and fractionals.
\$GPS_INVALID	States that GPS information is not valid. This message is not sent when the GPS has an accurate position lock.

Status Information on LEDs

The green LED toggles when a message is sent out USART0.

The red LED is off when the GPS positioning information is valid.

The red LED is on when GPS positioning information is not valid.

The yellow LED toggles when a GPS request is received on the radio.

All LEDs stay on if the MTS-420 is not connected to the mote on power-up.

Command and Control

The Mote GPS can be controlled using familiar AT-style command strings, similar in spirit to a Hayes smart modem. From any operating mode, the device can be put into a "command" mode by sending it the string "ATCOMMAND" (without the quotes) on USART0, followed by a carriage return [cr] character. The device will enter command mode only if additional characters are not received on USART0 within three seconds. This period of silence enables the device to discriminate between a bonafide command and situations where the command string is coincidentally part of a larger transmission.

It is envisioned that a systems integrator will configure the device using a terminal program such as HyperTerminal. When the integrator enters the command string "ATCOMMAND" (without the quotes) followed by a carriage return, the following is displayed:

```
COMMAND MODE BEGIN
OK
```

The integrator could now get the menu of commands available. This is accomplished by typing "ATH" followed by a carriage return. The following is displayed:

```
COMMANDS :
  ATBAUD4800 .. SETS 4800 BAUD
  ATBAUD9600 .. SETS 9600 BAUD
  ATBAUD19200 .. SETS 19200 BAUD
  ATBAUD38400 .. SETS 38400 BAUD
  ATBAUD57600 .. SETS 57600 BAUD
  ATMODEn .. SETS OPERATING MODE 0 or 1
  ATFHn .. SETS FAIL HALT MODE 1=ENABLE, 0=DISABLE
  ATLEDn .. SETS LED MODE 1=ENABLED 0=DISABLE
  ATDEFAULT .. RESTORES SYSTEM DEFAULTS
  ATSHOW .. SHOW CURRENT SETTINGS
  ATSAVE .. EXIT COMMAND MODE & SAVE
  ATEXIT .. EXIT COMMAND MODE WITHOUT SAVING
  ATH OR AT? .. THIS SCREEN
OK
```

Baud rate commands are self explanatory. The others we'll discuss here.

The setting ATFH, when enabled, causes the device to halt if the MTS-420 unit is not connected during power-on. This is a troubleshooting option that makes it easier to determine if a failure is with the MTS-420 control logic or with the GPS unit itself.

The setting ATLED either enables or disables the LEDs. Though it is usually most desirable to have LEDs enabled, it might be advantageous to disable them in some situations to save power.

Use ATMODE set the operating mode. Mode 0 is the default mode, where the device will not send messages out the radio unless queried. In Mode 1, the Mote GPS will automatically transmit GPS_TIME or, if time is not available, GPS_MV messages. These messages provide an easy way for passive listening devices to determine their proximity to the Mote GPS. These messages can be temporarily suspended by sending the Mote GPS a GPS_PAUSE message.

Use ATSHOW to display current settings. An example show is as follows:

```
SETTINGS:
MODE=1
BAUD=9600
ENABLE LEDS=1
ATFH=0
OK
```

Command ATSAVE is used to save the settings into flash and exit command mode. The unit will automatically reset and the new settings will take effect.

Command ATEXTIT is used to exit command mode without saving the settings into flash. The unit will automatically reset and the old settings will be restored.

Appendix A: Verifying Checksum of NMEA-0183 Messages

There are times when it may be desirable to verify the integrity of the NMEA message format. Examples would include transmission through very long serial IO lines, placement of the device within an electrically noisy environment, or any other situation where signaling integrity cannot be guaranteed.

An algorithm for verifying NMEA messages is provided as follows.

```
BYTE GPS_CharToHex(BYTE y)
{
  if ((y>='0')&&(y<='9')) y=y-'0';
  if ((y>='a')&&(y<='z')) y=y-'a';
  if ((y>='A')&&(y<='Z')) y=y-'A';
  return(y);
}

BOOL GPS_VerifyGPSChecksum(BYTE* pSentence)
{
  DWORD dw;
  WORD Checksum;
  BYTE x,y,z;

  Checksum=0;
  x=0;
  while (pSentence[x]!=0)
  {
    switch(pSentence[x])
    {
      case '$': // Ignore the dollar sign, if present
        break;

      case '*': // Stop processing before at the asterisk
        y=GPS_CharToHex(pSentence[x+1]);
        z=GPS_CharToHex(pSentence[x+2]);
        dw=y;
        dw=dw<<4;
        dw|=z;
        if (dw!=Checksum) return(FALSE);
        return(TRUE);

      default: // XOR the checksum with this character's value
        Checksum = Checksum ^ pSentence[x];
        break;
    }
    x++;
    if (x>75) return(FALSE); //This catches us if the whole thing is bizarre.
  }
  return(FALSE); //And we get here if the end of string is found but no asterisk.
}
```