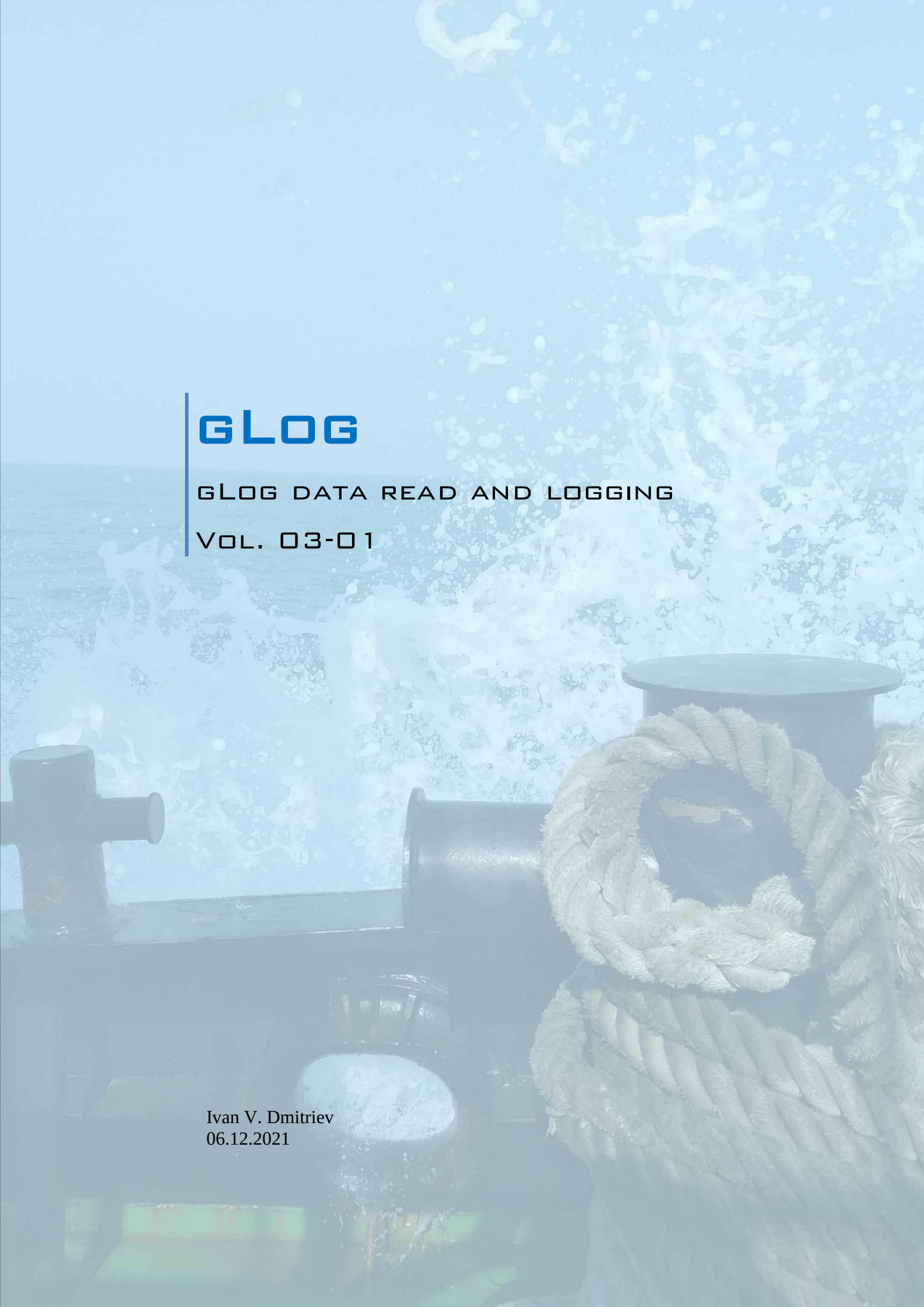


gLog

gLog data read and logging

Vol. 03-01

Ivan V. Dmitriev
06.12.2021

Contents

Tables list

Figures list

1. gLog general description

MatLab functions set for data logging form serial port; it includes several Loggers (***Figure 1.1***) for Windows, programmed in Free Pascal and a number for functions for logged data-flow reading. The Loggers and MatLab functions are shown in ***Table 1.1***.

Table 1.1 gLog functions and Loggers

Software name	Software description
gComLog	Serial port logger for Windows
gComLogOv	Serial port logger for Windows with asynchronous output to file
gComLogOvOut	Serial port logger for Windows with asynchronous output to file and relay data to second serial port
gComLogH	Serial port logger for Windows with asynchronous output to file and possibility to relay data to second serial port; the time can write in 0.1 of microseconds.
gComCC	Cable counters emulation using keyboard.
gCMaxCC	C-Max cable counter data receive and relay
Function name	Function description
gLogGpGguRead	Read $GPGGU data from gLog
gLogGpGgaRead	Read $GPGGA data from gLog
gLogGpGllRead	Read $GPGLL data from gLog
gLogGpHdtRead	Read $GPHDT data from gLog
gLogGpVtgRead	Read $GPVTG data from gLog
gLogGpZdaRead	Read $GPZDA data from gLog
gLogGpGstRead	Read $GPGST data from gLog
gLogTss1Read	Read Tss1 structure from gLog
gLogFixRead	Read fixes structure from gLog
gLogCableCounterRead	Read CableCounter from gLog
gLogGeoSenseNavFixRead	Read $CUSTOM data (Navigation-to-GeoResource48 MultiTrace seismic station) from gLog
gLogGpAvlRead	Read $GPAVL data from gLog
gLogTP2cRead	Read TP2 compatible output data (USBL) from gLog
gLogMagyReadSeaSpy	Read SeaSpy Magnetometer data from gLog
gLogMagyReadSeaSpyGrad	Read SeaSpy Gradiometer data from gLog
gLogGpsCompTimeDelta	Difference between CompTime and GpsTime
gLogGpsDayCalc	Calculate GpsDay using CompDay, CompTime, GpsTime.
gLogFilesFind	Create FilesList for time section
gLogFileFormatFix	Find “bad” strings for files logged
gLogCheckSum	Checksum calculation for NMEA message

The several data flows can be recorded using several Loggers, which run simultaneously.

The Data acquisition scheme is shown in ***Figure 1.2***. There are two binding/linking time sources defined:

-- the “global” source is GPS-data flow with 1 second discrecity;

-- the “local” source is the computer clock data flow with 0.001 second discrecity (or 0,1 microsecond discrecity for gComLogH).

On the one hand, the computer clock time is used for interpolation inside GPS 1-second intervals; on the other hand the GPS clock applies long-time-drift correction to computer's clock data.

gLog apply the Computer's clock time to each recorded data-message. This Computer-time can be used for Sensors' Data synchronization/linking within the Computer (includes GPS-time data).

The GPS-time data (recorded by gLog) is used for Computer-time and GPS-time synchronization/linking; the GPS-time can be linked to each Sensor data using Computer-time marks. A number of computers (Computers' clock) can be synchronized/linked using GPS-time data flow.

The RTK-GPS data flow is used as high-accuracy coordinates' source. The RTK-GPS coordinates are received with delay by Computer's clock (with real-time delay); the GPS-time data field is used for RTK-GPS coordinates binding/linking to another Sensors' Data.

The External Survey Software is applied to Survey data the Computers' clock marks too (MagLog's files, for example), Raw-GPS marks (any software to seg-y files, for example) or both marks types (jsf files, for example). We can link External Survey Software Data to other Sensors using this marks types or using any another data flow simultaneously incoming to External Survey Software and to gLog.

The simplified Data synchronization scheme is shown in ***Figure 1.3***. Recorded data-blocks had identical fields; these fields are used for other fields' data synchronization/linking.

The Loggers data, from sensors, are writing to own "Log-folders" (***Figure 1.4***). Using gLog functions the data-flow can be read from folders. The file names includes date and time by Computer clock; this date/time can be used for read files choice.

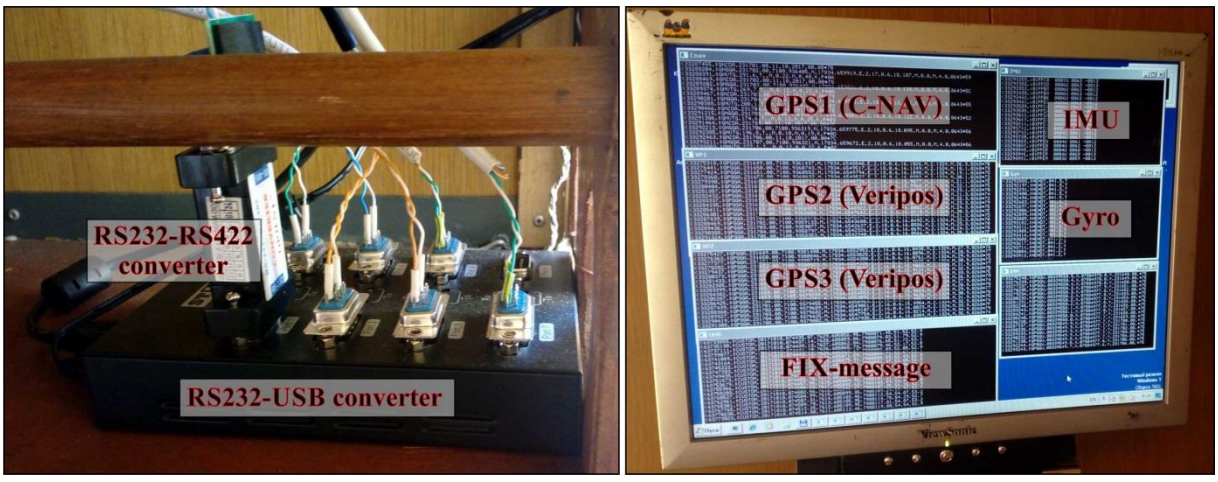


Figure 1.1 Navigation sensor's acquisition system (ultra-low-cost; peak time error ±0.003 seconds)

