GeoCOM

Reference Manual

TPS1100 - Version 1.05

1 GEOCOM

1.1 INTRODUCTION

TPS1100 series Theodolites are modern geodetic measurement instruments. Most of the main tasks can be fulfilled with these instruments implicitly by their integrated applications. Now, to fulfil a broader spectrum of tasks and applications an interface to the TPS1100 series sensor functions has been defined and will be published with this document.

With this interface it will be possible to write client applications based on MS-Windows and/or for any other platform which supports ASCII based communications.

1.2 TPS1100 SYSTEM SOFTWARE

The TPS1100 system software organises and controls the interplay of several sensor elements. Furthermore, it builds up a frame for applications, which can be executed on the TPS1100 Theodolite.

This document concentrates on the main interface to the sensor elements of the TPS1100 Theodolite. This main interface can be used to implement solutions for special customer problems if the already existing solution does not provide the needed functionality or just to enhance it.

1.2.1 Organisation of Subsystems

The TPS1100 system software is built around the sensor elements, which are parts and/or optional add-ons of the TPS1100 Theodolite instrument. It provides a set of functions to access sensors and calculated values. These functions are organised as subsystems. We will keep this segmentation in this document.

These functions can be grouped in the following sections:

AUS The subsystem ‘Alt User’ mainly contains functions behind the “FNC” button.

AUT Automatisation; a module which provides functions like the control of the Automatic Target Recognition, Change Face function or Positioning functions.

BAP Basic Applications; some functions which can easily be used to get measuring data.

BMM Basic Man Machine; functions which controls some basic input/output functionality, e.g. set beep alarm, etc.

COMF Communication; a module which handles the basic communication parameters. Most of these functions relate to both client and server side.

COM Communication; functions to access some aspects of TPS1100 control which are close to communication. These functions relate either to the client side or to the server side.

CSV Central Services; this module provides functions to get or set central/basic information about the TPS1100 instrument.

CTL Control task; this module contains functions of the system control task.

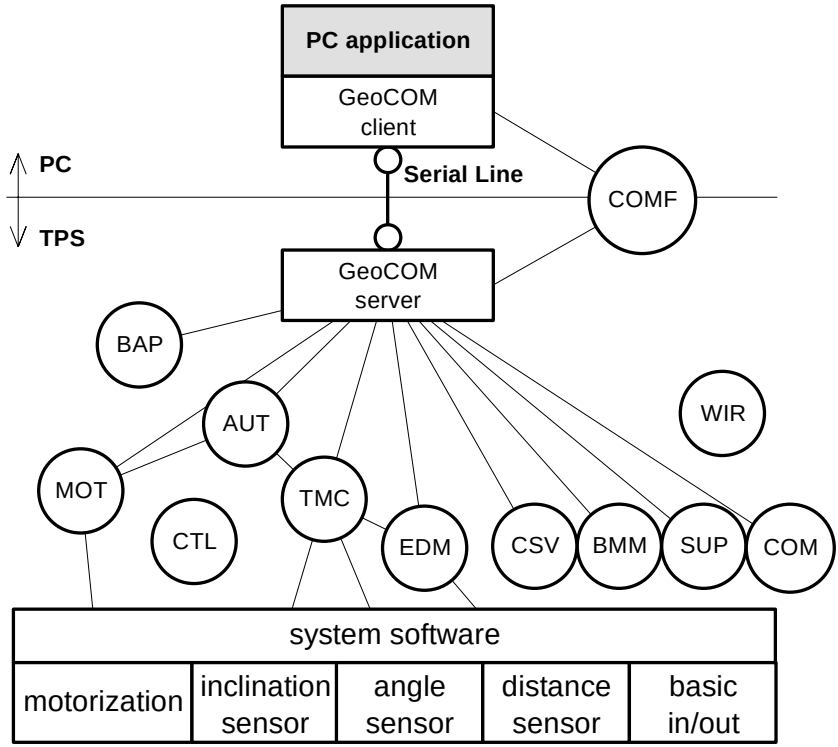
EDM Electronic Distance Meter; the module which measures distances.

MOT Motorization; the part which can be used to control the movement and the speed of movements of the instrument.

SUP Supervisor; functions to control some of the general values of the TPS1100 instrument, e.g. set the lower limit temperature.

TMC Theodolite Measurement and Calculation; the core module for getting measurement data.

