ITRAX02 EVALUATION KIT

iTALK protocol specification

Rev 1.5

14.3.2002 13:46
FastraX Oy

CONTENTS

1. REFERENCES

Ref. #	Date	Document
01		iTrax02 Evaluation Kit: Software Installation Manual
02		iTrax02 Evaluation Kit: Software Architecture Overview
03		iTrax02 Evaluation Kit: GPS Workbench Users Guide
04		iTrax02 Evaluation Kit: iTalk Protocol Specification
05		iTrax02 Evaluation Kit: Data Formats Description
06		iTrax02 Evaluation Kit: NMEA Protocol

2. INTRODUCTION TO ITALK PROTOCOL

iTALK communication protocol enables iTRAX GPS receiver to communicate with an external host such as PC, notepad computer or mobile phone. It is a low-level communication layer that allows host to control the receiver and retrieve data from it.

For mapping services iTRAX also supports NMEA-0183 format data.

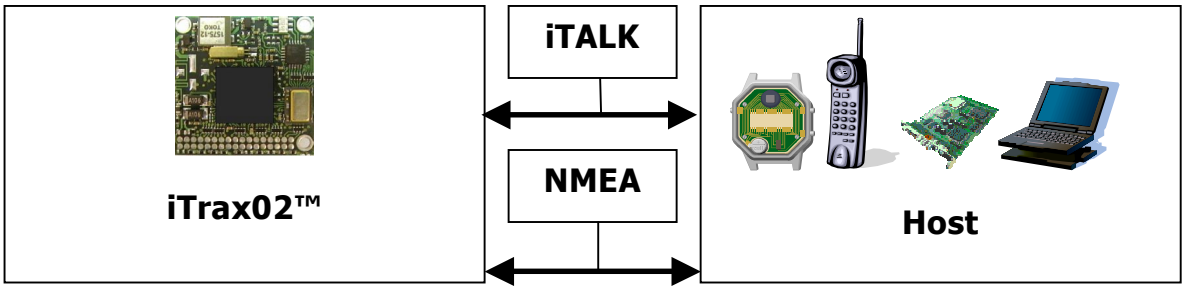


Figure 1 Support of iTalk and NMEA communication protocols.

2.1 Hexadecimal numbers

C type hexadecimal notation is used in this document. Actual hexadecimal value has prefix of "0x". Numerical values without this prefix are decimals unless stated otherwise.

Example: 0x7ab2 is hexadecimal number 7AB2 which equals decimal 31410.

2.2 Data types

Data type names used in this document refer to VS_DSP data types.

Integers:

- BYTE is 8bit unsigned integer
- WORD is 16bit unsigned integer
- DWORD is 32bit unsigned integer
- INT16 is 16bit signed integer
- INT32 is 32bit signed integer
- SHORT is same as INT16
- LONG is same as INT32

Float:

- DOUBLE is 48bit VS_DSP custom floating point value

2.3 Float value conversions

As noted above iTrax uses custom 3 word long floating point numbers. Following examples show how to convert these values from and to standard 4 word long IEEE-double values.

Experienced programmers will probably find the C-code example more familiar.

Conversion 64bit IEEE double to 48bit VS_DSP float

Pseudocode example:

```
DOUBLE mantissa = IEEE_double
DOUBLE ex
LONG exponent = 31

if mantissa = 0 then
      exponent = move_bits_of_( -1 )left( length_of_word-1 )
else if mantissa > min_double_value and mantissa < max_double_value then
      ex = log( abs( mantissa )) / log( 2.0 )
      exponent = ceiling_of( ex )
      mantissa = mantissa * ( 2^(31-exponent) )
else
      Error("Floating point overflow")
end if

mantissa_pointer = address_of( mantissa )
exponent_pointer = address_of( exponent )

if ( mantissa < 0 and IEEE_double > 0 ) or ( mantissa >= 0 and IEEE_double
< 0 ) then
      value_of_address( mantissa_pointer ) = move_bits_of_(
value_of_address( mantissa_pointer ) )right( 1 )
      value_of_address( exponent_pointer ) = value_of_address(
exponent_pointer ) + 1

      if IEEE_double >= 0 then
            value_of_address( mantissa_pointer ) = bitwise_and(
value_of_address( mantissa ),  not( move_bits_of( 1 )left( ( 2 *
length_of_word ) -1 ) ) )
      end if
end if

Rem VS_DSP float is now created as mantissa and exponent. Values should be
sent to iTrax as follows:

ioSend( lo_word( mantissa ) )
ioSend( hi_word( mantissa ) )
ioSend( lo_word( exponent ) )
```

C-Code example:

```
static void DoubleToVsFloat(double val, LONG *mantissa, LONG *exponent)
//
// Convert IEEE double to 3 WORD DOUBLE (VSDSP)
//
{
    double a = val, ex;
```

```
    LONG ee = 31;

    if (a == 0.0)
    {
        a = 0;
        ee = (DWORD)-1<<(sizeof(WORD)-1);
    }
    else
    {
        ex = log((a<0)?-a:a)/log(2.0);
        ee = ceil(ex);
        a *= pow(2,31-ee);
    }

    *mantissa = (DWORD)a;
    *exponent = ee;
    if ((*mantissa < 0 && val > 0) || (*mantissa >= 0 && val < 0))
    {
        *mantissa = (DWORD)(*mantissa) >> 1;
        *exponent += 1;

        if (val >= 0)
        {
            *mantissa = ((DWORD)(*mantissa)) && ~(1<<(2*sizeof(WORD)-1));
        }
    }
}
```

Conversion of 48bit VS_DSP float to 64bit IEEE double

Pseudocode example:

```
float_pointer = address_of( VS_DSP_float )

LONG mantissa = combine_to_make_long( float_pointer ) and ( float_pointer +
1)
SHORT exponent = value_of( float_pointer + 3)

DOUBLE IEEE_double = 0

if mantissa <> 0 then
      IEEE_double = calculate_exponent(mantissa,exponent-31)
end if
```

C-Code example:

```
double VsFloatToDouble(WORD* pW)
{
      LONG  mantissa = MAKELONG(*(pW), *(pW+1));
      SHORT exponent = short(*(pW+2));
      return mantissa ? ldexp(mantissa,exponent-31) : 0;
}

The above expects MAKELONG to be:
#define MAKELONG(a, b) \
    ((LONG) (((WORD) (a)) | ((DWORD) ((WORD) (b))) << 16))
```

2.4 Byte order

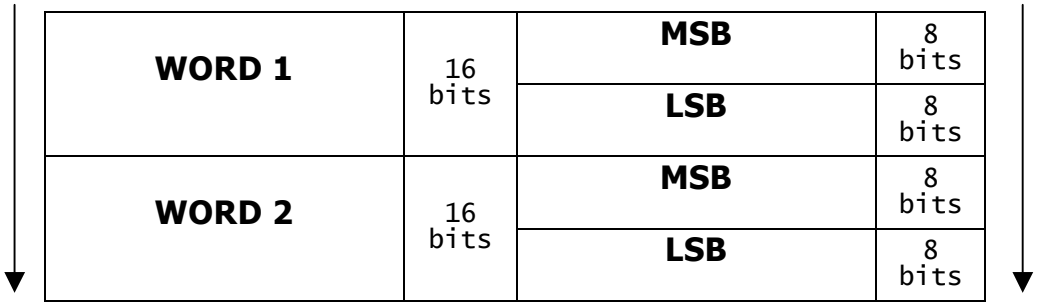


Figure 2 Order of bytes

iTalk handles data as words and follows VS_DSP byte order. This order is opposite to one used for example in Win32 environment (because of Intel CPUs) but is similar to one used in Motorolan CPU.