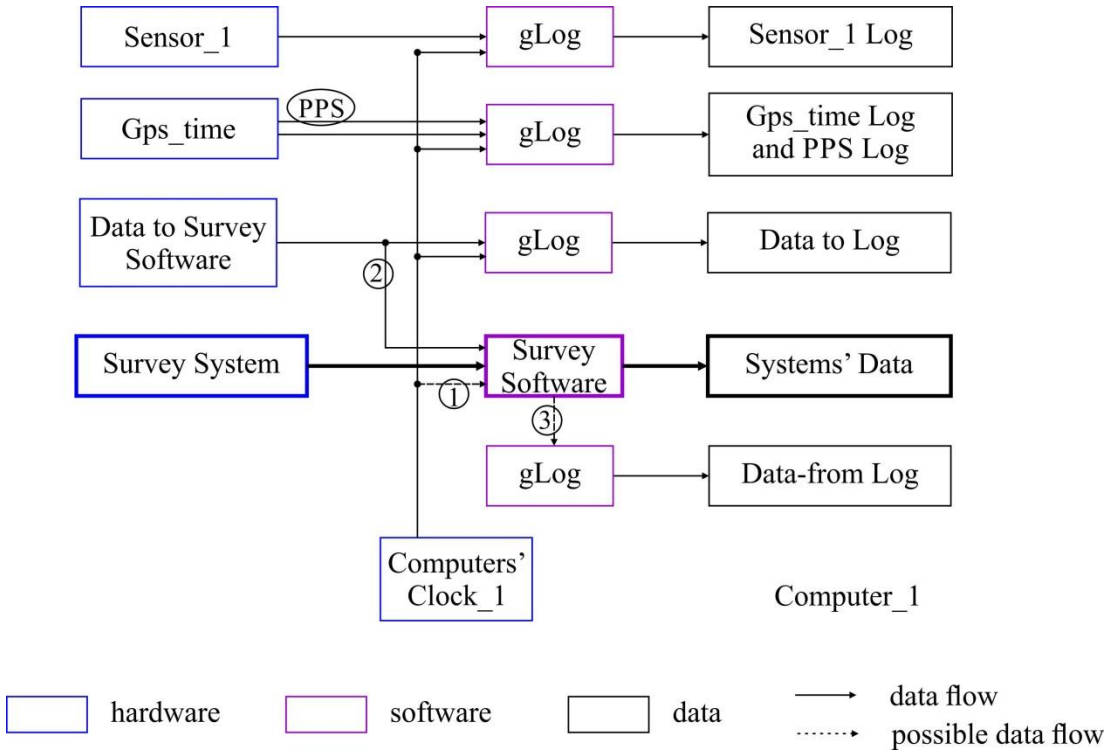


Figure 1.2 Data acquisition scheme

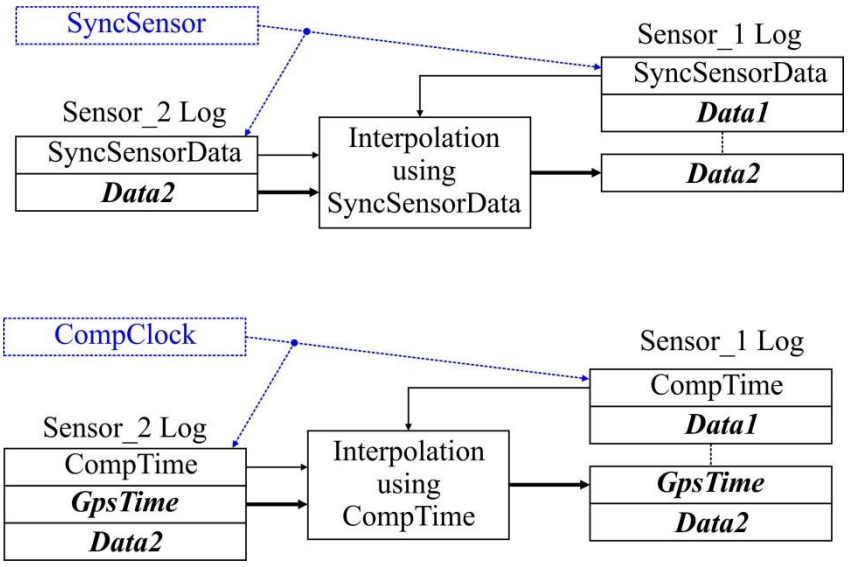


Figure 1.3 Interpolate ("synchronized") Data2 and GpsTime to Data1

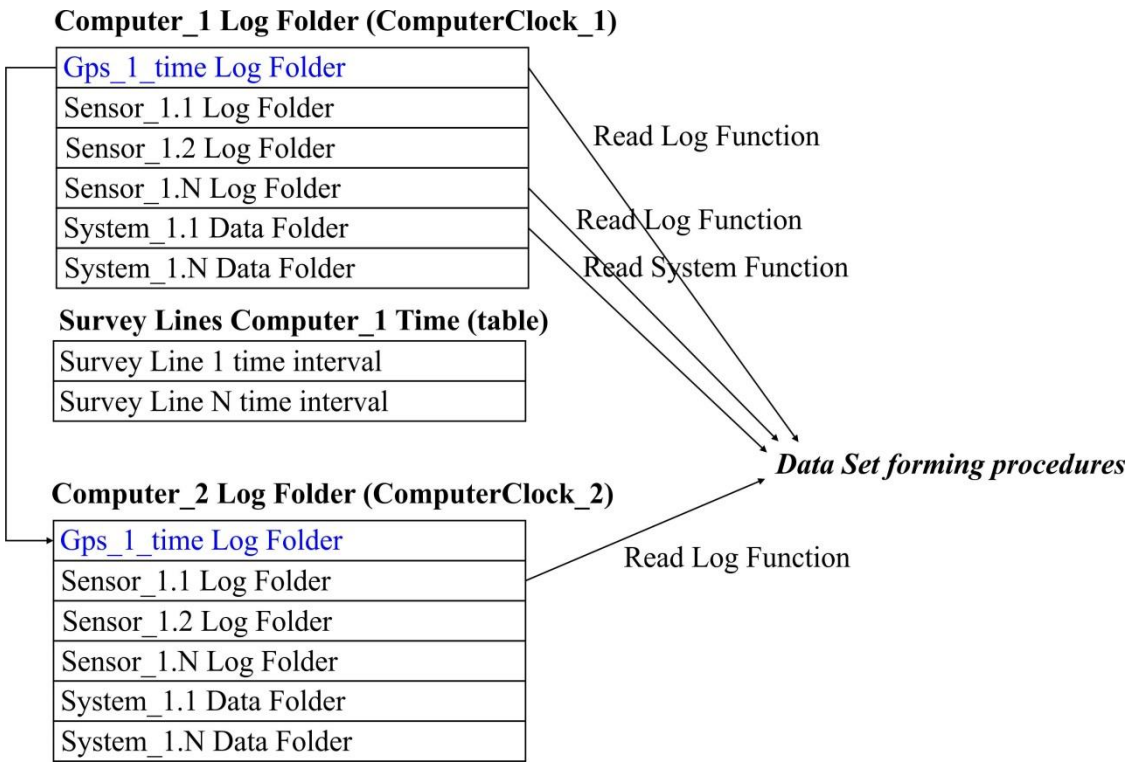


Figure 1.4 Sensors' Data Heap (folders structure for gLog/ge0mlib)

2. Data Loggers

The full functionality logger is gComLogH (2020.03.03); gComLog (2017.01.04), gComLogOv (2019.07.31), gComLogOvOut (2019.11.14) are partly-functional previous versions.

2.1 gComLog

gComLog (2017.01.04) is serial port logger for Windows (*Figure 2.1*). Free Pascal souses applied. Used kernell32.dll functions CreateFile, ReadFile; not used WaitCommEvent.

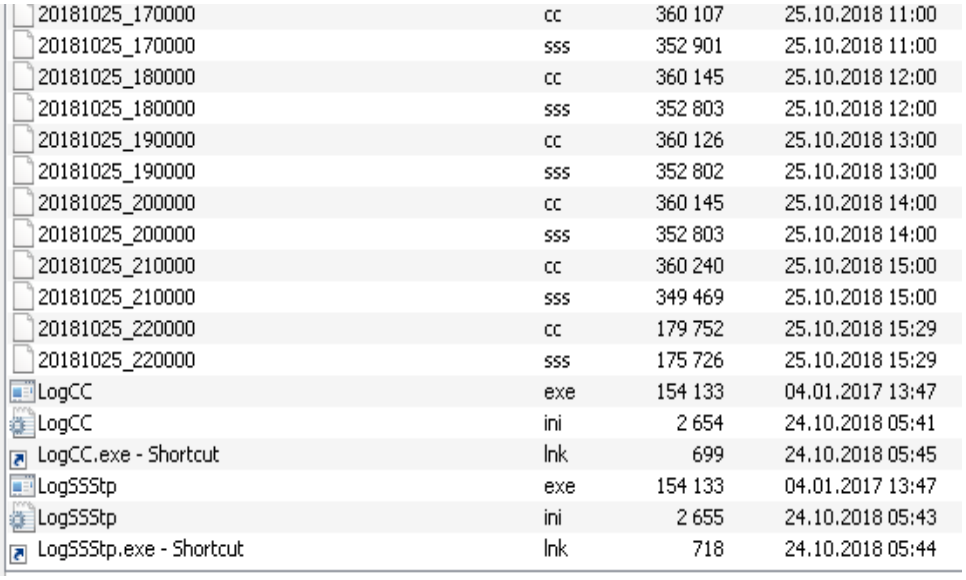
Figure 2.1 gComLog window

The program is open (search) ini-file (*Table 2.1*) in current directory with the name same exe-file. You can rename exe-file and ini-file with the same name. So, the several program copies can be start from one directory.