WIR WI Registration; this module contains function for GSI recording.



Picture (1) - Overview Client/Server Application

1.3 PRINCIPLES OF GEOCOM OPERATION

Communication takes place between two participants - a client and a server. The medium of communication is a serial communication line. Refer to Appendix B for further information about settings and needed hardware.

The idea of GeoCOM is based on SUN Microsystems' Remote Procedure Call (RPC) protocol.

On the low level of implementation, each procedure, which is executable on the remote instrument, is assigned a remote procedure call identification number. This number is used internally to associate a specific request, including the implicit parameters, to a procedure on the remote device. On this level, GeoCOM provides

an ASCII interface, which can be used to implement applications on platforms, which do not support MS-Windows.

On the high level, GeoCOM provides normal function call interfaces for C/C++ and MS-VBA to these remote functions. These interfaces enable a programmer to implement an application as if it would be executed directly on the TPS1100 instrument.

Note: Further on we will refer to a remotely executable system function as a *RPC*.

The TPS1100 instrument system software uses a multitasking operating system. Nevertheless, only one request can be executed at once. This means in respect of calling RPC's GeoCOM works synchronously only.

On the low level interface the server buffers subsequent requests if current request(s) has not been finished so far. If the queue is full then subsequent requests will be lost.

Instead on the high level interface a function call will not return until it has been completely finished.

2 GENERAL CONCEPTS OF USING GEOCOM

Here we will describe several aspects of using GeoCOM. One of them is how to execute a function at a TPS1100 instrument.

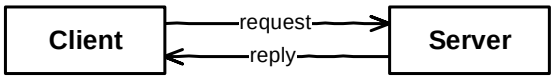
The current implementation of GeoCOM supports two (three) kinds of usage. We can distinguish between a rather rudimentary ASCII protocol and a high level function call interface.

The former - ASCII protocol - is made up of requests and replies. Using GeoCOM in this way means that an application assembles a request, sends it over the serial line to the listening TPS1100 instrument, wait for the answer and decode the received reply.

The latter uses normal function calls either in C/C++ or in VBA. For explanation purposes we will split it into two categories because the two supported programming environments differ in relation to their type systems. Using GeoCOM in this way means calling a function. Any necessary communication will be handled by GeoCOM implicitly.

2.1 GENERAL CONCEPT OF OPERATION

Fundamentally, GeoCOM is implemented as a point to point communication system. The two communication participants are known as the client (external device) and the server (TPS1100 instrument). One communication unit consists of a request and a corresponding reply. Hence, one communication takes place when the client sends a request to the server and the server sends a reply back to the client.



Picture 2-1: Basic communication

GeoCOM is implemented as synchronous communication. A request/reply pair may no be interrupted by another request/reply. Instead, a communication unit must be completed successfully before a new communication unit may be initiated.

Although the ASCII protocol allows sending the next request before the corresponding reply has been received, it is not recommended to do that. Of course, subsequent request will be buffered when the previous request has not been finished so far. But if the buffer content reaches its limit in size then data may be lost.

Note: In the current implementation, only one communication channel per session will be supported. Hence, only one instrument can be connected at a time. Nevertheless, the nature of the ASCII protocol makes it possible to connect and communicate to more than one instrument at a time.

2.2 ASCII PROTOCOL

In sequence we will define the syntax first and then give some information about how to use the ASCII protocol to call a function on the TPS1100 instrument.

The ASCII protocol is a line protocol, hence it uses a line terminator to distinguish between different requests (replies). One request must be terminated by one terminator.

2.2.1 ASCII Protocol Syntax

Syntax of an ASCII request:

```
[<LF>]%R1Q,<RPC>[,<TrId>]:[<P0>][,<P1>,...]<Term>
```

Optional items are in brackets []. The angled-brackets <> surround names or descriptions. These names have variable values depending on their types and meanings. The angled-brackets themselves are not part of the transferred text. Characters not surrounded by brackets are literal text and are part of the GeoCOM protocol.

<LF>	An initial line feed clears the receiver buffer.
%R1Q	GeoCOM request type 1.
<RPC>	Remote Procedure Call identification number in between 0 to 65535.

<TrId>	Optional transaction ID: normally incremented from 1 to 7. Same value in reply.
:	Separator between protocol header and following parameters.
<P0>,<P1>,...	Parameter 0, Parameter 1, ...
<Term>	Terminator string (default CR/LF, use COM_SetTerminator to change the terminator). As a common shortcut '^m' will be used in examples.