The 1st byte in each word is the Most Significant Byte (MSB) and the 2nd byte is the Least Significant Byte (LSB) in that word.

Example: To send WORD 0x1234 (decimal 4660) to iTrax one must be sure that most significant byte 0x12 is sent first and byte 0x34 after that. Following C function does just that:

```
void SendiTalk(void* p, WORD length)
{
      // p is the pointer to the iTalk message in this example
      // length is the length of the message in WORDS
      int n=0;
      WORD w;
      while(++n<length)
      {
            w=(WORD) *p;
            SendByte((w>>8)& 0x00ff);
            w=*p;
            SendByte(w & 0x00ff);
      };
}
```

Please note that this only applies to order of bytes not to order of words. Double word still consists of Least Significant Word (LSW) followed by Most Significant Word (MSW).

double word 1 0x12345678				double word 2 0xfedcba98			
LSW 0x5678		MSW 0x1234		LSW 0xba98		MSW 0xfedc	
MSB 0x56	LSB 0x78	MSB 0x12	LSB 0x34	MSB 0xba	LSB 0x98	MSB 0xfe	LSB 0xdc

As seen in previous picture these two doublewords are sent as bytes { 0x56, 0x78, 0x12, 0x34, 0xba, 0x98, 0xfe, 0xdc } whereas system such as Win32 would store them as bytes { 0x78, 0x56, 0x34, 0x12, 0x98, 0xba, 0xdc, 0xfe }.

3. TRANSFER MESSAGE STRUCTURE

iTALK transfer message encapsulates and carries the actual iTALK message trough i/o. It is formed of 6 parts:

1. Message start synchronization byte 1 (‘<’ hex 0x3C)
2. Message start synchronization byte 2 (‘*’ hex 0x2A)
3. Payload length (in words)
4. Payload (the iTALK message)
5. Checksum
6. Message end synchronization byte 3(‘>’ hex 0x3E)

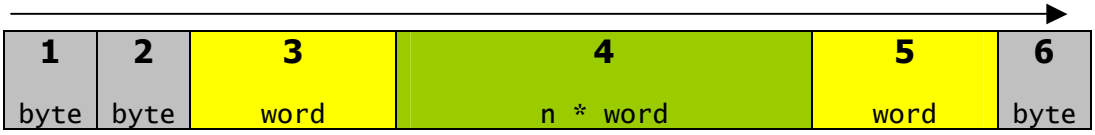


Figure 3 Parts of message

Note! that the byte order rules as introduced in subchapter 2.3 apply to parts 3, 4 and 5.

Chapter 7 introduces example messages.

3.1 Synchronization byte 1

Synchronization bytes precede and follow other parts of the message. Each iTALK message must start with synchronization bytes 1 and 2 and it must end with synchronization byte 3.

Hex	Dec	Char
3C	60	<

Figure 4 Synchronization byte 1 value

3.2 Synchronization byte 2

Hex	Dec	Char
2A	42	*

Figure 5 Synchronization byte 2 value

3.3 Payload Length

Second word of message must tell the length of the payload part of the message (which is the length of the iTALK message). Length must be measured in words NOT in bytes.

3.4 Payload

Payload is the actual iTalk message structure which is separate entity from iTalk transfer message. It consists of iTALK header (encapsulating an iSys header) and actual data.

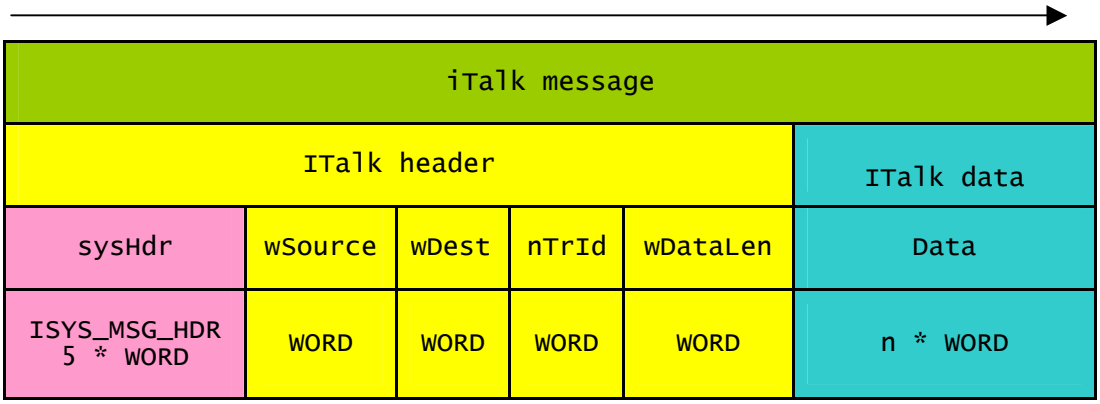


Figure 6 The iTalk message.

Header consists of five parts which are described below.

3.4.1 sysHdr

sysHdr is an iSys message header encapsulated inside iTalk header. It can be handled as 5 WORDs long part of iTalk header.

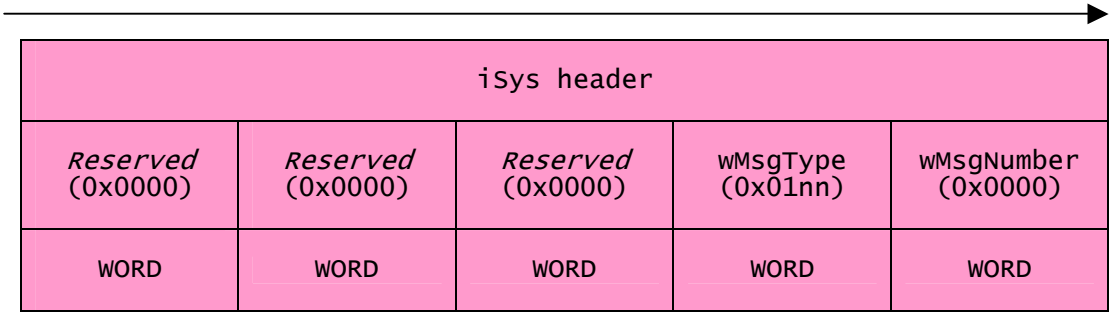


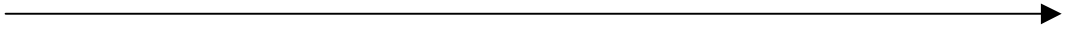
Figure 7 iSys header

Reserved words must be set 0x0000 (decimal 0) when host sends an iTalk message to iTrax. Messages sent by iTrax may have arbitrary values in these fields.

wMsgType is the only WORD in iSys header which has any importance to iTalk user. Most significant byte (MSB) of wMsgType must always be set to 0x01 (decimal 1) in iTalk messages. Least significant byte (LSB) identifies the message id. Message ids are listed along with appropriate message structures later in this document.

wMsgNumber must always be set to 0x0000 (decimal 0). Messages sent by iTrax may have arbitrary wMsgNumber values.