Table 2.1 Ini-file lines description for gComLog

Line N	Config example	Configuration file Description
Line1:	\\.\COM6	ComName.
Line2:	9600 8 0 0	BaudRate, ByteSize, Parity, StopBits. BaudRate: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Parity: NOPARITY==0; ODDPARITY==1; EVENPARITY==2; MARKPARITY==3; SPACEPARITY==4. StopBits: ONESTOPBIT==0; ONE5STOPBITS==1; TWOSTOPBITS==2.
Line3:	10 100	End Of Line symbol code; End Of Line time in ms.
Line4:	4096	RAM buffer size (when buffer full data moved to log-file). The gComLogOv used two buffers.
Line5:	.log	Log-file extension.
Line6:	D	Autobreak code for log-file. 'H'- new file will be created every hour; 'D'- every day; another- no auto break.
Line7:	~,	LeftDelimiter, RightDelimiter.
Line8:	11	DTR and RTS state. Example1:11 (set RTS, set DTR) Example2:00 (clr RTS, clr DTR) Example3:99 (will not manage RTS/DTR)

Program write log-files in current directory with 'data-and-time' name: YYYYMMDD_HHMMSS + string from ini-file's Line 5 (*Figure 2.2*). Program can create new file every hour or every day (Line 6). You can break current file and start new file (from End Of Line symbol) any time, using 'C' key.



20181025_170000	cc	360 107	25.10.2018 11:00
20181025_170000	sss	352 901	25.10.2018 11:00
20181025_180000	cc	360 145	25.10.2018 12:00
20181025_180000	sss	352 803	25.10.2018 12:00
20181025_190000	cc	360 126	25.10.2018 13:00
20181025_190000	sss	352 802	25.10.2018 13:00
20181025_200000	cc	360 145	25.10.2018 14:00
20181025_200000	sss	352 803	25.10.2018 14:00
20181025_210000	cc	360 240	25.10.2018 15:00
20181025_210000	sss	349 469	25.10.2018 15:00
20181025_220000	cc	179 752	25.10.2018 15:29
20181025_220000	sss	175 726	25.10.2018 15:29
LogCC	exe	154 133	04.01.2017 13:47
LogCC	ini	2 654	24.10.2018 05:41
LogCC.exe - Shortcut	lnk	699	24.10.2018 05:45
LogSSStp	exe	154 133	04.01.2017 13:47
LogSSStp	ini	2 655	24.10.2018 05:43
LogSSStp.exe - Shortcut	lnk	718	24.10.2018 05:44

Figure 2.2 Folders' content example (loggers, ini-files and log-files)

The program wait End_Of_Line symbol (Line 3) from serial port to detect "end of data message"; the follow received byte will linked to (day's second number)*1000. If program wait data longer than End_Of_Line time (Line 3), it terminate "end of data message" waiting and write (day's second number)*1000 for follow received byte.

Program prints each "data message" on the screen, when End_Of_Line symbol was received. When program move RAM buffer to file, it prints all c symbols from buffer to screen (not depend from End_Of_Line symbol).

Program not used WaitCommEvent function. Not make RAM buffer size big for data lost or false time-stamp exception.

Keys:

ESC – Hard Exit;

Q – Soft Exit (wail End_Of_Line symbol from Com);

C – Break log-file and start new.

Log Example:

~04268163,$GPRMC,005121.639,A,5000.00024,N,04500.00869,E,10.00,88.1,210714,0.0,E*57

~04268163,$GPGLL,5000.00024,N,04500.00869,E,005121.639,A*3A

~04268165,$GPGGA,005121.639,5000.00024,N,04500.00869,E,1,5,0.0,0.0,M,0.0,M,,*57

There are:

~ – LeftDelimiter;

04268165 – (Day's second number)*1000;

, – RightDelimiter;

$GP.. – Data block.

2.2 gComLogOv

gComLogOv (2019.07.31) is serial port logger for Windows the same to gComLog. There are follow differences:

- The two RAM buffers are used;
- The overlap structure is used for asynchronous output to file (form filled RAM buffer).

There is no delay for time stamps by the "save-to-file" evidence, that take place for gComLog.

2.3 gComLogOvOut

gComLogOvOut (2019.11.14) is serial port logger for Windows the same to gComLogOv, but it used additional serial port for transmit immediately each byte was received. The ini-file lines are shown in ***Table 2.2***.

Table 2.2 Ini-file lines description for gComLogOvOut

Line N	Config example	Configuration file Description
Line1:	\\.\COM6	ComName for data Input
Line2:	\\.\COM7	ComName for data Output
Line3:	9600 8 0 0	BaudRate, ByteSize, Parity, StopBits. BaudRate: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Parity: NOPARITY==0; ODDPARITY==1; EVENPARITY==2; MARKPARITY==3; SPACEPARITY==4. StopBits: ONESTOPBIT==0; ONE5STOPBITS==1; TWOSTOPBITS==2. For com Input and Output
Line4:	10 100	End Of Line symbol code; End Of Line time in ms.
Line5:	4096	RAM buffer size (when buffer full data moved to log-file). The gComLogOv used two buffers.
Line6:	.log	Log-file extension.
Line7:	D	Autobreak code for log-file. 'H'- new file will be created every hour; 'D'- every day; another- no auto break.
Line8:	~,	LeftDelimiter, RightDelimiter.
Line9:	11	DTR and RTS state. Example1:11 (set RTS, set DTR) Example2:00 (clr RTS, clr DTR) Example3:99 (will not manage RTS/DTR) For com Input only.

2.4 gComLogH

gComLogH (2020.03.03) is serial port logger for Windows the includes all specific features from gComLog, gComLogOv, gComLogOvOut:

- The two RAM buffers are used;
- The overlap structure is used for asynchronous output to file (form filled RAM buffer);
- Additional serial port for transmitting each byte was received.

The additional feature is the possible to use 'GetSystemTimePreciseAsFileTime' function to get time (named below as a P-time-mode); 'Now' function was used in previous software versions. The 'GetSystemTimePreciseAsFileTime' is not presented for Windows Systems earlier than Windows Vista; so the 'Now' function is keep in software. The function will used is defined in the ini-file (Line10).

Warning: P-time-mode (in 0,1 microsecond), presented UTC-time; old-time-mode (in millisecond) presented LocalTime. Please correct the time zone in accordance your aim.

There are **8th symbols** in Day's second number*1000 for old-time-mode (in millisecond). There are **12th symbols** in Day's second number*10 000 000 for P-time-mode (in 0,1 microsecond).

The ini-file lines are shown in ***Table 2.3***.

Table 2.3 Ini-file lines description for gComLogH

Line N	Config example	Configuration file Description
Line01:	\\.\COM6	Com1 Name for data Input
Line02:	\\.\COM7	OutputCom2 Name for data Output **If the first symbol is not '\' then OutputCom is not used.**
Line03:	9600 8 0 0	BaudRate, ByteSize, Parity, StopBits. Used for Com1 and OutputCom2 configuration. BaudRate: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Parity: NOPARITY==0; ODDPARITY==1; EVENPARITY==2; MARKPARITY==3; SPACEPARITY==4. StopBits: ONESTOPBIT==0; ONE5STOPBITS==1; TWOSTOPBITS==2.
Line04:	10 2000000	End Of Line symbol code; End Of Line **time in milliseconds** (if Line11~='P') **or 0,1 microseconds** (if Line11=='P').
Line05:	4096	RAM-buffer size in bytes (when buffer full data moved to log-file). The two swapped buffers are used.
Line06:	.log	Log-file's name last symbols (extension as a particular case).
Line07:	D	Autobreak code for log-file. 'H'- new file will be created every hour; 'D'- every day; another- no auto break.
Line08:	~,	LeftDelimiter, RightDelimiter.
Line09:	1111	**Com1 and OutputCom2 DTR and RTS state.** Example1:1111 (set RTS1, set DTR1, set RTS2, set DTR2) Example2:0000 (clr RTS1, clr DTR1, clr RTS2, clr DTR2) Example3:9999 (will not manage RTS/DTR).
Line10	P	**TimeMode. If set 'P' than 12th chars length "time in 0.1 microsecond"** write to file ('GetSystemTimePreciseAsFileTime' function is used to get time); else 8th symbols "time in 0.1 millisecond" write to file ('Now' function is used to get time).

2.5 gComCC

gComCC (2019.05.24) is cable counter emulation for Windows using keyboard. Program sends "CableCounter string" to log-file and to serial port. Free Pascal sources applied. Used kernell32.dll functions CreateFile, ReadFile; not used WaitCommEvent.

Program will open (search) ini-file in current directory with the name same exe-file. You can rename exe-file and ini-file with the same name. So, the several program copies can be start from one directory.

Program write log-files in current directory with 'data-and-time' name: YYYYMMDD_HHMMSS + string from ini-file's Line 3.

Program can create new file every hour or every day (Line 4). You can break current file and start new file (from End Of Line symbol) any time, using 'C' key.

Keys:

ESC – Hard Exit;

Q – Soft Exit (wail End_Of_Line symbol from Com);

C – Break log-file and start new;

“=” – Counter +1;

“-” – Counter -1 (cannot less than zero).

