Application Note AN0008

Data Logging Extension

For

Venus 8 GPS Receiver

Ver 1.4.17

July 25, 2014

Introduction

The data logging option for the SkyTraq GPS receiver firmware allows storage of position, velocity, and time information to external SPI serial flash memory. The criteria for storing the position velocity time information may be changed by the user.

SkyTraq implements proprietary data formats to store entries, which can help users to utilize the serial flash memory as efficiently as possible. The formats currently supported are full data format and compact data format.

A Windows-based utility is provided to allow easy configuration of the data logging function and retrieval of logged information. Through binary messages, **LOG Read Batch**, **LOG Configure**, **LOG Status**, and **LOG Clear** enable users to retrieve logged data, configure data logging criteria, retrieve log buffer status, and clear the flash memory log buffer. The **LOG Decompress** command on the utility is used to decompress proprietary binary data into readable tabulated output.

Data Logging Configuration Parameters

The data-logging algorithm allows user to set criteria for logging. Several parameters can be set through binary **LOG Configure** command. The LOG Configure command provides following configurable parameters:

- Time: thresholds of maximum time and minimum time in resolution of 1s.
- Distance: thresholds of maximum distance and minimum distance in resolution of 1m.
- Speed: thresholds of maximum speed and minimum speed in resolution of 1Km/hr.
- Enable: enable or disable Data Logging.

Data Logging Algorithm

Every second, PVT solution is generated by the GPS kernel. Logging to the serial flash memory is done according to the following rule:

T_{diff} = time of current fix – time of last stored position fix
$Diff\_D_x$ = absolute distance between current fix and last stored position fix in ECEF X axis
$Diff\_D_y$ = absolute distance between current fix and last stored position fix in ECEF Y axis
$Diff\_D_z$ = absolute distance between current fix and last stored position fix in ECEF Z axis
V = speed of current Fix
T_{th_max} = threshold of maximum time
T_{th_mim} = threshold of minimum time
D_{th_max} = threshold of maximum distance
D_{th_mim} = threshold of minimum distance
V_{th_max} = threshold of maximum speed
V_{th_mim} = threshold of minimum speed
$D_{diff} = sqrt(Diff\_D_x * Diff\_D_x + Diff\_D_y * Diff\_D_y + Diff\_D_z * Diff\_D_z)$
$T_{diff\text{-}min} = T_{diff} - T_{th\_mim}$
$D_{diff\text{-}min} = D_{diff} - D_{th\_mim}$
$T_{diff\text{-}max} = T_{diff} - T_{th\_max}$
$D_{diff\text{-}max} = D_{diff} - D_{th\_max}$

if $(((T_{diff\text{-}min} > 0) \&\& (D_{diff\text{-}min} >= 0) \&\& (V >= V_{th\_mim})) \| (T_{diff\text{-}max} > 0) \| (D_{diff\text{-}max} > 0) \| (V > V_{th\_max}))$
{
    if ((Boundary of FLASH Sector) || $(Diff\_D_x > 511)$ | $(Diff\_D_y > 511)$ | $(Diff\_D_z > 511)$ || $(T_{diff} > 65535)$)
        Store a full data entry to the flash.
    else
        Store a compact data entry to the flash.
}

Default Settings

Parameter	Default Vale	Maximum Vale	Minimum Vale
T_{th_max}	65535	2^32	0
T_{th_mim}	5	2^32	0
D_{th_max}	65535	2^32	0
D_{th_mim}	0	2^32	0
V_{th_max}	65535	2^32	0
V_{th_mim}	0	2^32	0
Enable	Enable		

Data Logging Binary Messages

Binary messages are used to communicate with GPS receiver. Please refer to “Application Note on Binary Messages” on basic binary message structures, commands and protocols. LOG Binary messages are used to communicate with data logging of GPS receiver. Below are detailed descriptions of the LOG binary messages.

LOG READ BATCH CONTROL – Enable data read from the log buffer (0x1D)

This is a request message which will control start reading log data from log buffer. This command is issued from the host to GPS receiver and GPS receiver should respond with an ACK or NACK. The payload length is 5 bytes.