Example: Host sends NAV_START_MSG to iTrax. This message starts navigation in iTrax. Only significant WORD in iSys header is the wMsgType and the message id of NAV_START_MSG is 0x33 (decimal 51). Most significant byte of wMsgType is always 0x01 for iTalk messages and least significant is 0x33 for NAV_START_MSG. wMsgType will thus be 0x0133 (decimal 307).



iSys header				
Reserved	*Reserved*	*Reserved*	wMsgType	wMsgNumber
0x0000	0x0000	0x0000	0x0133	0x0000

Figure 8 Example iSys header of NAV_START_MSG.

3.4.2 wSource

Source of message is identified with a WORD with least significant byte marking the task id and most significant byte marking the node id.

Node id must have one of the following values:

Node	Value (hex)	Value (dec)	Meaning
Host node	0x02	2	The host.
iTrax node	0x01	1	The iTrax.
same node	0x00	0	Message is coming from same node that it is going to. Target is defined in wDest.

Task id defines the task that sent the message. Host should identify itself with task id 0 (main task) or alternatively with task id higher than 11.

Task id	Value (hex)	Value (dec)	Meaning
Main	0x00	0	The main task. Host should usually identify itself with this id when sending messages to iTrax.
Reserved	0x01	1	Must not be used.
Reserved	0x02	2	Must not be used.

Task id	Value (hex)	Value (dec)	Meaning
Navigation	0x03	3	Navigation task in iTrax.
Message	0x04	4	Message task in iTrax.
CSP	0x05	5	CSP (Custom Serial Protocol) input task in iTrax.
CONTROL	0x06	6	Control task in iTrax.
CSP out	0x07	7	CSP (Custom Serial Protocol) output task in iTrax.
User 1	0x08	8	Task id reserved for custom iTrax task developed using iSuite SDK.
User 2	0x09	9	Task id reserved for custom iTrax task developed using iSuite SDK.
Reserved	0x0a	10	Must not be used.
Reserved	0x0b	11	Must not be used.
Archive	0x0c	12	Archive task in host. Most iTrax messages are directed to this task.
Monitor	0x0d	13	Monitor task in host. Some iTrax messages are directed to this task.
other	0x0e – 0x14	14 – 20	Any other task in host. iTrax never sends anything to these tasks.

Example: Host receives message with source 0x0103 (decimal 259). Most significant byte is 0x01 (decimal 1) which stands for iTrax node and least significant byte is 0x03 (decimal 3) which implies navigation task. Message came from navigation task in iTrax.

Example: Host sends message to iTrax. Host should identify itself as main task in host node. Most significant byte is set to 0x02 (decimal 2) for host node and least significant byte is set to 0x00 (decimal 0) for main task. wSource will therefore be 0x0200 (decimal 512).

Note that host may as well ignore the wSource field of iTrax-sent messages and handle all of them. Addressing is provided to make life easier for programmers developing iTalk protocol for multitasking hosts but host doesn't necessary need to read the whole field.

3.4.3 wDest

Destination of message is identified with a WORD exactly the same way as wSource.

Unless iTrax runs custom software developed with iSuite SDK it will send all trace messages to host's monitor task and all other messages to host's archive task. (Trace messages explained later in this document.) In this case host should send all it's messages to the main task of iTrax.

Example: Host sends a message to iTrax. iTrax runs normal software and therefore host needs to direct the message to main task of iTrax. Most significant byte is set to 0x01 (decimal 1) to imply iTrax node and least significant byte to 0x00 (decimal 0) which stands for main task. wDest will be 0x0100 (256). This is the wDest one should use when sending messages to iTrax module. (That is unless iTrax runs custom software.)

3.4.4 nTrId

Transaction id. Value is a INT16 and it is greater than 0 for transaction message, less than 0 for reply message and 0 for other messages.

If there is no need to verify that message was successfully transmitted to iTrax host should use value 0.

If host needs to verify that iTrax received the message it should send the message as a transact message. This is accomplished by setting the nTrId of the message to a positive value. iTrax responds to such a message with reply message. Reply message's nTrId's equals zero minus original transact message's nTrId.

Example: Host sends a message to iTrax with nTrId = 5. Next iTrax replies with a message with nTrId = –5.

Only one transact message at a time is allowed: Host must not send a transact message and then send another transact message before waiting for an answer for the first message.

nTrId is a running number starting from 1. Host must not send transact messages with identical nTrIds. nTrId must be increased by 1 every time a transact message is sent.

Example: Host sends another message to iTrax. This time nTrId will be 6 and iTrax will reply with message that has nTrId of –6. Next host sent message will have nTrId = 7 and iTrax will reply with nTrId = -7. After this host sends unsyncronized message with nTrId=0. iTrax won't reply and nTrId counter will not be increased. Next synchronized message's nTrId will be 8 and so on…

nTrId must never be greater than 32760. When host reaches 32761 it should reset it's counter to 1 and send the message with transact id 1. This is the only time that host is allowed not to follow the sequential numbering of transmission ids.

Example: Host sends message with nTrId = 32760, iTrax replies with nTrId = -32760. Next transact message that host sends will have nTrId = 1.

iTrax never sends transaction messages to host thus host never needs to answer to such a message.

Only message type listed in this document that can be sent as transact message is NAV_STATE_MSG. Programmers using iSuite to develop custom software may create tasks that expect transact messages.

3.4.5 wDataLen

wDataLen is a WORD implying the length of data measured in WORDs.

Example: Length of _NAV_STOP structure is 3 WORDs therefore length of data in NAV_STOP_MSG is 3.

3.4.6 Data

Data part of the message is the structure of appropriate message as described later in this document.

Example: STOP_NAV_MSG is an iTalk message from host node's main task to iTrax's main task with nTrId=0 and data part consisting of _STOP_NAV structure. wDataLen would be the size of _STOP_NAV in WORDs.

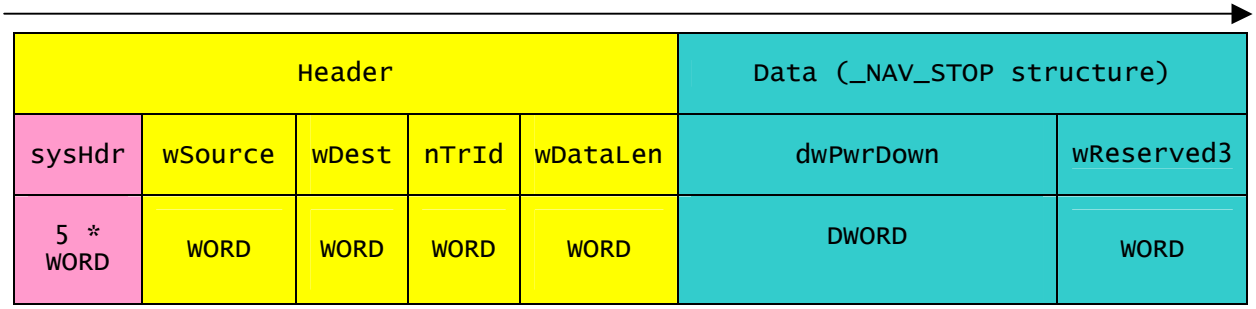


Figure 9 NAV_STOP_MSG

3.5 Checksum

Checksum should be checked for every iTALK message to assure that message transfer was error free.