Table 2.4 Ini-file lines description for gComCC

Line N	Config example	Config file Description
Line1:	\\.\COM212	ComName.
Line2:	9600 8 0 0	BaudRate, ByteSize, Parity, StopBits. BaudRate: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Parity: NOPARITY==0; ODDPARITY==1; EVENPARITY==2; MARKPARITY==3; SPACEPARITY==4. StopBits: ONESTOPBIT==0; ONE5STOPBITS==1; TWOSTOPBITS==2.
Line3:	.log	Log-file extension.
Line4:	H	Autobreak code for log-file. 'H'- new file will be created every hour; 'D'- every day; another- no autobreak.
Line5:	~,	LeftDelimiter, RightDelimiter.

Log Example:

~57606379,0021

~57607942,0022

~57609708,0023

~57642602,0022

~57644305,0021

There are:

~ – LeftDelimiter;

57606379 – (Day's second number)*1000;

, – RightDelimiter;

0021 – Data block.

2.6 gCMaxCC

gCMaxCC (2016.10.04) is the small terminal-window software for C-Max cable counter data receive and relay to another serial port (same to C-Max Pulley software). The ini-file lines description shown in ***Table 2.5***.

Keys:

ESC – Hard Exit;

Table 2.5 Ini-file lines description for gCMaxCC

Line N	Config example	Config file Description
Line1:	\\.\COM4	ComName for CC
	VRFLN	Init string for CC
	P+0002	Init length for CC
	\\.\COM11	ComName for Relay
Line2:	9600 8 0 0	BaudRate, ByteSize, Parity, StopBits. BaudRate: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Parity: NOPARITY==0; ODDPARITY==1; EVENPARITY==2; MARKPARITY==3; SPACEPARITY==4. StopBits: ONESTOPBIT==0; ONE5STOPBITS==1; TWOSTOPBITS==2.
Line3:	00	DTR and RTS state. Example1:11 (set RTS, set DTR) Example2:00 (clr RTS, clr DTR) Example3:99 (will not manage RTS/DTR)

The pulley connects to a serial interface and operates at 4800 baud with no handshaking. The data flow format shown in ***Table 2.6***. The computer's DB9 pins for C-Max cable counter connection shown in ***Table 2.7***; DTR and RTS should be set to active as they are used to provide power.

Table 2.6 C-Max cable counter message (raw data)

4th digit	3rd digit	2nd digit	1st digit	EOL	EOL
0..9	0..9	0..9	0..9	0x0D	0x0A
0	0	0	8	0x0D	0x0A

Table 2.7 Computer's DB9 pins for C-Max cable counter connection

Pin Number	In/Out	Description
1		Not used
2	In	RD
3	Out	TD
4	Out	DTR (power)

Pin Number	In/Out	Description
5	In	SG
6		Not used
7	Out	RTS (power)
8		Not used
9		Not used

The pulley starts in polled mode, giving a '[+/-]XXXX<CR>' string in response to a 'T' command but it can be set to transmit this string once per meter if needed. This non-polled mode needs to be set each time the pulley is connected by sending a 'N' command as there is no on-board non-volatile memory. The commands to the pulley such as 'N' or 'T' are single capital characters, with no <CR> needed.

The command list is:

R = reset counter to zero;

T = transmit count [2Bh (+), 30h (0), 30h (0), 30h (0), 30h (0), 0Dh (.)];

I = identify (returns 'P' - 50h);

P[+/-XXXX]<CR> = preset counter (version 4 or later) P+0020 P-0005;

F = set count direction forward;

B = set count direction backward;

V = transmit software version (example: Version 4.0 (C) C-MAX Ltd 2006);

L = append LF/CR to count;

C = append CR to count;

N = interrupt mode;

O = polled mode;

S = transmit status string (example: FCO);

X = swap End Of Line 0x0D to 0x0D 0x0A (not works?).

The status string consists of four characters. The first indicates the count direction ('F'/'B'), followed by count string termination type ('L'/'C') and then transmission mode ('O'/'N'), then finally a <CR> character. The power on status is 'FCO'.

3. gLog Read/Write functions

3.1 Read GPGGU

function GPGGU=gLogGpGguRead(fName,keyS,LDelim,CompTimeLocShift)

Read $GPGGU data from files created by gComLog program.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPGGU' or same);

LDelim – left delimiter for log-file;

CompTimeLocShift – Computer time minus Utc_Gps time (in seconds);

GPGGU – reading data structure ().

File Example:

<00011878,$GPGGU, 284619.5,X, 1726702.6,Y,160307.00,*76

or

<00012876,$GPGGU, 284616.3,X, 1726701.7,Y,160308.00*72

Where

< – left delimiter (symbol 1);

00012876 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$GPGGU – GPGGU data.

Example:

```
GPGGU=gLogGpGguRead(('c:\temp\','$GPGGU','<',10*3600);
```

Table 3.1 GPGGU structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsDay	Gps date number 1 corresponds to Jan-1-000; the year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
GpsTime	GPS's second in day. Vector; Row-content.
CompTimeShift	Calculated mean difference between Computer time and Gps time in seconds, calculated for structure time-interval. Scalar; Information-content.
CompTimeDelta	Calculated difference between Computer time and Gps time, calculated for each point decremented to CompTimeShift. Vector; Row-content.
GpsE	Calculated rectangular projection N for GPS. Vector; Row-content.
GpsN	Calculated rectangular projection E for GPS. Vector; Row-content.

GPGGU format description

$GPGGU, 284616.3,X, 1726701.7,Y,160308.00,*72

284616.3,X – Easting (m).

1726701.7,Y – Northing (m).

160308.00 – Fix taken at 16:03:08.00 UTC.

*72 – Checksum data, always begins with *.

3.2 Read GPGGA

function GPGGA=gLogGpGgaRead(fName,keyS,LDelim,CompTimeLocShift)

Read $GPGGA data from files was created by gComLog software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPGGA' or same);

LDelim – left delimiter for log-file;

CompTimeLocShift – Computer time minus Utc_Gps time (in seconds);

GPGGA – reading data structure (***Table 3.2***).

File Example:

~38995230,$GPGGA,004852.00,4549.3983338,N,14140.1657521,E,1,16,0.7,6.0013,M,27.7073,M,,*63

~38996240,$GPGGA,004853.00,4549.3987483,N,14140.1706720,E,1,16,0.7,5.8322,M,27.7073,M,,*6D

Where

~ – left delimiter (symbol 1);

38995230 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$GPGGA... – GPGGA data.

Using functions: gNavGpsDayCalc, gNavGpsCompTimeDelta.

Example:

```
>> GPGGA=gLogGpGgaRead('c:\temp\','~',10);
```

Table 3.2 GPGGA structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsDay	Gps date number 1 corresponds to Jan-1-000; the year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
GpsTime	GPS's second in day. Vector; Row-content.
CompTimeShift	Calculated mean difference between Computer time and Gps time in seconds, calculated for structure time-interval. Scalar; Information-content.
CompTimeDelta	Calculated difference between Computer time and Gps time, calculated for each point decremented to CompTimeShift. Vector; Row-content.
GpsLat	Latitude (WGS84 usually) for GPS; DD.DDDDDD. Vector; Row-content.
GpsLon	Longitude (WGS84 usually) for GPS; DDD.DDDDDD. Vector; Row-content.

GpsE	Calculated rectangular projection N for GPS. Vector; Row-content.
GpsN	Calculated rectangular projection E for GPS. Vector; Row-content.
GpsFixQuality	Fix quality: 0 = invalid; 1 = GPS fix (SPS); 2 = DGPS fix; 3 = PPS fix; 4 = Real Time Kinematic; 5=Float RTK; 6 = estimated (dead reckoning) (2.3 feature); 7 = Manual input mode; 8=Simulation mode.
GpsSatNum	Number of satellites being tracked. Vector; Row-content.
GpsHorizDilution	Horizontal dilution of position. Vector; Row-content.
GpsAltSea	Altitude, Meters, above mean sea level for GPS. Vector; Row-content.
GpsHgtGeoid	Height of geoid (mean sea level) above ellipsoid (WGS84 usually) for GPS. Vector; Row-content.
GpsDgpsUpdate	Time in seconds since last DGPS update. Vector; Row-content.
GpsDgpsId	DGPS station ID number. Vector; Row-content.

GPGGA format description

$GPGGA,004852.00,4549.3983338,N,14140.1657521,E,1,16,0.7,6.0013,M,27.7073,M,08,1004*63

004852.00 – Fix taken at 00:48:52.00 UTC.

4549.3983338,N – Latitude 45 deg 49.3983338' N.

14140.1657521,E – Longitude 141 deg 40.1657521' E.

Fix quality: 0 = invalid; 1 = GPS fix (SPS); 2 = DGPS fix; 3 = PPS fix; 4 = Real Time Kinematic; 5=Float RTK; 6 = estimated (dead reckoning) (2.3 feature); 7 = Manual input mode; 8=Simulation mode.

16 – Number of satellites being tracked.

0.7 – Horizontal dilution of position.

6.0013,M – Altitude, Meters, above mean sea level.

27.7073,M – Height of geoid (mean sea level) above WGS84 ellipsoid.

08 – Time in seconds since last DGPS update.

1004 – DGPS station ID number.

*63 – Checksum data always begins with *.

3.3 Read GPGLL

function GPGLL=gLogGpGllRead(fName,keyS,LDelim,CompTimeLocShift)

Read $GPGLL data from files was created by gComLog software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPGLL' or same);

LDelim – left delimiter for log-file;

CompTimeLocShift – Computer time minus Utc_Gps time (in seconds);

GPGLL – reading data structure (***Table 3.3***).

Log-file (created by gComLog software) example:

<68183656,$GPGLL,4303.70906,N,13152.96378,E,080636.00,A*04

<68184656,$GPGLL,4303.70906,N,13152.96378,E,080636.99,A*04

Where

< – left delimiter (symbol 1);

38995230 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$GPGLL... – GPGLL data.