Structure:

<0xA0,0xA1>< PL><1D>< message body><CS><0x0D,0x0A>

Example:

A0 A1 00 05 1D 00 00 00 02 1F 0D 0A

1 2 3 4 5

Field	Name	Example(hex)	Description	Type	Unit
1	Message ID	1D		UINT8	-
2-3	Start Sector	00000	Starting log sector	UINT16	
4-5	Number of Sectors	00002	How many sectors to read starting from Starting log sector	UINT16	
Payload Length : 5 bytes					

LOG STATUS CONTROL – Request Information of the Log Buffer Status (0x17)

This is a request message, which will request the GPS receiver to provide the log buffer status. This command is issued from the host to GPS receiver and GPS receiver will respond with an ACK or NACK. The payload length is 1 byte.

Structure:

<0xA0,0xA1>< PL><17>< message body><CS><0x0D,0x0A>

Example:

A0 A1 00 01 17 17 0D 0A

1

Field	Name	Example(hex)	Description	Type	Unit
1	Message ID	17		UINT8	
Payload Length: 1 bytes					

LOG CONFIGURE CONTROL – Configuration Data Logging Criteria (0x18)

This is a request message, which will request the GPS receiver to set the data logging criteria. This command is issued from the host to GPS receiver and GPS receiver should respond with an ACK or NACK. The payload length is 27 bytes.

Structure:

<0xA0,0xA1>< PL><18>< message body><CS><0x0D,0x0A>

Example:

A0 A1 00 1B 18 00 00 0E 10 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 02 0D 0A

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

Field	Name	Example(hex)	Description	Type	Unit
1	Message ID	18		UINT8	
2-5	max_time	00000E10	Default: 65535	UINT32	Second
6-9	min_time	00000005	Default: 5	UINT32	Second
10-13	max_distance	00000000	Default: 65535	UINT32	Meter
14-17	min_distance	00000000	Default: 0	UINT32	Meter
18-21	max_speed	00000000	Default: 65535	UINT32	Km/H
22-25	min_speed	00000000	Default: 0	UINT32	Km/H
26	datalog_enable	01	0: disable 1: enable	UINT8	
27	Reserved	00		UINT8	
Payload Length: 27 bytes					

LOG CLEAR CONTROL – Clear Data Logging Buffer (0x19)

This is a request message, which will request the GPS receiver to clear all the logged data. This command is issued from the host to GPS receiver and GPS receiver should respond with an ACK or NACK. The payload length is 1 byte.

Structure:

<0xA0,0xA1>< PL><19>< message body><CS><0x0D,0x0A>

Example:

A0 A1 00 01 19 19 0D 0A

1

Field	Name	Example(hex)	Description	Type	Unit
1	Message ID	19		UINT8	
Payload Length: 1 bytes					

LOG STATUS OUTPUT – Output Status of the Log Buffer (0x94)

This is a response message to ***LOG STATUS CONTROL, id 0x17***, which provides the log status of the GPS receiver. This message is sent from the GPS receiver to host. The payload length is 35 bytes.

Structure:

<0xA0,0xA1>< PL><94>< message body><CS><0x0D,0x0A>

Example:

A0 A1 00 23 94 AA EB 00 00 72 00 FF 00 10 0E 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

00 01 00 46 0D 0A

33 34 35

Field	Name	Example(hex)	Description	Type	Unit
1	Message ID	94		UINT8	
2-5	log_wr_ptr	AAEB0000	Current log buffer address. Sent as little endian bytes, refer to status field for how to construct the value at PC side.	UINT32	
6-9	status	7200FF00	Bytes: 8-9, total sectors Bytes: 6-7, sectors left Sent as little endian bytes, in the example here, the status at PC side is constructed as 00FF0072 Therefore, the total sectors are 0x00FF The sectors left are 0x0072	UINT32	
10-13	max_time	100E0000	Threshold of MAX time from last position fix Sent as little endian bytes, refer to status field for how to construct the value at PC side.	UINT32	Second
14-17	min_time	01000000	Threshold of MIN time from last position fix Sent as little endian bytes, refer to status field for how to construct the value at PC side.	UINT32	Second
18-21	max_distance	00000000	Threshold of MAX distance from last position fix Sent as little endian bytes, refer to status field for how to construct the	UINT32	Meter