Checksum is formed from the data part of the iTALK message (iTALK message is the payload) by calculating the sum of every word in it. Checksum itself is a word and thus rotates over maximum value during calculation.

Note! Do not calculate checksum from the whole payload. Use just the data part of the included iTALK message.

3.5.1 Checksum algorithm

Following pseudocode calculates the checksum.

```
INTEGER n = start_of_data
WORD CheckSum=0
while n < length_in_words(data)
   CheckSum = CheckSum + data[n]
end while
```

Note that it is assumed in this example that data index starts from 0 as in C. If that is not the case you will need to replace the '<' operator with '<='. Checksum is 16-bit unsigned integer, that is expected to rotate back to zero when an overflow occurs.

```
C-Code example:
WORD  RS_CalcChecksum(ITALK_MSG * pMsg)
{
register int i;
register WORD wChkSum = 0;
int nLen = pMsg->Hdr.wDataLen;
   // Exclude system header !
   WORD* pW = (WORD*)(&pMsg->Data[0]);
   // Calculate checksum
   for (i = 0; i < nLen; i++) {
         wChkSum += *pW++;
   }
   return wChkSum;
}
```

3.6 Synchronization byte 3

Hex	Dec	Char
3E	62	>

Figure 10 Synchronization byte 3 value

4. OUTPUT MESSAGES

This chapter lists messages that iTrax02 can send to host. These are the messages that may be filtered out by using iTalk message mask.

message id	message
1	PSEUDO_DATA_MSG
2	*reserved*
3	*reserved*
4	UTC_IONO_MSG
5	TRACK_MSG
6	ACQ_MSG
7	NAVIGATION_MSG
8	NAV_KALMAN_MSG
9	PRN_STATUS_MSG
10	CUSTOM_FIX_MSG
11	*AIDING_MSG* **
12	SUBFRAME_MSG
13	*AGC_CONTROL_MSG* *
14	*ACQ_AIDING_MSG* **
15	EPHEMERIS_MSG

Figure 11 Output messages

** AGC_CONTROL_MSG is the only message that can be sent by iTrax as well as by host. It is listed in chapter "Input/Output Messages".*

*** Host initially sends AGC_AIDING_MSG and AIDING_MSG to iTrax but iTrax responds to these messages by routing them to host's archive task. These tasks are listed in Input Messages chapter.*

Note! NAV_STATE_MSG is not listed here because it is pure response to host's request and cannot be masked out. Please refer to Input Messages chapter.

4.1 Pseudo data messages

These messages contain the raw measurement data from tracking channels along with rough receiver time stamps and precalculated pseudoranges and pseudorange rates.

4.1.1 PSEUDO_DATA_MSG

Message id 1. Contains raw pseudo data processed by the MSG task.

Message structure:

type	Parameter	Description
DWORD	DwMs	FTF instant (counter value in milliseconds) of observation
INT16	iGPSweek	Receiver assumed GPS week
LONG	lGPStow	Receiver assumed GPS TOW in milliseconds
INT16	inumObs	Number of valid observations to follow
PSEUDO_OBS	sv[RCVR_CHANNELS]	Channel states,see below. RCVR_CHANNELS = 12

Figure 12_PSEUDO_DATA structure

type	parameter	description
WORD	wPrn	Satellite's PRN code
WORD	wSnr	S/N measured in dBHz. Range from 0 to 63.
WORD	wLock	Lock status bitmap. bit set if 0 Carrier lock 1 Code lock 2 Channel is initialized 3 First track packet from this channel. Reserved for internal use 4 Bit sync 5 Subframe sync 6 First interrupt for the channel. Reserved for internal use 7 Reserved 8 Bit sync done for this bit value. Reserved for internal use 9 Bit time is valid 10 Reference PRN cycle counter is valid and thus reference time is valid 11 Channel is in code sweep mode 12 Reserved for future use 13 Reserved for future use 14 Ambiguous measurement 15 Last packet of a group
LONG	lTXtowMs	TOW of the signal transmit in full milliseconds
DOUBLE	dTXtowFrac	Fractional part of TOW of the signal transmit in seconds (by default equals zero) Exact GPS TOW can be calculated from lTXtowMs and dTXtowFrac with following algorithm: GPSTOW = (lTXtowMs/1000.0) + dTXtowFrac
DOUBLE	dDoppler	Doppler in m/s
DOUBLE	dPseudoRange	Code pseudorange in meters.

Figure 13_PSEUDO_OBS structure

4.1.2 SUBFRAME_MSG

Message id 12.

Contains one subframe of decoded navigation data stream.

Message structure:

type	parameter	Description
DWORD	dwMs	Receiver's millisecond ticks. This is arbitrary counter that rolls over approximately once a week.
WORD	wChan	Channel number
WORD	wPrn	Satellite's PRN code
WORD	wSid	Subframe ID number
DWORD	dwTow	TOW number
DWORD	dwData[10]	The 10 (30-bit) data words in the subframe

Figure 14_SUBFRAME structure

4.2 Navigation messages

Navigation task sends navigation messages to archive task of the host. These are the actual messages containing the position information. There are two types of navigation messages available: standard (LSE) and Kalman.

4.2.1 NAVIGATION_MSG

Message id 7.

Contains time, position, velocity and other basic navigational data computed by the least squares algorithm.

Message structure:

type	Parameter	Description
INT16	iFOM	Figure Of Merit in meters. Meaningful for overdetermined solutions (residual) In optimal situation value is 0, otherwise higher. Zero also for evenly determined solution.
DWORD	dwMs	Instant of observation as receiver's millisecond ticks. This is arbitrary counter that rolls over approximately once a week.
INT16	iWeek	GPS week of fix
LONG	lTow	Time Of Week (TOW) of the fix in full milliseconds
DOUBLE	dTow	Fractional part of TOW of the fix in seconds. Exact GPS TOW can be calculated from lTow and dTow with following algorithm: GPSTOW = (lTow/1000.0) + dTow
DOUBLE	dX	WGS84 X coordinate in meters
DOUBLE	dY	WGS84 Y coordinate in meters
DOUBLE	dZ	WGS84 Z coordinate in meters
DOUBLE	dLat	WGS84 latitude in radians
DOUBLE	dLon	WGS84 longitude in radians
DOUBLE	dAlt	Height from the surface of WGS84 reference ellipsoid in meters
DOUBLE	dVx	Velocity's WGS84 X directional component in m/s.
DOUBLE	dVy	Velocity's WGS84 Y directional component in m/s.

type	Parameter	Description
DOUBLE	dVz	Velocity's WGS84 Z directional component in m/s.
DOUBLE	dClockOffset	Clock offset in meters
DOUBLE	dClockDrift	Clock drift in m/s.
DOUBLE	dGdop	GDOP (Geometric Dilution Of Precision)
DOUBLE	dPdop	PDOP (Position DOP)
DOUBLE	dVdop	VDOP (Vertical DOP)
DOUBLE	dHdop	HDOP (Horizontal DOP)
DOUBLE	dTdop	TDOP (Time DOP)
INT16	iAltAided	Was this position calculated using altitude aiding? Possible values: 0 – no altitude aiding used 1 – altitude aiding was used
DOUBLE	dAidAltitude	If altitude aiding was used this is the altitude used. Value in meters.
DWORD	dwPrnMap	PRN bitmap that defines which satellites were used in navigation. Bit 0 defines PRN 1 and so on. I.e. if satellites PRN 3,12,20 and 21 were used dwPrnMap's value would be: 00000000 00011000 00001000 00000100
INT16	iSvCountUsed	Number of satellites used in fix. Note that altitude aiding counts as one extra satellite.
WORD	wDiffCorr	DGPS is currently not supported. This variable will be set to 0.

