Application Notes
TT8750+_AN004

TT8750+
Position Message
Decoding

Revision 1.0

07/18/2012

Index

General Message Structure:

The message we received from the TT8750Plus device can be separated into two parts: the API Header, and the Device Data. Below is a diagram of this structure:

Message Data

API Header	Device Data

API Header (UDP messages):

The base API message header is 4 bytes long for commands sent via UDP/IP.

Messages sent via UDP/IP has the following API header

Bytes 0 - 1: 16–bit API number (**refer to Table 1 below**)

Bytes 2: 8–bit Command Type information. This value determines the type of message being sent or received by the host (**refer to Table 1 below**)

Bytes 3: **Message type**: bit7, bit6, bit5, bit4 inform the message type. Please refer to the parameter <message type> in the command AT$TTMSGMASK.

Need Ack: bit3, bit2, bit1, bit0 inform whether the unit waits for the acknowledgement of the message.

0: do not wait for acknowledgement

1: wait for the acknowledgement

Bytes 4: API Optional Header Size. This field defines the size of the API Optional Header in Bytes. API Optional Header Size is up to the value of bit 27 of message mask.

Bytes 5 thru (5+m): API Optional Header. If it is 1, API optional header size will be 4 and the message mask will follow API optional header size. If it is 0, API optional header size will be 0. And this is only for the report from the unit to the server

API Number (bytes 0-1):

Two byte value that indicates the nature of the message, e.g. an Unsolicited Msg or ACK.

Command Type (byte 2):

One byte value that indicates that the command is requesting, e.g. Read or Write.

MSG type (byte 3):

Indicates the kind of message that is being generated. It is broken into two parts, the Message Type indicator, and the Need Ack indicator.

Message Type indicator: Bit 7, bit 6, bit 5, and bit 4. Possible decimal values are from 0-4.

Message Type Indicator
1: Position report messages
2: Counter messages
3: Variables messages
4: Geofence messages

Need Ack indicator: Bit 3, bit 2, bit 1, and bit 0. Possible decimal values are 0 or 1.

Need Ack Indicator
0: Do not wait for acknowledgement
1: Wait for the acknowledgement

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
|←-------**Message Type Indicator**-----→|				|←----------**Need Ack Indicator**--------→|		

API Optional Header Size (byte 4)
One byte value that indicates the size of the optional header. May have a maximum value of 255.

API Optional Header (byte 5 – m)
The size of the optional header is taken from byte 4 above.

API Header Table:

Below is the API Header table showing the combinations of the API Number, Command Type, Message Type Indicator, and API Optional Header Size.

Look first at the Command Type and find the corresponding API number.

API number (values given below are decimal)		**Command Type**	MSG type / Need ACK	API Optional Header Size	
Byte – 0	Byte – 1	Byte – 2	Byte – 3	Byte – 4	Direction
0 – 4 Reserved 5 – GPS Binary Read* 6– 65535 Reserved		0 (Read Request)		0	Modem ← OTA
0 – Unsolicited Msg Request 1 – 9 Reserved 10 – ACK 11 – Password 12 – 65535 Reserved		1 (Write Request)		0	Modem ← OTA

API number (values given below are decimal)		**Command Type**	MSG type / Need ACK	API Optional Header Size	
Byte – 0	Byte – 1	Byte – 2		Byte – 3	Direction
0 – Unsolicited Msg 1 – 3 Reserved 4 – ASCII Event Data (Param2>=256)* 5 – Binary Event Data* 6 – Reserved 7 – ASCII TAIP Data* 8 – $MSGSND Data 9 – Reserved 10 – ASCII Event Data (Param2<256)		2 (General Status Information)		(Size of API Optional Header)	Modem → OTA
Echo first 2 bytes of an incoming data		3 (Error)		(Size of API Optional Header)	Modem → OTA
0 – 65535		4 (AT Command)		0	Modem ← OTA
Echo first 2 bytes of an incoming AT command request		5 (AT Command Response)		(Size of API Optional Header)	Modem → OTA
First 2 bytes of AT$SNDMG command in ASCII format		6		(Size of API Optional Header)	Modem ←→ OTA
First 2 bytes of AT$SNDMG command in Binary format		7		(Size of API Optional Header)	Modem ←→ OTA
10		8		0	Modem ID / Modem → OTA (TCP only)

Table 1

Device Data:

This section describes the structure of the data that the device will transmit to the server. As indicated in the section API Header, the device can transmit 5 different Message Type Indicators. These indicators are as follows:

Message Type Indicator	
1:	Position report messages
2:	Counter messages
3:	Variables messages
4:	Geofence messages

Based on the Message Type Indicator that the device generated, the message needs to be decoded accordingly. Below are the steps to decode the various Message Type Indicators available on the TT8750Plus device.