Using functions: gNavGpsDayCalc, gNavGpsCompTimeDelta.

Example:

```
>> GPGLL=gLogGpGllRead('c:\temp\','<',10);
```

Table 3.3 GPGLL structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsDay	Gps date number 1 corresponds to Jan-1-000; the year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
GpsTime	GPS's second in day. Vector; Row-content.
CompTimeShift	Calculated mean difference between Computer time and Gps time in seconds, calculated for structure time-interval. Scalar; Information-content.
CompTimeDelta	Calculated difference between Computer time and Gps time, calculated for each point decremented to CompTimeShift. Vector; Row-content.
GpsLat	Latitude (WGS84 usually) for GPS; DD.DDDDDD. Vector; Row-content.
GpsLon	Longitude (WGS84 usually) for GPS; DDD.DDDDDD. Vector; Row-content.
GpsE	Calculated rectangular projection N for GPS. Vector; Row-content.
GpsN	Calculated rectangular projection E for GPS. Vector; Row-content.
GpsFixQuality	Fix quality: 0 = invalid; 1 = GPS fix (SPS); 2 = DGPS fix; 3 = PPS fix; 4 = Real Time Kinematic; 5=Float RTK; 6 = estimated (dead reckoning) (2.3 feature); 7 = Manual input mode; 8=Simulation mode.

GPGLL format description

$GPGLL,4303.70906,N,13152.96378,E,080636.00,A*04

4303.70906,N – Latitude 45 deg 03.70906' N.

13152.96378,E – Longitude 131 deg 52.96378' E.

080636.00 – Fix taken at 08:06:36.00 UTC.

Status field: A = Data valid; V = Data not valid.

*04 – Checksum data, always begins with *.

3.4 Read GPHDT

function GPHDT=gLogGpHdtRead(fName,keyS,LDelim)

Read $GPHDT data from files was created by gComLog software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPHDT' or same);

LDelim – left delimiter for log-file;

GPHDT – reading data structure (*Table 3.4*).

File Example:

~39600366,$INHDT,154.0,T*25

~39601360,$INHDT,153.9,T*2B

Where

~ – left delimiter (symbol 1);

39600366 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$INHDT... – $GPHDT data.

Example:

```
>> GPHDT=gLogGpHdtRead('c:\temp\','$INHDT','~');
```

Table 3.4 GPHDT structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
Heading	Heading from Nmea HDT. Vector; Row-content.

GPHDT format description

$GPHDT,154.0,T*25

154.0 – Heading degree.

T – True.

*25 – Checksum data, always begins with *.

3.5 Read GPVTG

function GPVTG=gLogGpVtgRead(fName,keyS,LDelim)

Read $GPVTG data from files created by gComLog program.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPVTG' or same);

LDelim – left delimiter for log-file;

GPVTG – reading data structure (*Table 3.5*).

File Example:

~00007880,$GPVTG,167.5,T,,M,6.16,N,11.40,K,A*23

~00008879,$GPVTG,167.3,T,,M,6.24,N,11.55,K,A*20

Where

~ – left delimiter (symbol 1);

00007880 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$GPVTG... - $GPVTG data.

Example:

```
>> GPVTG=gLogGpVtgRead('c:\temp\','$GPVTG','~');
```

Table 3.5 GPVTG structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsCourseT	True Course over ground, Degrees. Vector; Row-content.
GpsCourseM	Magnetic Course over ground, Degrees. Vector; Row-content.
GpsSpeed	Speed over ground, Km/hr. Vector; Row-content.
VtgPosInd	Positioning system Mode Indicator: A=Autonomous mode; D=Differential mode; E=Estimated (dead reckoning) mode; M=Manual input mode; S=Simulator mode; N=Data not valid. Chars Vector; Row-content.

GPVTG format description

$GPVTG,167.5,T,160,M,6.16,N,11.40,K,A*23<CR><LF>

167.5 – Course over ground.

T – Degrees, True.

160 – Course over ground.

M – Degrees, Magnetic.

6.16 – Speed over ground.

N – knots.

11.40 – Speed over ground.

K – Km/hr .

A – Positioning system Mode Indicator: A=Autonomous mode; D=Differential mode; E=Estimated (dead reckoning) mode; M=Manual input mode; S=Simulator mode; N=Data not valid.

*63 – Checksum data, always begins with *.

3.6 Read GPZDA

function GPZDA=gLogGpZdaRead(fName,keyS,LDelim)

Read $GPZDA data from files were created by gComLog software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPZDA' or same);

LDelim – left delimiter for log-file;

GPZDA – reading data structure (*Table 3.6*).

File Example:

~39600144,$INZDA,235959.0034,22,05,2016,,*70

~39601132,$INZDA,000000.0034,23,05,2016,,*70

Where

~ – left delimiter (symbol 1);

39600144 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$INZDA... – $GPZDA data.

Example:

```
>> GPZDA=gLogGpZdaRead('c:\temp\','$GPZDA','~');
```

Table 3.6 GPGLL structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsDay	Gps date number 1 corresponds to Jan-1-000; the year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
GpsTime	GPS's second in day. Vector; Row-content.
CompTimeShift	Calculated mean difference between Computer time and Gps time in seconds, calculated for structure time-interval. Scalar; Information-content.
CompTimeDelta	Calculated difference between Computer time and Gps time, calculated for each point decremented to CompTimeShift. Vector; Row-content.
GpsLocDay	Local zone hours -13..13
GpsLocTime	Local zone minutes 0..59

GPZDA format description

$GPZDA,235959.0034,22,05,2016,13,59*70

235959.0034 – Fix taken at 23:59:59.0034 UTC.

22 – UTC day.

05 – UTC month.

2016 – UTC year.

13 – Local zone hours, 00 to ±13 hrs

59 – Local zone minutes, 00 to +59

*63 – Checksum data, always begins with *.

3.7 Read GPGST (under construction)

function GPGST=gLogGpGstRead(fName,keyS,LDelim)

Read $GPGST data from files created by gComLog program.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPGST' or same);

LDelim – left delimiter for log-file;

GPGST – reading data structure (*Table 3.7*).

File Example:

Where

~ – left delimiter (symbol 1);

00007880 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$GPGST... - $GPGST data.

Example:

```
>> GPGST=gLogGpGstRead('c:\temp\','$GPGST','~');
```

Table 3.7 GPVTG structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsDay	Gps date number 1 corresponds to Jan-1-000; the year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
GpsTime	GPS's second in day. Vector; Row-content.
CompTimeShift	Calculated mean difference between Computer time and Gps time in seconds, calculated for structure time-interval. Scalar; Information-content.
CompTimeDelta	Calculated difference between Computer time and Gps time, calculated for each point decremented to CompTimeShift. Vector; Row-content.
GpsStdA	Root mean square (rms) value of the standard deviation of the range inputs to the navigation process. Range inputs include pseudoranges and differential GNSS (DGNSS) corrections. Vector; Row-content.
GpsStdB	Standard deviation of semi-major axis of error ellipse (meters). Vector; Row-content.
GpsStdC	Standard deviation of semi-minor axis of error ellipse (meters). Vector; Row-content.
GpsStdD	Orientation of semi-major axis of error ellipse (meters). Vector; Row-content.
GpsStdE	Standard deviation of latitude error (meters). Vector; Row-content.
GpsStdF	Standard deviation of longitude error (meters). Vector; Row-content.
GpsStdG	Standard deviation of altitude error (meters). Vector; Row-content.

GPPGST format description

$GPGST,hhmmss.ss,a.a,b.b,c.c,d.d,e.e,f.f,g.g *cc<CR><LF>

hhmmss.ss – UTC time in hours, minutes, seconds of the GPS position

a.a – Root mean square (rms) value of the standard deviation of the range inputs to the navigation process. Range inputs include pseudoranges and differential GNSS (DGNSS) corrections

b.b – Standard deviation of semi-major axis of error ellipse (meters)

c.c – Standard deviation of semi-minor axis of error ellipse (meters)

d.d – Orientation of semi-major axis of error ellipse (meters)

e.e – Standard deviation of latitude error (meters)

f.f – Standard deviation of longitude error (meters)

g.g – Standard deviation of altitude error (meters)

*cc – Checksum data, always begins with *.

3.8 Read TSS1

function Tss1=gLogTss1Read(fName)

Read Tss1 structure from files was created by gComLog software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

Tss1 – Tss1 structure (***Table 3.8***).

Example:

```
>> Data=gLogTss1Read('c:\temp\20160408_104958.mtn');
>> plot((Tss1.CompTime(:)-Tss1.CompTime(1))./100,Tss1.Heave);
```

Table 3.8 TSS1 structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name;
CompTime	Computer time from LOG-file (second per day)
MotionLAH	(HorizA) horizontal acceleration in cm/s2;
MotionLAZ	(VertA) vertical acceleration in cm/s2;
MotionHeave	Heave in m;
MotionF	Status flag: "U","G","H","F" indicate a settled condition; "u","g","h","f" are given during the 3 minutes settling period. "U" is unaided mode,"G" is GPS aided mode, "H" is heading aided mode, "F" is full aided mode. For TSS 320 messages, the status field is used to indicate heave quality control: space = OK, "?" = FAIL.
MotionRoll	Roll in degree;
MotionPitch	Pitch in degree

TSS1 packet description

TSS1 PACKET :XXAAAA_MHHHHQMRRRR_MPPPPZZ (***Table 3.9***).