Figure 15_NAVIGATION stucture

4.2.2 NAV_KALMAN_MSG

Message id 8.

Contains time, position, velocity and other basic navigational data computed by the Kalman algorithm.

Note! Kalman navigation is currently not supported!

Message structure is identical to that of *NAVIGATION_MSG*.

4.2.3 CUSTOM_FIX_MSG

Message id 10.

The localized position information.

Message structure:

type	parameter	description
DATE_TIME	date	UTC time and date
DWORD	dwMs	Instant of observation as receiver's millisecond ticks. This is arbitrary counter that rolls over approximately once a week.
INT16	iWeek	GPS week of fix
LONG	lTow	TOW of the fix in full milliseconds
DOUBLE	dTow	Fractional part of TOW of the fix in seconds. Exact GPS TOW can be calculated from lTow and dTow with following algorithm: GPSTOW = (lTow/1000.0) + dTow
INT16	iTimeFOM	Figure Of Merit for time

type	parameter	description
INT16	iFixFOM	Figure Of Merit for position in meters
DOUBLE	dLat	WGS84 latitude in radians
DOUBLE	dLon	WGS84 longitude in radians
DOUBLE	dAlt	Height from surface of WGS84 reference ellipsoid in meters
DOUBLE	dUndulation	Undulation (geoid height) in meters
DOUBLE	dVn	Velocity's north component in m/s.
DOUBLE	dVe	Velocity's east component in m/s.
DOUBLE	dVu	Velocity's up component in m/s.
DOUBLE	dSpeed	Horizontal speed in m/s
DOUBLE	dDir	True heading in degrees
DOUBLE	dClockOffset	Clock offset in meters
DOUBLE	dClockDrift	Clock drift in m/s.
DOUBLE	dHdop	HDOP (Horizontal DOP)
INT16	iAltAided	Was this position calculated using altitude aiding? Possible values: 0 – no altitude aiding used 1 – altitude aiding was used
DOUBLE	dAidAltitude	If altitude aiding was used this is the altitude used. Value in meters.
INT16	iMagDecl	Magnetic declination in 1/10 degrees
INT16	iSvCountUsed	Number of satellites used in fix. Note that altitude aiding counts as one extra satellite.
WORD	wDiffCorr	DGPS is currently not supported. This variable will be set to 0.

Figure 16_CUSTOM_FIX structure

4.3 Satellite and orbit messages

These messages carry satellite tracking related data.

4.3.1 EPHEMERIS_MSG

Message id 15.

Usually contains the precise orbit elements and clock corrections of one satellite but may also carry the almanac data of one satellite.

Message structure:

type	parameter	Description
WORD	wPrn	Satellite's PRN code
WORD	wHealth	This field contains two pieces of information: Lowest 12 bits: Is satellite not healthy? Possible values: 0 – Satellite is healthy. not 0 – Satellite is not healthy. Highest 4 bits: Is this structure used to pass the ephemeris or almanac of this satellite? Possible Values: 0 - This structure contains ephemeris data. not 0 – This structure contains almanac data.

type	parameter	Description
WORD	wIODC	Issue Of Data Clock
DOUBLE	dGroupDelay	Group delay in seconds
WORD	wTocWeek	GPS week of TOC
DOUBLE	dToc	Reference time for clock data in seconds
DOUBLE	dAf0	AF0 satellite clock correction in seconds
DOUBLE	dAf1	AF1 satellite clock correction in s/s
DOUBLE	dAf2	AF2 satellite clock correction in s/(s^2)
WORD	wIODE	Issue of Data Ephemeris
WORD	wToeWeek	GPS week of TOE
DOUBLE	dToe	Reference time for ephemeris data in seconds
DOUBLE	dDeltan	Mean motion difference in rad/s
DOUBLE	dM0	Mean anomaly at reference time TOE in radians
DOUBLE	dEcc	Eccentricity
DOUBLE	dSqrta	Square root of the semimajor axis in meters
DOUBLE	dA	Semimajor axis in meters
WORD	wFit	Curve fit interval
DOUBLE	dCrs	Sin harmonic correction to orbit radius in meters
DOUBLE	dCrc	Cos harmonic correction to orbit radius in meters
DOUBLE	dCus	Sin harmonic correction to argument of latitude in radians
DOUBLE	dCuc	Cos harmonic correction to argument of latitude in radians
DOUBLE	dCis	Sin harmonic correction to inclination in radians
DOUBLE	dCic	Cos harmonic correction to inclination in radians
DOUBLE	dOmega0	Right ascension at reference time TOE in radians
DOUBLE	dOmega	Argument of perigee in radians
DOUBLE	dOmegaDot	Rate of right ascension in rad/s
DOUBLE	dI0	Inclination at reference time TOE in radians
DOUBLE	dIdot	Rate of inclination in rad/s
DOUBLE	dN0	Mean motion in rad/s

Figure 17_EPHEMERIS structure

4.3.2 PRN_STATUS_MSG

Message id 9.

Contains the current position of the satellites.

Message structure:

type	parameter	description
DWORD	dwMs	
INT16	iGPSWeek	Receiver assumed GPS week
LONG	lGPSTow	Receiver assumed GPS TOW in milliseconds
DWORD	dwVisible	Bitmap that describes visibility of each satellite. Set bit is a visible satellite. Bit 0 refers to PRN 1, bit 1 to PRN 2 … and bit 31 to PRN 32. Example: bitmap: 00100000 00000000 00001111 01000010 means that satellites PRN 2,7,9,10,11,12 and 30 are visible.
WORD	wSnr[MAX_PRNS]	S/N measured in dBHz for each satellite (MAX_PRNS = 12)

INT16	iEl[MAX_PRNS]	Elevation of each satellite in degrees
INT16	iAz[MAX_PRNS]	Azimuth of each satellite in degrees

Figure 18_PRN_STATUS structure

Note! If ephemeris is available it will always be used rather than almanac.

4.3.3 TRACK_MSG

Message id 5.

iTrax sends track messages if at least one receiver's channel is frequency locked and code locked.

Message structure:

type	parameter	description
DWORD	dwMs	Receiver's millisecond ticks. This is arbitrary counter that rolls over approximately once a week.
INT16	iNumTrax	Number of valid channels.
TRACK_DATA	TrackData[RCVR_CHANNELS]	Track data for each of the 12 channels. Note that only *iNumTrax* of these are valid. See below.