All messages that the device will generate in our application will be in binary.

Message Type Indicator 1 – Position Report Message:

The message mask value is obtained as a result of selecting individual bit-fields from the table below.

Assume all fields will be used.

Message Format Mask Table:

Message Mask	Description
Bits 0	1 = send all data generated as a result of this table in Binary format 0 = send all data generated as a result of this table in ASCII format
Bits 1	1 = add parm1 data to UDP message (4 – bytes in Binary format, 11 – bytes of data in ASCII format) 0 = do not add parm1 data to outbound UDP message
Bits 2	1 = add $MDMID value (22 – bytes of ASCII data – irrespective of Bit– 0 setting) 0 = do not add $MDMID value
Bits 3	1 = add $IOGPA (GPIO data) in ASCII-HEX format (2 – bytes in Binary format, 6 – bytes in ASCII format) 0 = do not add GPIO direction and data value.
Bits 4	1 = add Analog input 1 to UDP message (2 – bytes in Binary format, 5 – bytes of data in ASCII format) 0 = do not add Analog input 1 data to outbound UDP message
Bits 5	1 = add Analog input 2 to UDP message (2 – bytes in Binary format, 5 – bytes of data in ASCII format) 0 = do not add Analog input 1 data to outbound UDP message
Bits 6	1 = Message is stored in non-volatile memory until it can be sent, regardless of network status. 0 = Code checks network status before storing message in non-volatile memory. If it appears that the message can be sent out immediately (network status is clear and message queue has few or no messages pending), the message is stored in the non-volatile message queue until it can be sent. Otherwise, the message is deleted.
Bits 7	1 = add input <function category> number (1 – byte in binary format, 3 – bytes in ASCII format) 0 = do not add input <function category> number
Bits 8	1 = add GPS data (3 – bytes of Date information in Binary format or up to 80 – bytes of $GPGGA NMEA message if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 9	1 = add 1-byte of STATUS information in Binary 0 = do not add this particular field of GPS data
Bits 10	1 = add GPS data (4 – bytes of Latitude information in Binary format or up to 80 – bytes of $GPGSA NMEA message if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 11	1 = add GPS data (4 – bytes of Longitude information in Binary format or

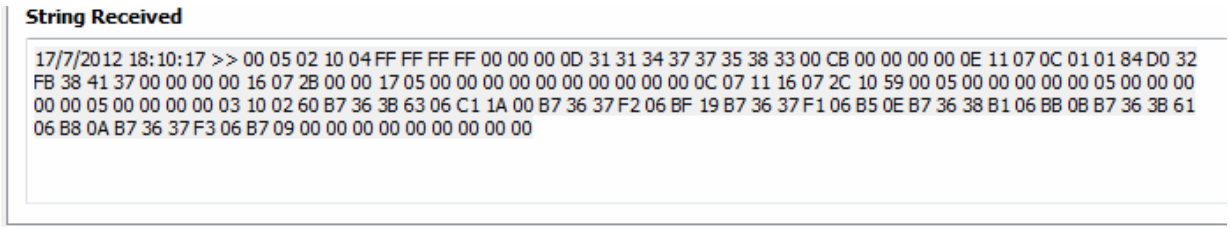
	up to two 80 – bytes of $GPGSV NMEA message if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 12	1 = add GPS data (2 – bytes of Velocity information in Binary format or up to 80 – bytes of $GPRMC NMEA message if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 13	1 = add 2-bytes of HEADING information in Binary 0 = do not add this particular field of GPS data
Bits 14	1 = add GPS data (3 – bytes of Time information in Binary format or 0 bytes if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 15	1 = add GPS data (3 – bytes of Altitude information in Binary format or 0 bytes if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 16	1 = add GPS data (1 – byte of Number Of Satellites In View information in Binary format or 0 bytes if Bit-0 is set to 0) 0 = do not add this particular field of GPS data
Bits 17	1 = add battery level percentage (2 - bytes of in Binary format or 3 - bytes if Bit-0 is set to 0) 0 = do not add this particular field
Bits 18	1 = send this OTA message via SMS when GPRS services is not available 0 = send this OTA message via GPRS only
Bits 19	1 = send Last Valid GPS data if current data is invalid 0 = send current GPS data – valid or invalid
Bits 20	1 = add Trip Odometer reading (4 - bytes of Trip Odometer information in Binary format or 11 - bytes if Bit-0 is set to 0) 0 = do not add this particular field of GPS data NOTE: The Trip Odometer is associated with the AT$TTTRODOM command.
Bits 21	1 = add Odometer reading (4 - bytes of Trip Odometer information in Binary format or 11 - bytes if Bit-0 is set to 0) 0 = do not add this particular field of GPS data NOTE: The Trip Odometer is associated with the AT$TTODOM command.
Bits 22	1 = add RTC time (6 – bytes of RTC time in Binary format or 13 – bytes if Bit-0 is set to 0) 0 = do not add RTC time with GPS data
Bits 23	1 = Replace/append device id field with 10-byte device id (including one leading and one ending space character) if bit-0 is set to 0. Replace/append it with 8-bytes long device id value if bit-0 is set to 1 (no leading or ending space characters in binary mode.) (NOTE: bit-22 setting overrides bit-2 setting) 0 = Sent the device id as defined by Bit-2
Bits 24	1 = add battery level percentage (2 - bytes of in Binary format or 3 - bytes if Bit-0 is set to 0) 0 = do not add this particular field
Bits 25	1 = add GPS overspeed data (18 – bytes of Odometer information in Binary format or 6 to 18 – bytes if Bit-0 is set to 0). Binary format: xxyyzz: xx is speed specified by AT$TTGPSOSI (unit: knots); yy is the maximum speed incurred during the interval (unit: knots, 1/10 knot accuracy); zz is the interval duration (unit: seconds);