Table 3.9 TSS1 format description

Field	Format	Description	Range	Units
01	“:”	start of packet character	“:”	3A Hex
02	XX	horizontal acceleration	0 to 9.81 m/s2	3.83 cm/s2
03	AAAA	vertical acceleration	–20.48 to +20.47 m/s2	0.0625 cm/s2
04	_	space character	“ “	20 Hex
05	MHHHH	heave	–99 to +99 m	0.01 m
06	Q	status flag	“u,U,g,G,H,f,F”, “?”, “ “	see note
07	MRRRR	roll	–99 to +99	0.01 °
08	_	space character	“ “	20 Hex
09	MPPPP	pitch	–99 to +99	0.01 °
10	ZZ	termination characters	<CR><LF>	0D Hex 0A Hex

Format Examples

:003D50 0000 1482 0085

:003D4E 0000 -1490 0065

:003D51 -0005 -0037 0074

The TSS1 data string contains 27 characters in five data fields.

A data status flag is included in the packet. This status flag can be modified by the user to allow compatibility with other Teledyne TSS products.

A heading status flag is also included in the packet identifying the compass status. This status flag can also be modified by the user to allow compatibility with other Teledyne TSS products. Refer to Table 6-24 below for flag details.

The acceleration fields contain ASCII-coded hexadecimal values: Horizontal acceleration uses units of 3.83cm/sI in the range zero to 9.81m/sI; Vertical acceleration uses units of 0.0625cm/sI in the range –20.48 to +20.48m/sI.

Motion measurements contained in the data string are in real time, valid from the instant when the system transmits the packet start character (‘:’). Motion measurements include ASCII-coded decimal values.

Heave measurements are in cm in the range –99.99 to +99.99 metres. Positive heave is above datum.

Roll and pitch measurements are in degrees in the range –90.00° to +90.00°. Positive roll is port-side up, starboard down.

Positive pitch is bow up, stern down.

Status flag: “U”,”G”,”H”,”F” indicate a settled condition; “u”,”g”,”h”,”f” are given during the 3 minutes settling period. “U” is unaided mode,”G” is GPS aided mode, ”H” is heading aided mode, ”F” is full aided mode. For TSS 320 messages, the status field is used to indicate heave quality control: space = OK, “?” = FAIL.

The primary coordinate system:

^ x(forward)

|

o---> y(right)

z(up)

Where Zm for Heading, Ym for Pitch (Left rotation sign is +), Xm for Roll (Left rotation sign is +).

Warning! Pitch, Roll, Heading must be re-signed to Right Rotation (usually Roll need to change sign).

3.9 Read Fix

function fixs=gLogFixRead(fName)

Read fixes structure from files was created by gComLog software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

fixs – fix structure (***Table 3.10***).

File Example:

~36019588,.~36023856,.~36028544,.

Example:

```
>> Fix=gLogFixRead('d:\zzz\');plot(diff(FIX.CompTime),'.-');
```

Table 3.10 Fixs structure field's names

Field name	Field Description
CompDay	Computer day from LOG-file name;
CompTime	Computer time (second per day)

3.10 Read Cable Counter

function CC=gLogCableCounterRead(fName,Form)

Read CableCounter structure from files was created by gComLog/gComCC software.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

Form – Data Message Format: gLogCMax, gLogCMaxRaw, gLogHYTek, gComCC;

CC – Cable Counter structure (***Table 3.11***).

Format logging Examples:

C-Max (0x0D line delimiter): ~36019588,+0033m

C-Max_Raw (0x0D or 0x0D 0x0A line delimiter): ~70634035,+0059

T_count (0x0D 0x0A line delimiter): ~40618534,1:-0002m

gComCC (0x0D 0x0A line delimiter): ~36019588,0033

Example:

```
>> CC=gLogCableCounterRead('c:\temp\mag\CC\',2);
>> plot(CC.CompTime(:)-CC.CompTime(1),CC.LenCC,'.');
```

Table 3.11 Cable Counter structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name;
CompTime	Computer time from LOG-file (second per day)
CableLen	Cable length, m.

CableCounters formats descriptions are showed below.

Table 3.12 C-Max format description

Sign	4th digit	3rd digit	2nd digit	1st digit	Unit	EOL
+ or -	0..9	0..9	0..9	0..9	"m"	
+	0	0	0	8	m	0x0D

Table 3.13 C-Max Raw Data format description

Sign	4th digit	3rd digit	2nd digit	1st digit	EOL*	EOL*
+ or -	0..9	0..9	0..9	0..9	0x0D	0x0A
+	0	0	0	8	0x0D	0x0A

*0x0D or 0x0D 0x0A depends from settings

Table 3.14 T-count format description

Prefix	Prefix	Sign	4th digit	3rd digit	2nd digit	1st digit	Unit	EOL	EOL
1	:	+ or -	0..9	0..9	0..9	0..9	"m"	0x0D	0x0A
1	:	+	0	0	0	8	m	0x0D	0x0A

Table 3.15 gComCC format description

4th digit	3rd digit	2nd digit	1st digit	EOL	EOL
0..9	0..9	0..9	0..9	0x0D	0x0A
0	0	0	8	0x0D	0x0A

3.11 Read $CUSTOM message for GeoResource48 MultiTrace

function CSTM=gLogGeoSenseNavFixRead(fName,keyS,LDelim,CompTimeLocShift)

Read $CUSTOM data (Navigation-to-GeoResource48 MultiTrace seismic station) from files created by gComLog program.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$CUSTOM' or same);

LDelim – left delimiter for log-file;

CompTimeLocShift - Computer time minus Utc_Gps time (in seconds);

CSTM – reading data structure (*Table 3.16*).

File Example:

<07093294,$CUSTOM,20170825,155821.15,7102,661756.05,8147722.95,133.52,123.2

Where

< – left delimiter (symbol 1);

07093294 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$CUSTOM – data.

$CUSTOM,YYYYMMDD,HHMMSS.SS,Fix,EEEEEE.EE,NNNNNNN.NN,Heading,Depth<CR><LF>

Example:

```
>> CSTM=gLogGeoSenseNavFixRead('c:\temp\','$CUSTOM','<',10*3600);
```

Table 3.16 GeoSense Log structure fields' names

Field name	Field Description
CompDay	Computer day from LOG-file name; Computer date number of 1 corresponds to Jan-1-0000. The year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
CompTime	Computer time from LOG-file (second per day); Vector; Row-content.
GpsDay	Gps date number 1 corresponds to Jan-1-000; the year 0000 is merely a reference point and is not intended to be interpreted as a real year. Vector; Row-content.
GpsTime	GPS's second in day. Vector; Row-content.
CompTimeShift	Calculated mean difference between Computer time and Gps time in seconds, calculated for structure time-interval. Scalar; Information-content.
CompTimeDelta	Calculated difference between Computer time and Gps time, calculated for each point decremented to CompTimeShift. Vector; Row-content.
GpsE	Calculated rectangular projection N for GPS. Vector; Row-content.
GpsN	Calculated rectangular projection E for GPS. Vector; Row-content.
Fix	Fix-point number. Vector; Row-content.
HeadingTrue	Ship-body's True heading by differential_GPS/Gyrocompass/MRU. Vector; Row-content. Vector; Row-content.
DepthSea	Sea depth by echosounder. Vector; Row-content.

3.12 Read $GPAVL

function GPAVL=gLogGpAvlRead(fName,keyS,LDelim,CompTimeLocShift)

Read $GPAVL data from files created by gComLog program.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

keyS – key string ('$GPAVL' or same);

LDelim – left delimiter for log-file;

CompTimeLocShift – Computer time minus Utc_Gps time (in seconds);

GPAVL – reading data structure with fields: CompDay,CompTime,GpsDay,GpsTime,Rn,GpsLat, GpsLon, GpsHgtGeoid, Veast, Vnorth, Vup, Xecef, Yecef, Zecef, Vxecef, Vyecef, Vzecef.

Log-file (created by gComLog program) example:

~45157697,$GPAVL,R1,5561000.000,46.21880000,142.79169707,27.256,-1.657,0.539,0.809,351174.000,-3520921.763,2673329.360,4582128.127,0.866,1.423,0.957*3A

~45157748,$GPAVL,R2,5561000.000,46.22315193,142.79206287,25.175,0.230,0.005,-0.014,351174.000,-3520659.509,2673094.804,4582461.317,-0.128,-0.191,-0.007*1F

Where

~ – left delimiter (symbol 1);

45157697 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

$GPAVL... – GPAVL data.

Using functions: gNavGpsDayCalc, gNavGpsCompTimeDelta.

Example:

```
>> GPAVL=gLogGpAvlRead('c:\temp\','$GPAVL','~',10*3600);
```

GPAVL format description

$GPAVL,R1,5561000.000,46.21880000,142.79169707,27.256,-1.657,0.539,0.809,351174.000,-3520921.763,2673329.360,4582128.127,0.866,1.423,0.957*3A

$GPAVL,R2,5561000.000,46.22315193,142.79206287,25.175,0.230,0.005,-0.014,351174.000,-3520659.509,2673094.804,4582461.317,-0.128,-0.191,-0.007*1F

R1 – Remote number (R1, R2…).

5561000.000 – Utc UTC milliseconds of the day.

46.21880000 – Latitude (degrees).

142.79169707 – Longitude (degrees).

27.256 – Height (meters).

-1.657 – Velocity East.

0.539 – Velocity North.

0.809 – Velocity Up.

351174.000 – Gpstime seconds of the week.

-3520921.763 – X ECEF coordinate.

2673329.360 – Y ECEF coordinate.

4582128.127 – Z ECEF coordinate.

0.866 – X ECEF velocity.

1.423 – Y ECEF velocity.

0.957 – Z ECEF velocity.

*3A – Checksum data, always begins with *.

3.13 Read TP2c

function out=gLogTP2cRead(fName,LDelim)

Read TP2 compatible output data (USBL) from gLog.

Parameters:

fName – name of file with USBL data;

LDelim – left delimiter for log-file;

out – output structure, which includes fields: CompDay, CompTime, UsblTime, TargNum, SHead, TBearing, TDist, XOffset, YOffset, ZOffset, Reserv1, Reserv2.