Figure 19_TRACK structure

type	parameter	description
DWORD	dwMs	Receiver's millisecond ticks. This is arbitrary counter that rolls over approximately once a week.
WORD	wChan	Correlator channel
WORD	wPrn	Satellite's PRN code
WORD	wLock	Lock status bitmap. bit set if 0 Carrier lock 1 Code lock 2 Channel is initialized 3 First track packet from this channel. Reserved for internal use 4 Bit sync 5 Subframe sync 6 First interrupt for the channel. Reserved for internal use 7 Reserved 8 Bit sync done for this bit value. Reserved for internal use 9 Bit time is valid 10 Reference PRN cycle counter is valid and thus reference time is valid 11 Channel is in code sweep mode 12 Reserved for future use 13 Reserved for future use 14 Reserved for future use 15 Last packet of a group
WORD	wDet	Carrier lock detector value. Measured in custom units.
LONG	lPrnTow	Reference GPS time of transmission in milliseconds
WORD	wPrnCycles	PRN cycles since reference (1.023MHz)
WORD	wPrnChip	PRN chip (within the C/A code)
WORD	wPrnPhase	PRN phase (within chip)

DWORD	dwCarrCount	16 most significant bits of LO NCO cycle counter value
WORD	wCarrPhase	Hi byte: 8 LSBs of LO NCO counter value Lo byte: LO NCO phase

Figure 20_TRACK_DATA structure

4.3.4 ACQ_MSG

Message id 6.

This message is sent when new satellite is acquired.

Message structure:

type	parameter	Description
ACQ_DATA	AcqData	See below.
ACQ_DEBUG	AcqDebug	Reserved for future use.

Figure 21_ACQ structure

type	parameter	Description
DWORD	dwMs	Instant of acquire as receiver's millisecond ticks. This is arbitrary counter that rolls over approximately once a week.
WORD	wPrn	Satellites PRN code
WORD	wAmplitude	Signal amplitude measured in custom units
WORD	wCodePhase	Code phase (chip)
DWORD	dwCodeFreq	Code frequency measured in custom units
DWORD	dwCarrFreq	Carrier frequency measured in custom units

Figure 22_ACQ_DATA structure

ACQ_DEBUG structure is 28 WORDs of arbitrary data and should be ignored by the host.

4.4 Time conversion and atmospheric modeling messages

These messages carry information concerning ionospheric modeling and UTC conversion.

4.4.1 UTC_IONO_MSG

Message id 4.

Contains the conversion factors between GPS and UTC time as well as the coefficients for ionospheric modeling.

In message structure UTC parameters marked with yellow and ionosphere model parameters with turquoise:

type	parameter	description
LONG	LTow	GPS TOW If value is -1 then rest of this message is invalid.
INT16	iGPSweek	GPS week If value is -1 then we don't have reliable GPS week yet

type	parameter	description
DOUBLE	dA0	Constant term of GPS time / UTC difference in seconds.
DOUBLE	dA1	First order term for GPS time / UTC difference (s/s)
INT16	iDtls	Delta time due to leap seconds, in seconds.
INT32	lTot	Reference time for UTC data in seconds.
INT16	iWNt	UTC reference week number
INT16	iWNlsf	UTC week number for future leap seconds
INT16	iDN	Day number for future leap seconds
INT16	iDtlsf	Delta time due to future leap seconds
DOUBLE	dAlpha0	Coefficients for vertical delay calculation.
DOUBLE	dAlpha1	"
DOUBLE	dAlpha2	"
DOUBLE	dAlpha3	"
DOUBLE	dBeta0	Coefficients for model period calculation.
DOUBLE	dBeta1	"
DOUBLE	dBeta2	"
DOUBLE	dBeta3	"

Figure 23_UTC_IONO structure

5. INPUT MESSAGES

This chapter lists the messages that host can send to iTrax.

message id	message
11	*AIDING_MSG **
13	*AGC_CONTROL_MSG *
14	*ACQ_AIDING_MSG **
41	NAV_PARAMS_MSG
42	KALMAN_PARAMS_MSG
43	LSE_PARAMS_MSG
51	NAV_START_MSG
52	NAV_STOP_MSG
53	*NAV_STATE_MSG ***
54	NAV_START_RECONFIG_MSG
61	ALARM_MSG
100+	GPS_SIMULATION_MSG

Figure 24 Input messages

** AGC_CONTROL_MSG is the only message that can be sent by iTrax as well as by host. It is listed in chapter "Input/Output Messages".*

*** Host initially sends AIDING_MSG and ACQ_AIDING_MSG to iTrax but iTrax responds to these messages and sends them to host's archive task.*

**** Host sends NAV_STATE_MSG to iTrax which fills the message structure with appropriate data and sends it to host's archive task.*

5.1 Parameter messages

Host manages parameter settings at iTrax02 using set of parameter messages.

5.1.1 NAV_PARAMS_MSG

Message id 41.

Sets navigation parameters at iTrax.

Message structure:

type	parameter	description
INT16	iPermanent	Should navigation parameters be stored in iTrax flash when navigation is stopped? Possible values: 1 - store (semi)permanently to flash when NAV_STOP_MSG is called: Same navigation parameters will be used when navigation is restarted. 0 – do not store to flash when NAV_STOP_MSG is called: Same navigation parameters will not be available when navigation is restarted. -1 – restore navigation parameters to default values: rest of this message is ignored.
INT16	iNavMode	Navigation algorithm to use. Possible values: 1 – NAV_MODE_LSQ Use LSQ algorithm 2 – NAV_MODE_KALMAN Use Kalman algorithm (currently not supported)
INT16	iRawDataRate	Interval of TRACK data messages in milliseconds.
DWORD	dwNavFixRate	Interval of NAVIGATION data output in milliseconds.
INT16	iAddInitParams1	Should initialization parameter set 1 (next 7 parameters in this structure) be used? Possible values: 0 – ignore next 7 parameters 1 – use next 7 parameters The initialization parameter set 1 is marked here with yellow.
INT16	iUseTropo	Should tropospheric model be used to correct navigation data? Possible values: 0 – ignore tropospheric model 1 – use tropospheric model
INT16	iUseIono	Should ionospheric model be used to correct navigation data? Possible values: 0 – ignore ionospheric model 1 – use ionospheric model
INT16	iStatic	Use static test mode? Possible values: 0 – Receiver is not static or not known if mobile. This is the normal operation mode. 1 – Receiver is static and it's location is defined in initialization parameter set 2. Deviation of defined location to actual location must be smaller than 300m. This is a test mode that can be used to test various navigational corrections such as atmospheric models.