Example:

The following example uses the RPC TMC_SetPrismCorr to set the prism constant on 34.4mm ('^m' denotes the terminator):

```
%R1Q,2024:34.4^m
```

Note: Additional characters at the beginning of a request, between parameters or at the end are not allowed. They might lead to errors during interpretation.

Syntax of an ASCII reply:

```
%R1P,<GRC>[,<TrId>]:<RC>[,<P0>,<P1>, ...]<Term>
```

Optional items are in brackets []. The angled-brackets <> surround names or descriptions. These names have variable values as described in the types they have. The angled-brackets themselves are not a part of the communication text. Characters not surrounded by angled-brackets are literal text and are part of the GeoCOM protocol.

%R1P	GeoCOM reply type 1.
<GRC>	GeoCOM return code. This value denotes the success of the communication. 0 = RC_OK means the communication was successful (see table 'GeoCOM return codes' in the appendix for further information).
<TrId>	Transaction ID - identical to that of the request. If the request had no Transaction ID then it will be 0.
:	Separator between protocol header and following parameters.

<RC>	Return code from the called RPC and denotes the successful completion if it is set to 0 (see table 'RPC return codes' in the appendix for further information).
<P0>,<P1>,...	Parameter 0, Parameter 1, … These parameters will be valid only if <GRC> is equal to 0 (RC_OK).
<Term>	Terminator string (default CR/LF, use COM_SetTerminator to change the terminator).

Example:

The following example shows the reply to the RPC 5008 - CSV_GetDateTime.

```
%R1P,0,0:0,1996,'07','19','10','13','2f'^m
      ¦ ¦ ¦ ----------------------------
      ¦ ¦ ¦       ¦      The values for month, day, hour,
      ¦ ¦ ¦       +---   minute and second are replied in the byte-
      ¦ ¦ ¦              format (see table communication parameter
      ¦ ¦ ¦              for further information)
      ¦ ¦ +------  Return code from the RPC: 0 means  no error
      ¦ ¦            (see RPC return codes for further  information)
      ¦ +-----  The Transaction ID of the request. If there was  no ID
      ¦          the value returned is 0.
      +----  Return code from GeoCOM: 0 means no error  (see
             GeoCOM return codes for further information)
```

2.3 FUNCTION CALL PROTOCOL - C/C++

The implementation of GeoCOM for C/C++ conforms to normal function calls. GeoCOM itself handles all necessary communication. No intervention of the programmer in respect to the communication is necessary with one exception. If the GeoCOM reports a communication error the programmer has to make sure that either the problem will be solved - by calling GeoCOM support functions - or no further RPC's will be called - by terminating the running task.

Nevertheless, the programmer has to initialise GeoCOM and set up the port's settings to make sure that communication can take place. Moreover the user has to make sure that the TPS1100 instrument is well connected.

Example:

An example code fragment for using TMC_GetSimpleMea could be the following. We do not take care of the necessary initialisation and set up of GeoCOM here.

Please refer to chapter 3.2.3 Basic GeoCOM Application Frame for C/C++ for this information.

```
RC_TYPE       RetCode;
TMC_HZ_V_ANG  Angles;
double        dSlopeDist;
RetCode = TMC_GetSimpleMea( 1000, Angles,
                                  dSlopeDist,
                                  TMC_AUTO_INC );
if (RetCode == RC_OK)
{
  // do something - use values
}
else
{
  // handle error
}
```

2.4 FUNCTION CALL PROTOCOL - VBA

Here almost all is valid for VBA as for C/C++. Please refer to Chapter 2.3. The only difference between VBA and C/C++ is that VBA has a different type system. Hence, the defined data types differ slightly in their definition. Furthermore, because of implementation reasons the RPC names must have an additional prefix, which is “VB_” for the current implementation of GeoCOM.

Example:

We take the same example as in Chapter 2.3.

```
Dim RetCode     As Integer
Dim Angles      As TMC_HZ_V_ANG
Dim dSlopeDist  As Double
RetCode = VB_TMC_GetSimpleMea( 1000, Angles,
                               dSlopeDist,
                               TMC_AUTO_INC )
If RetCode = RC_OK Then
  ' do something - use values
Else
  ' handle error
  End If
```

3 FUNDAMENTALS OF PROGRAMMING GEOCOM

We will describe how programs can be written using the different protocols. Certainly, the type system, where the main differences lie between the protocols, will be described in more detail.