			value at PC side.		
22-25	min_distance	00000000	Threshold of MIN distance from last position fix Sent as little endian bytes, refer to status field for how to construct the value at PC side.	UINT32	Meter
26-29	max_speed	00000000	Threshold of MAX speed from last position fix Sent as little endian bytes, refer to status field for how to construct the value at PC side.	UINT32	Km/H
30-33	min_speed	00000000	Threshold of MIN speed from last position fix Sent as little endian bytes, refer to status field for how to construct the value at PC side.	UINT32	Km/H
34	datalog_enable	01	0: data log disabled 1: data log enabled	UINT8	
35	log_fifo_mode	00	One way buffer	UINT8	
Payload Length: 35 bytes					

Remark:

Log status output may provide more information if GPS firmware supports datalog with POI function.

Evaluation Software Data Logging Interface

Data logging commands can be accessed through SkyTraq evaluation software under LOG submenu. Please refer to SkyTraq binary message document for message protocol. SkyTraq has provided the related APIs for reference. Summary of each command are:

- LOG Read Batch: Read logged entries from the flash log buffer. Binary data output to file with **.LOG** extension.
- LOG Configure: Set data logging criteria.
- LOG Clear: Clear all the entries in the flash log buffer.
- LOG Status: Retrieve status of the flash log buffer.
- LOG Decompress: Decompresses binary data to text data, output to a file with **.LOGG** extension*. A Google Earth KML file is also generated.

The code to decompress binary logged data into full PVT solution is provided in linkable library format. Please refer Appendix A for storage format.

Message Flow of LOG Read Batch

LOG Read Batch command (mentioned in section above) will launch a 3-step procedure: The first step is to get log status from receiver. The 2nd step is to set the communicate port to have higher baud rate. The third step is to get logged data from receiver. The detailed procedure is elaborated as below. Please refer to Figure 1 & also SkyTraq API source codes.

- PC sends a LOG STATUS binary message to retrieve information on how many sectors of data being logged by GPS receiver. The sectors left in the log status means not used up. For example, if the sectors left are 510, it means the sector number 510 is not used up and could be empty or partial used. Therefore, the user still need to do a log read on sector number 510.
- PC then change the baud rate of receiver to 115200 if it's not a bluetooth GPS receiver.
- PC will then send LOG READ BATCH, id 0x1D binary message to request GPS receiver to start output log data from the starting log sector to the ending log sector, i.e. the number of sectors beginning at the starting sector. Following the output log data, GPS receive will send END and CHECKSUM messages to PC. PC then verifies the received log data against the received checksum as an indication of a failure or success. Once it is a success, PC will continue the next starting sector read to next ending sector (start sector + number of sectors) until all the log data being read. On the other hand, if it is a failure, the same sectors will be retried again until success.

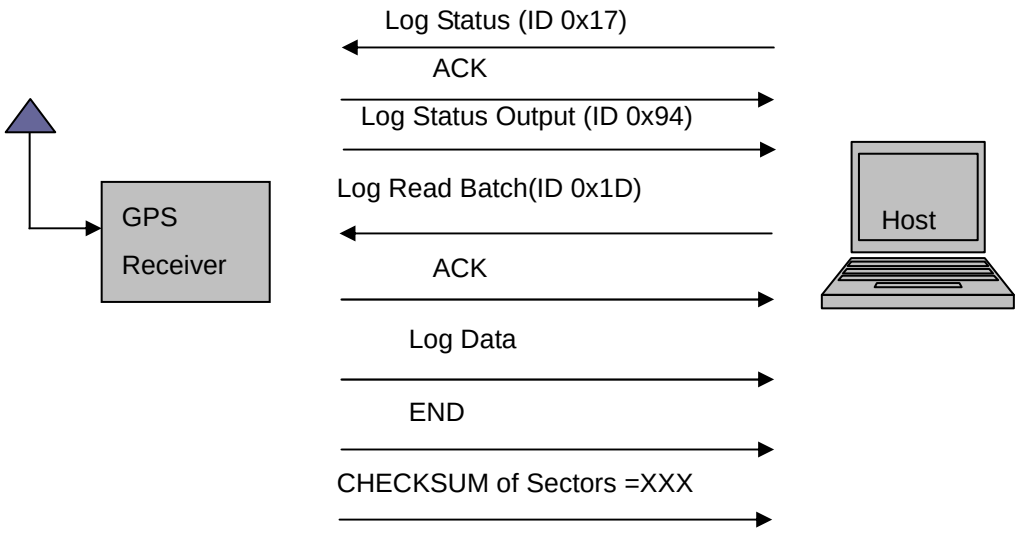
Log Status (ID 0x17)
ACK
Log Status Output (ID 0x94)
Log Read Batch(ID 0x1D)
ACK
Log Data
END
CHECKSUM of Sectors =XXX
GPS
Receiver
Host