	ASCII format: " x y z": space delineated, length of each field varies with its value 0 = do not add this particular field of GPS data
Bits 26	1 = Add cell information as follows: If Binary format (Bit0=1) is selected, please refer to the "Bit 25 Binary Format" table in section (Bit 26 Binary Format Table) If ASCII format (Bit0=0) is selected please refer to the "Bit 25 ASCII Format" table in section (Bit 26 ASCII Format Table) 0 = Do not add cell information
Bit 27	1 = Add Position report mask 0 = do not add Position report mask
Bits 28	1 = Add sequence number 0 = Do not add sequence number
Bits 29 - 31	Reserved

Raw Data Example:

Format: 11111111111111111111111111111111
Decimal: 4294967295

HEX Data:



HEX Data Breakup Example:

00 05 02 10 04 FF FF FF FF 00 00 00 0D 31 31 34 37 37 35 38 33 00 CB 00 00 00 00 0E 11 07 0C 01 01 84 D0 32 FB 38 41 37 00 00 00 00 16 07 2B 00 00 17 05 00 32 00 00 00 00 00 00 02 4E 0C 07 11 16 07 2C 10 59 00 05 00 00 00 00 00 05 00 00 00 00 00 05 00 00 00 00 03 10 02 60 B7 36 3B 63 06 C1 1A 00 B7 36 37 F2 06 BF 19 B7 36 37 F1 06 B5 0E B7 36 38 B1 06 BB 0B B7 36 3B 61 06 B8 0A B7 36 37 F3 06 B7 09 00 00 00 00 00 00 00 00 0C

Index	Description	Size	Value (HEX)	Meaning
0	API Header (5 bytes)	5 bytes	00 05 02 10 04	(From Table 1) API Number: (00 05)H -> Binary Event Data (2 bytes) Command Type: (02)H -> General Status Information (1 byte) MSG Type: (10)H -> Broken into two parts: (10)h = (00010000)b (1 byte) Message Type Indicator (bit 7,6,5 & 4) -> (0001)b = 1d => Position Message Need Ack Indicator (bit 3,2,1 & 0) -> (0000)b = 0d = No Ack needed. (If 1, refer to section "ACK Messages") API Optional Header Size: (04)H -> Optional Header includes 4 bytes of information.
1	Report Mask (only added when bit 27 is active on Mask)	4 bytes	FF FF FF FF	This is the Mask Code of the message. Convert the entire string into decimal. (FF FF FF FF)H = 4294967295
2	Param1 (Status code)	4 Bytes	00 00 00 0D	This is the Status code of the message. Convert the entire string into decimal. (00 00 00 0D)H = 10d
3	Device ID(if bit 23 is not active then field is 22 bytes)	8 bytes	31 31 34 37 37 35 38 33	This is the device ID of the device. First and Last bytes are spaces. Convert each byte individually to ASCII. (31)H (31)H (34)H (37)H (37)H (35)H (38)H (33)H = 11477583
4	IO Data	2 bytes	00 CB	I/O Data from the device. Convert to binary to get the status of the I/Os. (00 CB)H=(011001011)b Bit0 |←------→ | Bit8 Bit 0 = 1 (Input 1) Bit 1 = 1 (Input 2) Bit 2 = 0 (Output 1)