3.1 ASCII PROTOCOL PROGRAMMING

Implementing an application, which uses the ASCII protocol, is based on simple data transfers using a serial line. The programmer is responsible to set up the serial line parameters of the client such that they correspond to the settings of the TPS1100 instrument. Then Remote calls are done by just sending the valid encoded requests and receiving and decoding the replies of them.

For debugging purposes, it might be helpful to use a so-called Y-cable, which enables you to observe the communication on the serial line using either a terminal or a terminal emulator. For further details see Appendix B-2 Debugging Utility.

Note: If the settings of the active COM port will be set by any software part and if the server is online, then it is strongly recommended to use a leading <LF> to clear the receiver buffer at the server side. This will reduce unnecessary error messages of the next RPC.

3.1.1 Data Types in ASCII Protocol

Each parameter of a RPC has its own associated data type with it. There are varieties of different data types, which have been defined for the set of published functions. The ASCII protocol supports simple data types only. All data types, which are different from the base, types in name and aggregated data types are converted and reduced to there base types. Conversion means to serialise the aggregated data into a comma-separated list of its elements. Therefore, the programmer has the responsibility to interpret the values depending on the associated data type.

The supported base types and their value range are defined below:

Format Type	Valid range	Len	Valid input representations	Typical output representations
`boolean`	0 = false 1 = true	1	0,1	0,1
`byte`	0...255	2 (4)	‘00’,’FF’,’ff’,’7a’, ’A7’	‘00’,’FF’,’ff’, ’7a’,’A7’
`string`	-	<512	”abc\x0d\x0a”	”abc\x0d\x0a”
`double`	±2.225E-308... ±1.797E+308	17+3	1, 1.0, 1.0e4, -0.1e-07, -2	-0.1234567+e67
`long`	$(-2^{31})...(2^{31}-1)$	11	0x7FFFFFFF, -54321	15, -154836, 900000
`short`	-32768...32767	6	0, -1, -32700, 45, 56, 0x45e, 0X3AA	0, -1, -32700, 45, 56
`unsigned long`	$0...(2^{32}-1)$	10	0xFFFFFFFF	0, 1, 3400065, 95735
`unsigned short`	0...65535	5	0, 1, 34000, 65, 65535, 0x3a, 0x00, 0xFFFF	0, 1, 34000, 65, 65535

Table 3-1: Communication Parameter Types

Note: Bytes are always represented in two-character hexadecimal notation. Hexadecimal notation can use upper- or lower-case representation: 0..9 + [a .. f | A .. F].
Characters sent within a string which do not fall within the ASCII character range 0x20 to 0x7E (32 to 126 decimal) are sent using an adapted byte notation - e.g. ”\x9A”, where \x (or \X) introduces a byte value in hexadecimal notation.
Types of integer (short, unsigned short, long, unsigned long) can also be represented in hexadecimal notation, introduced by 0x or 0X.

The following rules are for generating/interpreting values with a type different from the base types and aggregated data types:

Numerical and string data type

The numerical data types correspond to the C-parameters in value, range and precision as close as possible. If no identical data type is available then the next best one will be taken. Character and string will be replaced by the `string` data type.

Enumerations

If the corresponding C-parameter is an enumeration data type, then the enumeration value of the ASCII parameter is equal to the implicit value of the declaration of the C-data type. For clarification, we will give always the name and the associated value in the description of an enumeration data type.

Structures

Structure data types will be converted into a comma separated list of elements. One element's representation conforms to the data type representation of its base type. If an element itself is a structure then depth first conversion will take place. If this rule does not apply then the types and their ASCII parameters are described explicitly.

Arrays

An array will be converted into a comma-separated list of elements. One element's representation conforms to the data type representation of its base type.