Example:

```
>> out=gLogTP2c('c:\temp\whghghhe\20190901_140000.mur','~');
```

Log-file (created by gComLog program) example:

```
~50400477,12 13:59:55   0 201.1   184.0    -65.3   -169.4    30.2      0.0
~50401997,12 13:59:57   0 201.4   187.1    -66.5   -169.4    43.1      0.0
```

where

~ – left delimiter (symbol 1);

50400477 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

12 13:59:... – USBL data.

TP2 compatible output format description

```
~50400477,12 13:59:55   0 201.1   184.0    -65.3   -169.4    30.2      0.0
~50401997,12 13:59:57   0 201.4   187.1    -66.5   -169.4    43.1      0.0
```

Target number – Character 1

Time – Character 3-10;

Ship heading – Character 12-14;

Target Bearing – Character 16-20;

Slant Range – Character 22-28;

X Offset – Character 30-37;

Y Offset – Character 39-46;

Z Offset – Character 48-54;

Reserved – 56-63;

Reserved – Character 65-66 (blank);

CR – 67;

LF – 68;

3.14 Read SeaSpy magnetometer data

function out=gLogMagyReadSeaSpy(fName,Delim,EqId)

Read SeaSpy data (Standard Format) from gLog

Parameters:

fName – path to folder/file with SeaSpy Magnetometer data;

Delim – [left_delimiter_symbol right_delimiter_symbol] for gLog record;

EqId – serial numbers [mag, alt, depth], NaN is "no devise";

out – output structure, which includes fields: CompDay, CompTime, MagDay, MagTime, MagAbsT, MagPrecSignal, MeasurTime, Depth, Altitude, Leak, SignalQuality, FlagsWPMG.

Example:

```
>> out=gLogMagyReadSeaSpyGrad('c:\temp\whghghhe\TEST.Nav.txt','~',1);
```

Log-file (created by gComLog program) example:

~42042805,*05.001/10:27:11.0 F:050945.927 S:178 D:+010.3m A:001.02m L0 0349ms Q:43 G

where

~ – left delimiter (symbol 1);

42042805 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

*19.212/... – SeaSpy magnetometer data.

SeaSpy magnetometer format description

*05.001/10:27:11.0 F:050945.927 S:178 D:+010.3m A:001.02m L0 0349ms Q:43 G

*05.001/10:27:12.0 F:050948.573 S:177 D:+010.3m A:001.02m L0 0372ms Q:54 G

*YY.JJJ/HH:MM:SS.S F:FFFFFF.FFF S:SSS D:+DDD.Dm A:AAA.A L:L TTTms Q:QQ !!!! CR LF

Y- Year; JJJ- Julian day; HH- Hour; MM- Minute; SS.S- Second;

FFFFFF.FFF- Magnetic field (nT);

SSS – Signal strength of reading. This is a raw number generated by the magnetometer that gives (in part) a good indication of the quality of the final total field measurement. Anything over 80 is considered an acceptable signal, and anything over 130 is considered excellent (SSS should be between 130 and 200 for good quality readings).

DDD.D – Towfish Depth. The value shown is in meters. The depth sensor can be calibrated using the P and p commands.

AAA.A – Towfish Altitude. The value shown is in meters. If no altimeter is installed, this field will not be present. If an altimeter is installed, but it cannot obtain a ‘lock’ on the seafloor (for example if it is too far away) this value will be 000.0m.

L – Leak sensor output, 0-9. 0 indicates no leak, and 9 indicates that a leak is present.

TTT – Measurement time. Ideally, this should be the magnetometer’s cycling time minus 35ms, with a maximum of 965ms (965 when F is greater than 42000 and 465 when F is less than 42000). If you see a G message, indicating that measurement was prematurely terminated due to a high gradient condition, this value will tell you how severe the gradient is.

QQ – Signal quality. This is a two-digit number between 00 to 99. The left digit is a good indication of signal strength, and the right digit indicates how much information was available for measurement.

!!!! – Warning Messages "WPMG" or " ":

W – Weak signal. This message is displayed if the signal strength for the reading is below a threshold value;

P – Poor reading. This message is displayed if the signal is sampled for too short a time period, for whatever the reason. Expect this message under conditions of extremely high magnetic gradient.

M – Instrument Mistuned. The magnetometer may decide to display this message under extremely poor signal conditions, which is characteristic of poor tuning settings. When this message occurs, the

instrument will attempt to retune by executing an initialize tuning procedure, if the auto-tuning feature is enabled.

G – Gradient condition. In high magnetic gradients, the signal produced by the sensor decays more quickly. This message occurs if the measurement time was prematurely terminated due to a quickly decaying signal. The strength of the gradient can be estimated by observing the measurement time. Note that sensitivity will decrease as the measurement time decreases.

CR LF – Carriage Return (ASCII code 13), Line Feed (ASCII code 10).

3.15 Read SeaSpy gradiometer data

function out=gLogMagyReadSeaSpyGrad(fName,Delim,EqId)

Read SeaSpy Gradiometer data from gLog

Parameters:

fName – path to folder/file with SeaSpy Gradiometer data;

Delim – [left_delimiter_symbol right_delimiter_symbol] for gLog record;

EqId – serial numbers [mag1, alt1, depth1; mag2, alt2, depth2], NaN is "no devise";

out – output structure, which includes fields: CompDay, CompTime, MagDay, MagTime, MagAbsT, MagPrecSignal, MeasurTime, Depth, Altitude, FlagsGWP.

Example:

```
>> out=gLogMagyReadSeaSpyGrad('c:\temp\whghghhe\TEST.Nav.txt','~',3);
```

Log-file (created by gComLog program) example:

~43322102,*19.212/04:12:40.0 F[017345.893 074 0013 +0001.7 000.00 G_P] R[021818.742 080 0015 -0027.0 000.00 G_P] -04472.849

where

~ – left delimiter (symbol 1);

43322102 – second per day./1000 (symbols 2-9);

, – right delimiter (symbol 10);

*19.212/... – SeaSpy Gradiometer data.

SeaSpy gradiometer output format description

*19.212/04:12:40.0 F[017345.893 074 0013 +0001.7 000.00 G_P] R[021818.742 080 0015 -0027.0 000.00 G_P] -04472.849

*19.212/04:12:41.5 F[018227.327 089 0213 +0001.7 000.00 G__] R[021852.546 047 0005 -0027.0 000.00 G__] -03625.219

*YY.JJJ/HH:MM:SS.S F[FFFFFF.FFF SSS TTTT DDDD.D AAA.AA !!!] R[FFFFFF.FFF SSS TTTT DDDD.D AAA.AA !!!] –GRADIEN CR LF

Y – Year; JJJ- Julian day; HH- Hour; MM- Minute; SS.S- Second;

FFFFFF.FFF- Magnetic field (nT);

SSS – Signal strength of reading. This is a raw number generated by the magnetometer that gives (in part) a good indication of the quality of the final total field measurement. Anything over 80 is considered

an acceptable signal, and anything over 130 is considered excellent (SSS should be between 130 and 200 for good quality readings).

TTTT – Measurement time. Ideally, this should be the magnetometer's cycling time minus 35ms, with a maximum of 965ms (965 when F is greater than 42000 and 465 when F is less than 42000). If you see a G message, indicating that measurement was prematurely terminated due to a high gradient condition, this value will tell you how severe the gradient is.

DDDD.D – Towfish Depth. The value shown is in meters. The depth sensor can be calibrated using the P and p commands.

AAA.AA – Towfish Altitude. The value shown is in meters. If no altimeter is installed, this field will not be present. If an altimeter is installed, but it cannot obtain a 'lock' on the seafloor (for example if it is too far away) this value will be 000.0m.

!!! – Warning Messages "GWP" or "___", formed as bits-flag for each sensor:

G – Gradient condition. In high magnetic gradients, the signal produced by the sensor decays more quickly. This message occurs if the measurement time was prematurely terminated due to a quickly decaying signal. The strength of the gradient can be estimated by observing the measurement time. Note that sensitivity will decrease as the measurement time decreases.

W – Weak signal. This message is displayed if the signal strength for the reading is below a threshold value;

P – Poor reading. This message is displayed if the signal is sampled for too short a time period, for whatever the reason. Expect this message under conditions of extremely high magnetic gradient.

M – Used for compatibility with "gLogMagyReadSeaSpy" Instrument Mistuned, always set to 0 for "gLogMagyReadSeaSpyGrad" (the magnetometer may decide to display this message under extremely poor signal conditions, which is characteristic of poor tuning settings. When this message occurs, the instrument will attempt to retune by executing an initialize tuning procedure, if the auto-tuning feature is enabled).

F[...] and R[...] – Front and Rear towfishes;

–GRADIEN – Front field minus Rear field and devide dy distance;

CR LF – Carriage Return (ASCII code 13), Line Feed (ASCII code 10).

4. gLog Synchronization, check and Database

4.1 Difference between CompTime and GpsTime

function [CompTimeDelta,CompTimeShift]=

gLogGpsCompTimeDelta(CompDay,CompTime,GpsDay,GpsTime)

Calculate difference between [GpsDay,GpsTime] and [ComputerDay,ComputerTime] in seconds.

Parameters:

GpsDay – a serial Gps date number of 1 corresponds to Jan-1-0000 or used GpsS=gNGpsDayCalc(GpsS);

GpsTime – GpsTime (second in day);

CompDay – a serial Computer date number of 1 corresponds to Jan-1-0000;

CompTime – Computer time (second in day);

CompTimeShift – mean difference -- computer time minus Gps time (in seconds);

CompTimeDelta – vector, difference between Computer time and Gps time for each point decrement to CompTimeShift.