Figure 1

Appendix A
Datalog Storage Format

The binary data of position fix information stored in memory can have different types and are automatically logged according the algorithm described in section, **Data Logging Algorithm**. Table 1 provides information of storage header types. Table 2 describes the variables that were saved in memory. Tables 3~6 list the detailed format of each data logging type. Note that U16 is unsigned short in C notation.

The datalog information currently were saved in SPI flash with uniform sector of 4096 bytes. Beginning of each sector of logged data, the first stored entry must be of type FIX FULL or FIX fULL POI. On the other hand, if the bytes left near the end of each sector are not enough for a data entry, data will be logged at the beginning of next sector.

3 Bits[15:13]	Type	Size In bytes	Description
111	EMPTY	2	Empty data
010	FIX FULL	18	Position fix information, full storage format
100	FIX_COMPACT	8	Position fix information, compact storage format
011	FIX FULL POI	18	Position fix information, full storage format, user POI

Table 1 Storage Header Types

Variables	Size	Unit	Description
V	10 Bit	Km/h	User velocity
WN	10 Bit	week	GPS Week Number
TOW	20 Bit	Sec.	GPS Time of Week
ΔTOW	16 Bit	Sec.	Difference between the current and last TOW
X/Y/Z in ECEF	32 Bit	Meters	XYZ position in ECEF Coordinate
ΔX/Y/Z in ECEF	10 Bit	Meters	Difference of last and current XYZ in ECEF Coordinate, Note: please refer to “Example of Log Data” below.

Table 2: Position Fix Information

Bit / Length	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3 Empty Storage

Bit / Length	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U16	0	1	0	Reserved				V[9:0]								
U16	TOW[3:0]				Reserved			WN[9:0]								
U16	TOW[19:4]															
U16	X[15:0]															
U16	X[31:16]															
U16	Y[15:0]															
U16	Y[31:16]															
U16	Z[15:0]															
U16	Z[31:16]															

Table 4: FIX_FULL

Bit / Length	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U16	1	0	0	Reserved				V[9:0]								
U16	ΔTOW [15:0]															
U16	ΔX[9:0]										ΔY[5:0]					
U16	ΔY[9:6]				Reserved		ΔZ[9:0]									

Table 5: FIX_COMPACT

Bit / length	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U16	0	1	1	Reserved				V[9:0]								
U16	TOW[3:0]				Reserved			WN[9:0]								
U16	TOW[19:4]															
U16	X[15:0]															
U16	X[31:16]															
U16	Y[15:0]															
U16	Y[31:16]															
U16	Z[15:0]															
U16	Z[31:16]															

Table 6: FIX_FULL_POI

Example of Log Data

The “log data” read from SPI flash in Figure 1 is shown below as an example. The log data is a continuous binary data. Users need to check the first byte to know the storage header type and in consequence its length is known.

40 6A 61 E7 61 8E 71 43 00 13 FA F0 FF BE 86 1B 00 45 => first byte 0x40 (FIX_FULL)
80 6A 00 01 01 D6 00 13 => first byte 0x80 (FIX_COMPACT)
80 6B 00 01 01 57 00 13 => first byte 0x80 (FIX_COMPACT)
80 6B 00 01 01 D6 00 12 => first byte 0x80 (FIX_COMPACT)
80 6B 00 01 01 96 00 13 => first byte 0x80 (FIX_COMPACT)

In the example, the first 2 bytes of FIX _FULL entry are 0x40 0x6A and according to Table 4, bit number 15, 14 and 13 are 010 which is identified as a FIX full entry. The same type identification mechanism applies to FIX_COMPACT. For example, 80 6A 00 01 01 D6 00 13, the first 2 bytes are 0x80, 0x6A and according to Table 5, bit number 15, 14 and 13 are 100 which is identified as FIX_COMPACT.