Example for Enumeration Data Types and Structures

The following example gives a typical data type declaration and the corresponding procedure declaration used in this manual for `TMC_GetSimpleMea` from the subsystem Theodolite Measurement and Calculation:

Constants and Types

```
typedef long SYSTIME;

struct TMC_HZ_V_ANG
{
  double dHz;
  double dV;
}

enum TMC_INCLINE_PRG
{
  TMC_MEA_INC,              // encoded as 0
  TMC_AUTO_INC,             //            1
  TMC_PLANE_INC             //            2
}
```

C-Declaration

```
TMC_GetSimpleMea(SYSTIME          WaitTime,
                 TMC_HZ_V_ANG     &OnlyAngle,
                 double           &dSlopeDistance,
                 TMC_INCLINE_PRG  Mode)
```

ASCII-Request

`%R1Q,2108:`*WaitTime[long],Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC,Hz[double],V[double],dSlopeDistance[double]*

Please, notice that the RPC has two input and two output parameters. Anytime a request must encode and send input and in/out parameters only and a reply must encode and send in/out and output parameters only!

Note: Unnecessary parameters must not be sent.
Although the enclosed header file `com_pub.hpp` denotes default values for certain function parameters they will not be supported. Hence, they have to be sent.

The ASCII Request to call this RPC with the value for `WaitTime` = 1000 and the inclination measure mode `TMC_AUTO_INC` has the following form (note that the value 1 is used for the `Mode` parameter because the counting of enumeration data types start at 0):

```
%R1Q,2108:1000,1^m
```

A possible reply can be as follows:

```
%R1P,0,0:0,0.9973260431694,1.613443448007,1.3581^m
```

Where the second and third value after the colon corresponds to the `dHz` and `dV` parts of the structure `TMC_HZ_V_ANG` and the fourth value corresponds to the variable `dSlopeDistance`. (Note that the first value after the ':' is not a parameter but the return code value of the RPC).

3.1.2 ASCII Protocol Program Example

For getting a feeling of how requests and replies are build up and work see also the provided `geocom.trm` file in the samples directory. Please refer to Appendix C-1 Settings for Terminal Emulator for further information.

3.1.3 Modes of Operation Concerning Communication

Section 3.5 - TPS1100 Instrument Modes of Operation - explains the different modes of operation of GeoCOM concerning communication. Similar to that the following is valid for the ASCII protocol.

Since the client has to remind which mode is active, no support can be given from the TPS1100 instrument. The only way to distinguish between modes is to remind the actions an application has initiated and their resulting replies. So far no other possibility exists to determine the current mode.

To switch on the instrument a single character is sufficient. It is recommended to ignore the subsequent reply (one or two lines). Please note, that if the autoexec mechanism ([menu] CONF - [item] Autoexec-application) is enabled then the instrument will not switch into Remote mode, but will start the autoexec application instead.

When turning into local mode the TPS1100 instrument sends the “sign-on” message:

“%N1,0,255,,0%T0,0,0,:%R1P,0,0:0”.

If the “sign-off” message is enabled (see `COM_EnableSignOff`) then the following message will be sent if the instrument goes into sleep mode:

“%N1,0,255,,0%T0,0,0,:%R1P,1,0:0,1”,

and the following message will be sent if the instrument shuts down:

“%N1,0,255,,0%T0,0,0,:%R1P,1,0:0,0”.

Please notice that these two messages are different in the last character.

3.2 C/C++ - PROGRAMMING

Programming in C/C++ is based on the well-known DLL concept, defined by Microsoft Corp. To compile a project successfully first you have to include the file `com_pub.hpp`, which defines all necessary constants, data types and function prototypes. Second `gcom100.lib` has to be included in the project, which enables the linker to resolve the DLL exported functions. To operate successfully the `gcom100.dll` file must be accessible for the operating system, hence it must be located in a directory which the operating system looks up for the requested DLL file.

Project Options	**GCOM100.lib**
Structure byte-alignment	4 bytes
Memory model	N/A
Special #defines (if not using MFC)	STRICT

3.2.1 Data Types in C/C++

Since the main programming language of implementation of TPS1100 instruments Firmware is C/C++ all data types are initially defined in C/C++. Therefore, no conversion of values or data types is necessary.