Example (***Figure 4.1***):

```
>> [CTDelta,CTShift]=gLogGpsCompTimeDelta(Z.CompDay,Z.CompTime,Z.GpsDay,Z.GpsTime)
```

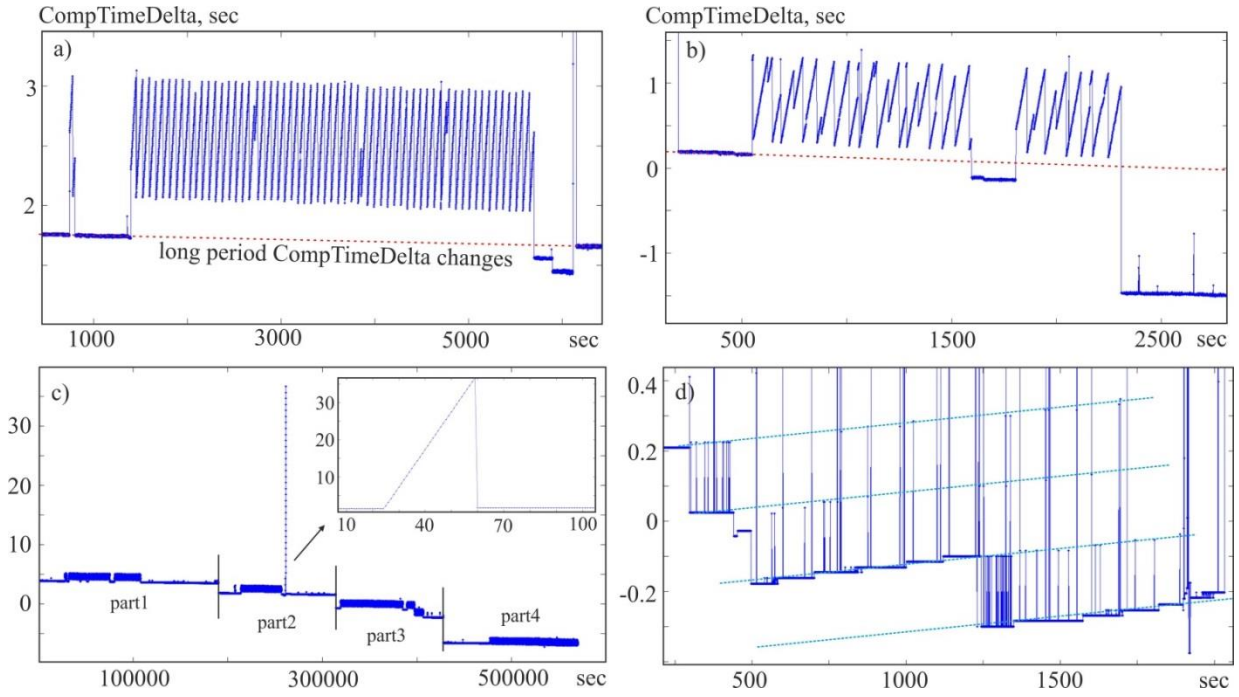


Figure 4.1 Example of difference between [GpsDay,GpsTime] and [ComputerDay,ComputerTime] in seconds (based on Geometrics MagLog files), GGA string come from Qinsy through serial port

a) GPS time has small “late step”; when time difference set about one second, the GPS time has 1-second jump;
b) the same a), but jump period more arbitrary;
c) GPS time stopped about 40 seconds; the time period between record parts is the several days;
d) difference between [GpsDay,GpsTime] and [ComputerDay,ComputerTime] include specific law error.

4.2 Calculate GpsDay

function GpsDay=gLogGpsDayCalc(CompDay,CompTime,GpsTime,CompTimeLocShift)

Calculate GpsDay using CompDay, CompTime, GpsTime.

Parameters:

GpsTime – GpsTime (second in day);

CompDay – a serial Computer date number of 1 corresponds to Jan-1-0000;

CompTime – Computer time (second in day);

CompTimeLocShift – Computer time minus Gps time (in seconds);

GpsDay – Estimated serial Gps date number of 1 corresponds to Jan-1-0000.

Example:

```
>> GpsDay=gLogGpsDayCalc(Z.CompDay,Z.CompTime,Z.GpsTime,10);
```

GpsDay estimation method:

$$\mathrm{GpsDay}\cdot 86400+\mathrm{GpsSec}=\mathrm{CompDay}\cdot 86400+\mathrm{CompSec}-\mathrm{LocHour}\cdot 3600+\mathrm{Err};$$

$$\mathrm{GpsDay}=\mathrm{round}\left(\frac{\mathrm{CompDay}\cdot 86400+\mathrm{CompSec}-\mathrm{LocHour}\cdot 3600-\mathrm{GpsSec}}{86400}+\frac{\mathrm{Err}}{86400}\right);$$

$$\mathrm{GpsDay}-\Delta=\mathrm{round}\left(\frac{(\mathrm{CompDay}-\Delta)\cdot 86400+\mathrm{CompSec}-\mathrm{LocHour}\cdot 3600-\mathrm{GpsSec}}{86400}+\frac{\mathrm{Err}}{86400}\right);$$

$$\Delta=\mathrm{round}\big(\mathrm{mean}(\mathrm{CompDay})\big);$$

$\mathrm{Err}<0.5$ day.

4.3 Create FilesList for time section

function fNameList=gLogFilesFind(rootD,TimeS,TimeF)

Create FilesList using start and final date/time (for files created by gComLog software).

Parameters:

rootD – root folder with files were created by gComLog program; file names format: YYYYMMDD_HHMMSS.extension (for example 20170909_040000.imu);

TimeS – start time in format [YYYY MM DD HH MM SS] (for example [2017 05 12 13 40 00]);

TimeE – final time in format [YYYY MM DD HH MM SS];

fNameList – files list; the time includes interval with TimeS and TimeF.

Example:

```
>> fNam=gLogFilesFind('d:\gLogZ\Imu\',[2017 09 10 00 00 00],[2017 09 10 05 12 15]);
>> Tss1=gLogTss1Read(fNam);
```

4.4 Find “bad” strings for files logged

function gLogFileFormatFix(fName,Form,LenFl)

Read strings from position-formatted files and check it in accordance with Form-mask; copy "bad" files content to new files with '_good' and '_bad' postfixes.

Parameters:

fName – reading file name or files name or folder name with files (last name's symbol must be '\');

Form – position-formated file "mask", includes ascii code limits for each symbol, there can be:

- vector with two rows of mask, contained ascii code limits for strings' symbols;
- predefined mask name;
- cells, includes "vectors with two rows"; it is the method "multiple mask" realized.

The last rows Form element is StringTerminate symbol code.

if LenFl==0, then string can include several "not described in mask" chars before StringTerminate symbol code.

Delim – [first_delimiter second_delimiter] for data recorded by gLog;

LenFl – flag==1 for condition "string length is equal Form length";

varargin{1} – the equipment components serial numbers, it is used for Form-string choice as an additional equipment description (see 'gLogSeaSpyGrad' below);

flZ2 – "bad" files quantity;

For "bad" files create two additional files with follow content: if string is "good", then it coped to [rootD fNameN '_good'], else it coped to [rootD fNameN '_bad'].

Example:

```
>> gLogFileFormatFix('c:\05_Prog\123.txt','MagLogMagy',['~' ','],1);
>> gLogFileFormatFix('c:\05_Prog\TSS\',Form,['~' ','],1);
```

Form Example1 (Geometrics Mag recorded by MagLog):

$ 55865.675,0854,3867,0808 09/07/14 22:21:04.031

Form(1,:)=[36 32 48 48 48 48 48 46 48 48 48 44 48 48 48 48 44 48 48 48 48 44 48 48 48 48 32 32 48 48 47 48 48 47 48 48 32 48 48 58 48 48 58 48 48 46 48 48 48 13 10];

Form(2,:)=[36 32 57 57 57 57 57 46 57 57 57 44 57 57 57 57 44 57 57 57 57 44 57 57 57 57 32 32 57 57 47 57 57 47 57 57 32 57 57 58 57 57 58 57 57 46 57 57 57 13 10];

Form Example2 (<21600444,:0BFC79 0029U-0054 0331 there is TSS1 PACKET :XXAAAA_MHHHHQMRRRR_MPPPPZZ recorded by gLog)

Form(1,:)=[60 48 48 48 48 48 48 48 48 44 58 48 48 48 48 48 48 32 32 48 48 48 48 32 32 48 48 48 48 32 32 48 48 48 48 13 10];

Form(2,:)=[60 57 57 57 57 57 57 57 57 44 58 70 70 70 70 70 70 32 45 57 57 57 57 122 45 57 57 57 57 32 45 57 57 57 57 13 10];

4.5 Checksum calculation for NMEA message

function q=gLogCheckSum(st)

Checksum calculation for NMEA message.

Parameters:

st – string for checksum calculation;

q – checksum in Hex.

Not used $,!,* for: $GPGLL,5057.970,N,00146.110,E,142451,A*27<CR><LF> used

'GPGLL,5057.970,N,00146.110,E,142451,A'

Example:

```
>> q=gLogCheckSum(['$GPGLL,5057.970,N,00146.110,E,142451,A*27' char([13 10])]);
```

Citation

1) NMEA 0183 Standard For Interfacing Marine Electronic Devices Version 3.01 January 1, 2002
2) Teledyne TSS Ltd // DPN 060101 Issue 12.6
3) SeaSPY Operation Manual Revision 5.02 // Marine Magnetics Corp
4) LinkQuest Inc. TrackLink 1500 USBL Tracking System // User's Guide Version 1500.7.2
5) Multi-Trace Data Acquisition software // User manual rev. 1.4, Created on: 2014-05-11, Last modified: 2016-05-04 By Sergio Monteleone // Copyright GEO Marine Survey Systems.