				Bit 3 = 1 (Main Power State) Bit 4 = 0 (Output 2) Bit 5 = 0 (Output 3) Bit 6 = 1 GSM led Bit 7 = 1 GPS led Bit 8 = 0 (Ignition State)
5	Analog Input 1	2 bytes	00 00	Analog Input 1 (in mvolts)
6	Analog Input 2	2 bytes	00 00	Analog Input 2 (in mvolts)
7	Function Category	1 byte	0E	This is why the message was triggered. Convert entire string to decimal: (0E)H = 14d (14 is the input function category for a timer)
8	GPS Date	3 bytes	11 07 0C	GPS Data that the message was generated. Convert each byte individually to ASCII. (11)H (07)H (0C)H = 17 (day) 7 (month) 12 (year)
9	GPS Status	1 byte	0 1	GPS Status from the device. Convert to decimal. 1 = Valid Position
10	GPS Latitude	4 bytes	01 84 D0 32	Refer to section "Lat/Long Conversion" for conversion instructions.
11	GPS Longitude	4 bytes	FB 38 41 37	Refer to section "Lat/Long Conversion" for conversion instructions
12	GPS Velocity	2 bytes	00 00	The velocity of the device. Convert to decimal, and divide by 10 to get the velocity in knots. (00 00)H = 0D 0D/10 = 0.0 knots
13	Heading	2 bytes	00 00	Heading of the device. Convert to decimal, and divide by 10 to get the heading in degrees. (00 00)H = 0D 0D/10 = 0.0 degrees
14	GPS Time	3 bytes	16 07 2B	The GPS UTC time that the message was generated. Convert each byte individually to decimal. (16)H (07)H (2B)H = 22 (hours) 07 (minutes) 43 (seconds)

15	GPS Altitude	3 bytes	00 00 17	Altitude convert from Hex to decimal number (00 01 43 42)H = 82754D If Altitude is negative first substract (FF FF FF)H - (Negative Altitude)H= XXH then convert to decimal XXH = YYD
16	No. of Satellites	1 byte	0 5	The number of satellites seen by the device. Convert to decimal. (05)H = 5 satellites
17	Battery level percentage	2 bytes	00 32	The Battery percentage level. (00 32)H = 50D 50/100= 0.5%
18	Trip Odometer	4 bytes	00 00 00 00	The odometer reading of the device. Convert to decimal to obtain the odometer in meters. (00 00 00 00)H = 0D meters
19	Odometer	4 bytes	00 00 00 00	The odometer reading of the device. Convert to decimal to obtain the odometer in meters. (00 00 02 4E)H = 590D meters
20	RTC Time	6 bytes	0C 07 11 16 07 2C	The RTC time of the device when the message was generated. Convert each byte individually. (0C)H (07)H (11)H (16)H (07)H (2C)H = 12 (year) 7 (month) 17 (day) 22 (hour) 7 (minutes) 4 (seconds)
21	Battery Voltage Level	2 bytes	10 59	Battery voltage value of the main power supply. Convert to decimal and divide by 1000. (10 59)H = 4185D 4185D/1000 = 4.185 V
22	GPS overspeed	18 bytes	00 05 00 00 00 00 00 05 00 00 00 00 00 05 00 00 00 00	The velocity of the device. Convert to decimal, and divide by 10 to get the velocity in knots. The format goes as follows xx xx yy yy zz zz: xx xx is speed specified by AT$TTGPSOSI (unit: knots); yy yy is the maximum speed incurred during the interval (unit: knots, 1/10 knot accuracy);

				zz zz is the interval duration (unit: seconds); (00 05)H = 5D 5D/10 = 0.5 knots (00 00)H = 0D 0D/10 = 0.0 knots (00 00)H = 0 seconds
23	Cell information	54 bytes	03 10 02 60 B7 36 3B 63 06 C1 1A 00 B7 36 37 F2 06 BF 19 B7 36 37 F1 06 B5 0E B7 36 38 B1 06 BB 0B B7 36 3B 61 06 B8 0A B7 36 37 F3 06 B7 09 00 00 00 00 00 00 00	Refer to Binary Format Table. No conversion needed.
24	Sequence No.	2 bytes	00 0C	Sequence number of the message (00 0C)H = 12D

Binary Format Table:

Bit 25 Binary Format Table	
MCCMNC	Mobile Country Code Mobile Network Code (2 bytes)
Serving Cell LAC	Location Area Code (2 bytes)
Serving Cell CellID	Cell Tower Identification (2 bytes)
Serving Cell ARFCN	Absolute radio frequency channel number (2 bytes)
Serving Cell signal strength	Cell signal strength (1 byte)
Serving Cell timing advance	Cell timing advance (1 byte)
Neighbor Cell 0 LAC	LAC for Neighbor Cell Tower 0 (2 bytes)
Neighbor Cell 0 CellID	CellID for Neighbor Cell Tower 0 (2 bytes)
Neighbor Cell 0 ARFCN	ARFCN for Neighbor Cell Tower 0 (2 bytes)
Neighbor Cell 0 rxlev	Signal quality for Neighbor Cell Tower 0 (1 byte)
Neighbor Cell 1 LAC	LAC for Neighbor Cellular tower 1 (2 bytes)
Neighbor Cell 1 CellID	CellID for Neighbor Cell Tower 1 (2 bytes)
Neighbor Cell 1 ARFCN	ARFCN for Neighbor Cell Tower 1 (2 bytes)
Neighbor Cell 1 rxlev	Signal quality for Neighbor Cell Tower 1 (1 byte)
Neighbor Cell 2 LAC	LAC for Neighbor Cellular tower 2 (2 bytes)
Neighbor Cell 2 CellID	CellID for Neighbor Cell Tower 2 (2 bytes)
Neighbor Cell 2 ARFCN	ARFCN for Neighbor Cell Tower 2 (2 bytes)
Neighbor Cell 2 rxlev	Signal quality for Neighbor Cell Tower 2 (1 byte)
Neighbor Cell 3 LAC	LAC for Neighbor Cellular tower 3 (2 bytes)
Neighbor Cell 3 CellID	CellID for Neighbor Cell Tower 3 (2 bytes)
Neighbor Cell 3 ARFCN	ARFCN for Neighbor Cell Tower 3 (2 bytes)
Neighbor Cell 3 rxlev	Signal quality for Neighbor Cell Tower 3 (1 byte)
Neighbor Cell 4 LAC	LAC for Neighbor Cellular tower 4 (2 bytes)
Neighbor Cell 4 CellID	CellID for Neighbor Cell Tower 4 (2 bytes)
Neighbor Cell 4 ARFCN	ARFCN for Neighbor Cell Tower 4 (2 bytes)
Neighbor Cell 4 rxlev	Signal quality for Neighbor Cell Tower 4 (1 byte)
Neighbor Cell 5 LAC	LAC for Neighbor Cellular tower 5 (2 bytes)
Neighbor Cell 5 CellID	CellID for Neighbor Cell Tower 5 (2 bytes)
Neighbor Cell 5 ARFCN	ARFCN for Neighbor Cell Tower 5 (2 bytes)
Neighbor Cell 5 rxlev	Signal quality for Neighbor Cell Tower 5 (1 byte)

Lat/Long Conversion:

If Lat or Long is > 7F FF FF FF, then it has a negative degree

Part 1: Positive Coordinate (Latitude Example)

Lat: **$(01\ 84\ CF\ 6A)_H$**

Since Lat is <= 7F FF FF FF, the latitude is positive

Step 1 – Convert to decimal: ddmm.mmmm
Lat_decimal = 25481066

Step 2 – Take last 6 digits and divide by 10000 to get the minutes. The remaining digits to the left are the degrees.
dd = 25
mm.mmmm = 48.1066

Step 3 – Divide mm.mmmm by 60 to convert to degrees
mm.mmmm/60 = 48.1066/60 = 0.801777

Step 4 – Add dd + mm.mmmm/60 to get the full Latitude degrees:

Lat_full = **dd** + **mm.mmmm/60**
25 + 0.801777 = +25.801777

Part 2: Negative Coordiante (Longitude Example)

Long: **$(FB\ 38\ 41\ 37)_H$**

Since Lat is > 7F FF FF FF, the latitude is negative

Step 1 – 2's complement
$Long\_comp = (FF\ FF\ FF\ FF)_H - Long$
$Long\_comp = (FF\ FF\ FF\ FF)_H - (FB\ 38\ 41\ 37)_H = (04\ C7\ BE\ C8)_H$

Step 2 – Convert to decimal: ddmm.mmmm
Lat_decimal = 80199368

Step 3 – Take last 6 digits and divide by 10000 to get the minutes. The remaining digits to the left are the degrees.
dd = 80
mm.mmmm = 19.9368

Step 4 – Divide mm.mmmm by 60 to convert to degrees
mm.mmmm/60 = 19.9368/60 = 0.33228

Step 5 – Add dd + mm.mmmm/60 to get the full Latitude degrees, and put a negative sign:

Lat_full = **dd** + **mm.mmmm/60**
80 + 0.33228= (-)80.33228