3.2.2 Basic GeoCOM Application Frame for C/C++

A C/C++ GeoCOM application consists at least of the following parts:

- Initialise GeoCOM
- Open a connection to the server
- One or more GeoCOM RPC's
- Close the active connection to the server
- Finalise GeoCOM

A sample implementation of above points could be:

```
// include standard system headers
#include "com_pub.hpp"
// include application headers
#define RETRIES_1 1

RC_TYPE   RetCode;
BOOLE     bOpenAndRunning = FALSE;

// initialize GeoCOM
RetCode = COM_Init();
if (RetCode == RC_OK)
{
  // open a connection to the TPS1000 instrument
  RetCode = COM_OpenConnection (  COM_1, COM_BAUD_19200,
                                  RETRIES_1);
  if (RetCode == RC_OK)
  {
```

```
      bOpenAndRunning = TRUE;
    }
  }

  // optionally set up other comm. parameters here

  if (RetCode == RC_OK)
  {
    // -- functionality of the application --
    // here we just test if communication is up
    RetCode = COM_NullProc();
    if (RetCode != RC_OK)
    {
      // handle error
    }
  }

  // close channel
  if (bOpenAndRunning)
  {
    RetCode = COM_CloseConnection ();
    if (RetCode != RC_OK)
    {
      // handle error
    }
  }

  // anytime finalize and reset GeoCOM
  RetCode = COM_End();
  if (RetCode != RC_OK)
  {
    // handle error
  }
```

3.2.3 C/C++ Development System Support

GeoCOM system files have been developed using Microsoft Visual C/C++ 5.0. Although this development environment were the basis for the current GeoCOM implementation, it has been emphasised that it is independent of it, hence other development environments can be used too. But please notice that it has not been tested thoroughly so far.

3.2.4 Programming Hints

Order of Include Statements

Since GeoCOM redefines `TRUE`, `FALSE` and `NULL` we recommend the following include order:

1. Include system headers like `stdio.h` or `stdafx.hpp`
2. Include `com_pub.hpp`
3. Include the current project headers

BOOLE Definition

GeoCOM defines its own Boolean type as an enumeration type of `FALSE` and `TRUE`. It is called `BOOLE`. With one exception, this does not produce any problems. Only if a `BOOL` type value will be assigned to a `BOOLE` type variable or parameter the compiler (MS-VisualC/C++) generates an error. To solve this problem the expression, which will be assigned to, has to be converted by a `CAST` statement to `BOOLE`.

3.3 VBA - PROGRAMMING

Similar to C/C++ programming the programming of VBA is based on the DLL concept. To enable access to GeoCOM the special module `stubs32p.bas` has to be included in the project. `stubs32p.bas` includes all constants, data types and function prototypes, which are available in GeoCOM.

3.3.1 Data Types in VBA - General rules for derivation

This subsection gives a summary of general derivation rules VBA-parameters from C-data types. Basically the C/C++ - data types are given in a C/C++ notation before they are used in a RPC-description.

If the appearance of a VBA data type does not follow the general rules then they are described explicitly.

In general, the following rules can be applied:

Numerical data type

The numerical data types correspond to the C/C++-parameters in value and range as close as possible. If it cannot be replaced directly then the best possible replacement will be taken.

String data type

Character and string types are replaced by `string` data types. Since string data types of C/C++ and VBA are not directly interchangeable, the programmer has to take certain care of the necessary pre- and post-processing of variables of this data type. Please refer to the example below.

Enumeration data type

Conceptually VBA does not have enumeration data types. Therefore, `Long` data types will be used instead. The enumeration values will be defined by constants. Using the numerical value is also valid. Notice that some of the enumeration values are reserved words in VBA. That is why we had to define different identifiers. Enumerated return values are numerical values and correspond to the position of the enumeration value in the C/C++-definition. For clarification, also the numerical values are given in the description of an enumeration data type.

Structures and Arrays

They are defined as in C/C++.

Example for Enumeration Data Types and Structures

The following example gives the data type declaration and the procedure declaration usually used in this manual for an example procedure (`TMC_GetSimpleMea` from the subsystem Theodolite Measurement and Calculation):

VBA-Declaration

```
VB_TMC_GetSimpleMea(
          WaitTime        As Long,
          OnlyAngle       As TMC_HZ_V_ANG,
          SlopeDistance   As Double,
          Mode            As Long)
```

In the file `stubs32p.bas` the corresponding items are defined:

```
Global Const TMC_MEA_INC = 0
Global Const TMC_AUTO_INC = 1
Global Const TMC_PLANE_INC = 2
Global Const TMC_APRIORI_INC = 3
Global Const TMC_ADJ_INC = 4
Global Const TMC_REQUIRE_INC = 5

Type TMC_HZ_V_ANG
  dHz  As Double
  dV   As Double
```

```
End Type
```