type	parameter	description
INT16	iUseAltAiding	Should altitude aiding be used? Possible values: 0 – NO_ALT_AIDING Do not use altitude aiding. 1 – ALT_EXTERNAL Use altitude data from external source. 2 – ALT_CONSTANT Use constant altitude value. Value defined in *iConstantAlt*.
INT16	iConstantAlt	If *iUseAltAiding* is set to ALT_CONSTANT this WGS altitude (in meters) will be used.
INT16	iDatumID	Local datum ID. Datum ids are listed in [06].
WORD	wDiffCorr	Should differential correction (DGPS) be used? Possible values: 0 – Do not use DGPS 1 – Use DGPS (currently not supported)
INT16	iAddInitParams2	Should initialization parameter set 2 (next 10 parameters in this structure) be used? Possible values: 0 – ignore next 10 parameters 1 – use next 10 parameters The initialization parameter set 2 is marked here with turquoise.
INT16	iGPSweek	Estimated GPS week Set to -1 if not known
LONG	lGPStowMs	Estimated TOW Set to -1 if not known
DWORD	dwRefMs	Receiver tick for GPS stamp in milliseconds. Set this to -1 and let receiver decide.
DOUBLE	dElMask	Elevation mask in degrees up from horizon. Possible values 0-90. Satellites with lower angle than this will not be used in navigation.
DOUBLE	dRefX	Reference GPS position's WGS84 X element (m). Reference location is only used when testing receiver in static mode (see *iStatic*).
DOUBLE	dRefY	Reference GPS position's Y element. See *dRefX*.
DOUBLE	dRefZ	Reference GPS position's Z element. See *dRefX*.
DOUBLE	dMaxAltitude	Maximum allowed altitude in meters. For U.S. trade regulations this must be set to 18288m (60000ft).
DOUBLE	dMaxVelocity	Maximum allowed velocity in m/s. For U.S. trade regulations this must be set to 514m/s (1000nmls/h).
DOUBLE	dMaxAcceleration	Maximum allowed acceleration in m/(s^2). This is extra limitation in addition to two above. Recommended value 8m/(s^2).

Figure 25_NAV_PARAMS structure

5.1.2 KALMAN_PARAMS_MSG

Message id 42.

Sets parameters for Kalman navigation at iTrax. Most cases there should be no need to alter the default values.

Note! Kalman navigation is currently not supported! Host must not send this message at the moment.

Message structure:

type	parameter	description
INT16	iPermanent	Should Kalman navigation parameters be stored in iTrax flash when navigation is stopped? Possible values: 1 - store (semi)permanently to flash when NAV_STOP_MSG is called: Same Kalman navigation parameters will be used when navigation is restarted. 0 – do not store to flash when NAV_STOP_MSG is called: Same Kalman navigation parameters will not be available when navigation is restarted. -1 – restore Kalman navigation parameters to default values: rest of this message is ignored.
DOUBLE	dSigmapos2	Position variance
DOUBLE	dSigmavelo2	Velocity variance
DOUBLE	dRaimon	Position RAIM coefficient
DOUBLE	dRaimonv	Velocity RAIM coefficient
DOUBLE	dSp1	Increase in uncertainty of position during temporal update
DOUBLE	dSp2	Increase in uncertainty of velocity during temporal update

Figure 26_KALMAN_PARAMS structure

5.1.3 LSE_PARAMS_MSG

Message id 43.

Sets parameters for LSE navigation at iTrax. Most cases there should be no need to alter the default values.

Message structure:

type	parameter	description
INT16	iPermanent	Should LSE navigation parameters be stored in iTrax flash when navigation is stopped? Possible values: 1 - store (semi)permanently to flash when NAV_STOP_MSG is called: Same LSE navigation parameters will be used when navigation is restarted. 0 – do not store to flash when NAV_STOP_MSG is called: Same LSE navigation parameters will not be available when navigation is restarted. -1 – restore LSE navigation parameters to default values: rest of this message is ignored.
INT16	iUseCarrierSmoothing	Should carrier smoothing be used? Possible values: 0 – do not use carrier smoothing (not recommended) 1 – use carrier smoothing (recommended)
INT16	iResidualLimit	Residual limit in meters. Must be greater than 0. Recommended value 20m.
DOUBLE	dHDOPlimit	Worst HDOP value allowed in navigation. Must

type	parameter	description
		be greater than 1. Recommended value 6.5.
INT16	iFilterPar	Reserved for future use Must be set to 0
LONG	lFilterPar	Reserved for future use Must be set to 0
DOUBLE	dFilterPar	Reserved for future use Must be set to 0
DOUBLE	dErrorCovariance	Reserved for future use Must be set to 0
DOUBLE	dEstimVelX	Reserved for future use Must be set to 0
DOUBLE	dEstimVelY	Reserved for future use Must be set to 0
DOUBLE	dEstimVelZ	Reserved for future use Must be set to 0
DOUBLE	dMeasVar	Default value 0.0025
DOUBLE	dInputNoise	Default value 0.5

Figure 27_LSE_PARAMS structure

5.1.4 NAV_START_RECONFIG_MSG

Message id 54.

This message uses _NAV_START structure and thus is identical to NAV_START_MSG. NAV_START_RECONFIG_MSG may be used to set iTalk and NMEA message masks and speeds when not wanting to start navigation.

5.2 Control messages

Host uses control messages to start and stop iTrax02 navigation.

5.2.1 NAV_START_MSG

Message id 51.

Starts the navigation task.

Message structure:

type	parameter	description
INT16	iStartMode	Navigation start mode. Possible values are: 0 - AUTO_START Lets iTrax select the optimal start mode 1 - COLD_START Force cold start (last downloaded ephemeris too old or no ephemeris data, no PVT data) 2 - WARM_START Attempt warm start (currently not supported) 3 - HOT_START Attempt hot start (assume that last received ephemeris data can be used, PT data OK) 4 - QUICK_START Attempt quick start (assume that GPS-time doesn't need to be downloaded either)

type	parameter	description
		10 - AGPS_START, iTrax assumes A-GPS mode and expect ephemeris and time data to be sent by host before start message. Note that if host requests faster start mode than possible (I.e. hot start when there is no ephemeris data available) start mode 0 will be used.
DWORD	dwItalkMsgMask	Bit mask for iTALK messages (port 0). 0 if no iTALK used Host may use this parameter to filter out uninteresting iTrax messages. Bit 0 allows message 1 (PSEUDO_DATA_MSG) to be sent and so on. Host should only allow those messages to be sent which it truly needs. Allowing all messages (by setting filter hex 0xffffffff) might overload messaging system and result unreliable messaging.
DWORD	dwNMEAMsgMask	Bit mask for NMEA messages wanted (port 1). 0 if no NMEA used See above.
DWORD	dwITalkSpeed	Speed of iTALK serial port in bps. 0 if use default value of 115200 (currently only default baudrate == 115200 supported)
DWORD	dwNMEASpeed	Speed of NMEA port in bps. 0 if use default value of 9600
DWORD	dwReserved1	Reserved for future use Must be set to 0
DWORD	dwReserved2	Reserved for future use Must be set to 0

Figure 28_NAV_START structure

5.2.2 NAV_STOP_MSG

Message id 52.

Stops the navigation task.

Message structure:

type	parameter	description
DWORD	dwPwrDown	Should iTrax go to powerdown state? Possible values: 0xffffffff – powerdown until GPIO 11 interrupt, 0 – no powerdown, just stop navigation, any other value n – powerdown for n seconds or GPIO 11 interrupt which ever comes first, then wake up
WORD	wReserved3	Reserved for future use Must be set to 0

Figure 29_NAV_STOP structure

5.3 Aiding messages

Host can use these messages to send iTrax data that it may use as aid in navigation. Note that iTrax responds to these messages by sending them to host's archive task.