With the first entry of FIX_FULL type, users can calculate the PVT of the following FIX_COMPACT type. The values of ΔX, ΔY and ΔZ of FIX_COMPACT type is a difference between the current XYZ from the last stored XYZ in ECEF coordinate. If ΔX, ΔY and ΔZ are numbers bigger than 511 (512~1022), it means the difference is a negative number. And if ΔX, ΔY and ΔZ are numbers between 0~511, the difference is a positive number. Ex. if the ΔX is 512, the equations of $X_{FIX\_COMPACT}$ is shown below,

$X_{FIX\_COMPACT} = X_{FIX\_FULL} – (\Delta X_{FIX\_COMPACT} - 511)$

i.e.

$X_{FIX\_COMPACT} = X_{FIX\_FULL} – 1$

Pseudo codes of Log Data

The pseudo codes of FIX_FULL and FIX_COMPACT are as below

```
typedef signed      char            S08;
typedef unsigned char               U08;
typedef signed      short int       S16;
typedef unsigned short int          U16;
typedef signed      long int        S32;
typedef unsigned long int           U32;
typedef float                       F32;
typedef double                      D64;

typedef struct{
U16 word[9];
```

```
}FIX_FULL_DATA;

typedef struct{
U16 word[4];
}FIX_INC_DATA;

typedef struct{
U08   Type;
U32   WNO;
U32   TOW;
U32   DTOW;
S32   ECEF_X;
S32   ECEF_Y;
S32   ECEF_Z;
S16   DECEF_X;
S16   DECEF_Y;
S16   DECEF_Z;
U16   V;
}POS_FIX_REC;

void Set_Rec_Data(POS_FIX_REC* Current, POS_FIX_REC* Last)
{
   Current->TOW        = GPS tow;    //tow calculated by GPS kernel
   Current->WNO        = GPS wn;    //week number calculated by GPS kernel
   Current->V           =    GPS speed; //speed calculated by GPS kernel
   Current->ECEF_X     = GPS ECEF x; //position fix in ECEF X calculated by GPS kernel
   Current->ECEF_Y    = GPS ECEF y; // position fix in ECEF Y calculated by GPS kernel
   Current->ECEF_Z    = GPS ECEF z; // position fix in ECEF Z calculated by GPS kernel
   //caculate the difference between now and the last storage time
   Current->DTOW       = Current->TOW - Last->TOW;
   Current->DECEF_X = Current->ECEF_X - Last->ECEF_X;
   Current->DECEF_Y = Current->ECEF_Y - Last->ECEF_Y;
   Current->DECEF_Z = Current->ECEF_Z - Last->ECEF_Z;
}
void Full_Data(FIX_FULL_DATA* FixFull, POS_FIX_REC* Current)
{
   memset(FixFull,0,sizeof(FIX_FULL_DATA));
   FixFull->word[0]    = Current->V                &0x3ff;
   FixFull->word[0] |= (0x40<<8);
   FixFull->word[1]    = Current->TOW              &0xf;
```

```
    FixFull->word[1]    = FixFull->word[1]<<12;
    FixFull->word[1] |= Current->WNO              &0x3ff;
    FixFull->word[2]    = Current->TOW>>4         &0xffff;
    FixFull->word[3]    = Current->ECEF_X         &0xffff;
    FixFull->word[4]    = Current->ECEF_X>>16     &0xffff;
    FixFull->word[5]    = Current->ECEF_Y         &0xffff;
    FixFull->word[6]    = Current->ECEF_Y>>16     &0xffff;
    FixFull->word[7]    = Current->ECEF_Z         &0xffff;
    FixFull->word[8]    = Current->ECEF_Z>>16     &0xffff;
}
void Inc_Data(FIX_INC_DATA* FixInc, POS_FIX_REC* Current)
{
    if(Current->DECEF_X<0) Current->DECEF_X = 511+Current->DECEF_X*(-1);
    if(Current->DECEF_Y<0) Current->DECEF_Y = 511+Current->DECEF_Y*(-1);
    if(Current->DECEF_Z<0) Current->DECEF_Z = 511+Current->DECEF_Z*(-1);
    memset(FixInc,0,sizeof(FIX_INC_DATA));
    FixInc->word[0]    = Current->V                  &0x3ff;
    FixInc->word[0] |= (1<<15);
    FixInc->word[1]    = Current->DTOW               &0xffff;
    FixInc->word[2]    = Current->DECEF_X            &0x3ff;
    FixInc->word[2]    = FixInc->word[2]    << 6;
    FixInc->word[2] |= Current->DECEF_Y           &0x3f;
    FixInc->word[3]    = Current->DECEF_Y >> 6    &0xf;
    FixInc->word[3]    = FixInc->word[3]    << 12;
    FixInc->word[3] |= Current->DECEF_Z           &0x3ff;
}
void BackupLastData(POS_FIX_REC* Current, POS_FIX_REC* Last)
{
    Last->TOW       = Current->TOW;
    Last->ECEF_X = Current->ECEF_X;
    Last->ECEF_Y = Current->ECEF_Y;
    Last->ECEF_Z = Current->ECEF_Z;
}
main()
{
    POS_FIX_REC          Current, Last;
    FIX_FULL_DATA        Fix_Full;
    FIX_INC_DATA         Fix_Inc;
    Set_Rec_Data(&Current, &Last);
    Full_Data(&Fix_Full, &Current,);
```

```
    Inc_Data(&Fix_Inc    , &Current);
    BackupLastData(&Current, &Last);
}
```

Note: The SkyTraq GPS receiver will send the FIX_FULL in big-endian order. The data structure FIX_FULL_DATA, word[0] will be sent first and word[8] last. Bit 15 in word[0] will be sent first.

Change Log

Ver 1.4.17 July 25, 2014

1. Update default values of Tmax, Dmax and Vmax in Data logging section and Log configure control binary message.

Ver 1.4.16 Sep 24, 2010

1. Update log status output message for field, status, max_time, min_time, max_distance, min_distance, max_speed, min_speed & log_wr_ptr.

Ver 1.4.15 Sep 23, 2010

1. Update Log Configure Control binary command default values.

Ver 1.4.14 Sep. 22 , 2009
.
1. Update Appendix A "**Datalog Storage Format**" example of log data section.
2. Add Appendix A "**Datalog Storage Format**" Pseudo codes of log data section.

Ver 1.4.13 Aug. 12 , 2009

1. Update Section "**Message Flow of Log Read Batc**h".
2. Update Appendix A "**Datalog Storage Format**".

Ver 1.4.12 June 24, 2009

1. Update Log Read Batch Control (0x1D) command

Ver 1.4.11 Jan. 05, 2009

1. Update **Evaluation Software Data Logging Interface** section.
2. Update Figure 1.
3. Add an example in Appendix A.

Ver 1.4.10 Dec 11, 2008

1. Modify datalog default settings in **Data Logging Algorithm** section.

Ver 1.4.9 Nov 28, 2008

1. Add Appendix A for data log storage format.

Ver 1.4.8 Oct 28, 2008

1. Update log sector read command to log read batch command.
2. Modify :**Message Flow of LOG Read Batch"** section.

Ver 0.4.8, Aug. 28, 2007

1. Update Log Sector Read binary message.
2. Add message flow of log read

Ver 0.4.7, June 28, 2007

1. Update LOG Sector Read Message ID
2. LOG data output is raw data by each sector request

Ver 0.4.3, May 08, 2007

1. Modify Log read message
2. Modify Log status message

Ver 0.3.9, April 4, 2007

1. Add binary messages

Ver 0.3.8, March 12, 2007

1. Initial release

SkyTraq Technology, Inc.
4F, No.26, Minsiang Street, Hsinchu, Taiwan, 300
Phone: +886 3 5678650
Fax: +886 3 5678680
Email: info@skytraq.com.tw