5.3.1 AIDING_MSG

Message id 11.

Contains the data that iTrax should use as aiding information.

Message structure:

type	parameter	description
INT16	iUsedCount	Number of usage (internal)
DATE_TIME	date	Local date and time
INT16	iUTCdiff	Difference of local time - UTC. In minutes.
INT16	iAlt	Altitude in meters
INT16	iMagIncl	Magnetic inclination
INT16	iweek	GPS week estimate
LONG	lTow	TOW estimate in full milliseconds
DOUBLE	dTow	Fractional part of TOW estimate in seconds. Exact GPS TOW can be calculated from lTow and dTow with following algorithm: GPSTOW = (lTow/1000.0) + dTow
DOUBLE	dLat	WGS84 latitude estimate in radians
DOUBLE	dLon	WGS84 longitude estimate in radians
DOUBLE	dX	WGS84 X coordinate estimate in meters
DOUBLE	dY	WGS84 Y coordinate estimate in meters
DOUBLE	dZ	WGS84 Z coordinate estimate in meters

Figure 30_AIDING structure

5.3.2 ACQ_AIDING_MSG

Message id 14.

Satellite signal acquirement aiding message. Host can send this to iTrax in order to point out where satellites can be found (in frequency domain)and to switch the SV ACQ aiding on or off.

Message structure:

type	parameter	description
INT16	iSats	Number of satellites
WORD	wAcqOn	Should satellite acquirement aiding be used? Possible values: 0 - switch SV ACQ aiding off 1 - switch SV ACQ aiding on
ACQ_FREQ_AID	AcqAid[MAX_PRNS]	MAX_PRNS = maximum valid PRN. See below.

Figure 31_ACQ_AIDING structure

type	parameter	description
INT16	iElev	Satellite elevation in degrees

INT32	lFreq	Satellite frequency in ACQ domain
INT32	lFreqAcc	Satellite accuracy in ACQ domain

Figure 32_ACQ_FREQ_AID structure

5.4 Misc. messages

5.4.1 NAV_STATE_MSG

Message id 53.

Host sends this as message with no data to iTrax which fills this structure with appropriate data and sends it to host's archive task.

Message structure:

type	parameter	description
INT16	iStarted	
NAV_START	start	_NAV_START structure that was last used to start iTrax. See NAV_START_MSG for definition of this structure.
SW_VERSION	swVersion	Software version of iTrax.
HW_VERSION	hwVersion	Hardware version of iTrax.

Figure 33_NAV_STATE structure

type	parameter	description
WORD	wMajorVersion	Major software revision
WORD	wMinorVersion	Minor software revision
WORD	wBuildNumber	Build number of the software version
WORD	wReserved	Reserved for future use

Figure 34SW_VERSION structure

type	parameter	description
WORD	wFabId	Factory ID
WORD	wSerialYear	Serial year
DWORD	dwSerialNumber	Serial number
DWORD	dwBomRevision	BOM
WORD	wPcbRevision	PCB
DWORD	dwTesterSWRevision	Tester revision
DWORD	dwReserved	Reserved for future use

Figure 35HW_VERSION structure

5.4.2 GPS_SIMULATION_MSG

NAVIGATION_MSG, NAV_KALMAN_MSG and CUSTOM_FIX_MSG may be simulated. Message ids of simulated messages are formed by adding 100 to the message id of the original message. I.e. simulated navigation message would have message id (100 + 7 =) 107.

6. INPUT/OUTPUT MESSAGES

These messages can be sent by host to iTrax or by iTrax to host as well.

6.1 Control/information messages

Host uses these messages to control certain features of iTrax whereas iTrax uses these same messages to pass information about that feature to the host.

6.1.1 AGC_CONTROL_MSG

Message id 13.

Automatic Gain Control message. Host can send this to iTrax and force gain settings (with iIGain and iQGain) or switch AGC on or off. iTrax sends this every time it automatically alters gain settings as information about the new settings.

Message structure:

type	parameter	description
INT16	iIGain	Force I gain control value. Min 0, max 22. See below.
INT16	iQGain	Force Q gain control value. Min 0, max 22. value gain 0 +0dB 1 +2dB 2 +4dB 3 +6dB 4 +8dB 5 +11dB 6 +13dB 7 +15dB 8 +17dB 9 +20dB 10 +22dB 11 +24dB 12 +26dB 13 +29dB 14 +31dB 15 +33dB 16 +35dB 17 +38dB 18 +40dB 19 +42dB 20 +44dB 21 +46dB 22 +48dB
WORD	wIMeas	iTrax measured I (as information to host) Arbitrary units.
WORD	wQMeas	iTrax measured Q (as information to host) Arbitrary units.
INT16	wAutoMode	Should iTrax tune gain automatically? Possible values: 0 – Do not allow AGC. Force iIGain and iQGain. 1 – Allow AGC to tune gain when needed

Figure 36_AGC_CONTROL structure

7. EXAMPLE MESSAGES

This is example NAV_START message that host sends to tell iTrax to start navigation.

sect	type	hex	value	meaning
Start sync. byte 1	BYTE	3C	'<'	Message start synchronization byte 1
Start sync. byte 2.	BYTE	2A	'*'	Message start synchronization byte 2
	WORD	0016	22	Payload length in WORDs
iSys header	WORD	0000	0	Reserved
	WORD	0000	0	Reserved
	WORD	0000	0	Reserved
	WORD	0133	307	Message type (NAV_START_MSG)
	WORD	0000	0	Message number
iTalk header	WORD	0200	512	Source of message (main task of host node)
	WORD	0100	256	Destination (main task of iTrax)
	WORD	0000	0	Transact id (do not reply)
	WORD	000D	13	Length of following data part in WORDS (the _NAV_START structure)
Data = NAV_START structure	INT16	0000	0	Start mode (automatic)
	DWORD	00000040	7th bit	iTalk message mask (only allow NAVIGATION_MSG)
	DWORD	00000000	0	NMEA message mask (no NMEA)
	DWORD	00000000	0	iTalk speed (use default)
	DWORD	00000000	0	NMEA speed (not relevant)
	DWORD	00000000	0	Reserved
	DWORD	00000000	0	Reserved
	WORD	0480	1152	Payload checksum
End sync. byte	BYTE	3E	'>'	Message stop synchronization byte

Note that each word is sent MSB first and LSB last as described in chapter 2. However first 2 bytes and last byte (the synchronization bytes) are sent as separate bytes and therefore byte 0x3c precedes byte 0x2a in start sync. bytes. All other parts of message are full words and thus sent MSB first LSB last.

Example: In the example above the checksum value is 0x0480. This will be sent as bytes 0x04 followed by 0x80.

8. APPENDIX

8.1 _DATE_TIME structure

Many of the structures listed in this document refer to _DATE_TIME structure. It is introduced here.

type	Parameter	Description
INT16	iYear	Year
INT16	iMonth	Month
INT16	iDay	Day
INT16	iHour	Hour
INT16	iMinute	Minute
DOUBLE	dSec	Seconds with fractions

Figure 37_DATE_TIME structure