Obviously all enumeration values are encoded as global constants. The VBA structure definition equals to the C structure definition. A valid procedure call would be:

```
Dim WaitTime       As Long
Dim OnlyAngle      As TMC_HZ_V_ANG
Dim SlopeDistance  As Double

WaitTime = 1000

VB_TMC_GetSimpleMea( WaitTime,
                     OnlyAngle,
                     SlopeDistance,
                     TMC_AUTO_INC)
```

3.3.2 Basic GeoCOM Application Frame for VBA

Like in section 3.2.2 - Basic GeoCOM Application Frame for C/C++ - a VBA GeoCOM application consists at least of the following parts:

- Initialise GeoCOM
- Open a connection to the server
- One or more GeoCOM RPC’s
- Close the active connection to the server
- Finalise GeoCOM

A sample implementation of above points could be:

```
CONST RETRIES_1 = 1
DIM RetCode As Integer
DIM bOpenAndRunning as Integer

' initialize GeoCOM
bOpenAndRunning = False
RetCode = VB_COM_Init()
If (RetCode = RC_OK) Then
```

```
    ' open a channel to the TPS1100 instrument
    RetCode = VB_COM_OpenConnection(COM_1, COM_BAUD_19200,
                                    RETRIES_1)
    If (RetCode = RC_OK) Then
      bOpenAndRunning = True
    End If
  End If
  ' optionally set up other comm. parameters here

  If (RetCode = RC_OK)  Then
    ' functionality of the application
    ' we just test if communication is up
    RetCode = VB_COM_NullProc()
    If (RetCode <> RC_OK) Then
      ' handle error
    End If
  End If

  If (bOpenAndRunning) Then
    ' close channel
    RetCode = VB_COM_CloseConnection ()
    If (RetCode <> RC_OK) Then
      ' handle error
    End If
  End If

  ' finalize and reset GeoCOM
  RetCode = VB_COM_End()
  If (RetCode <> RC_OK) Then
    ' handle error
  End If
```

3.3.3 VBA Development System Support

This interface has been written for Microsoft Visual Basic for Applications 5.0 and higher only. Hence, no other development environment will be supported.

3.3.4 Programming Hints

Output Parameters of String Data Type

The internal representation of strings is not directly compatible between C/C++ and VBA. Therefore the one has to pre- and post-process such an output parameter. In the following example, we know that the output parameter will be less than 255 characters in length from the description of the RPC.

```
Dim s As String

' initialise string
s = Space(255)
Call VB_COM_GetErrorText(RC_IVPARAM, s)
' trim string, justify string length
s = Trim$(s)
```

Note: Incorrectly handled string output parameters may lead to severe runtime problems.

3.4 UNITS OF VALUES

All parameters are based on the SI unit definition, if not explicitly indicated differently. The SI units, and their derivatives, used are:

Abbreviation	Unit	Description
M	(Meters)	for lengths, co-ordinates, ...
Rad	(Radians)	for angles
Sec	(Seconds)	for time
Hpa	(Hekto Pascal)	for pressure
C	(Celsius)	for temperature

Table 3-2: SI Units

3.5 TPS1100 INSTRUMENT MODES OF OPERATION

In respect to communication, the TPS1100 instrument knows several states in which it reacts differently. The main state for GeoCOM is online state or mode. There it is possible to use all RPC's, which are described in this manual. Especially we will describe the possibilities of changing the state by the built-in RPC's. For the ASCII protocol refer to section 3.1.3 - Modes of Operation Concerning Communication.

The possible states can be described as follows:

Off The instrument is switched off and can be switched on and put into online mode by using `COM_SwitchOnTPS`.

Local The instrument is in local mode. GeoCOM is not active hence, RPC's cannot be used. To switch into online mode start the "Configuration" menu on the instrument, open the submenu "Communication mode" and select "GeoCOM On-Line mode" (in default configuration).

Online Also called Remote mode. The instrument accepts RPC's. `COM_Local` can be used to switch into local mode. `COM_SwitchOffTPS` will switch off the instrument or put it into sleep mode.

RCS The instrument accepts Remote Control sequences. This is not subject of this manual and will be described elsewhere.

GSI/Meas In this mode the user can measure coordinates and save the values onto the PC-Card. GSI commands will be accepted in this mode. Since this is not subject of this manual this mode will not described here in more detail.

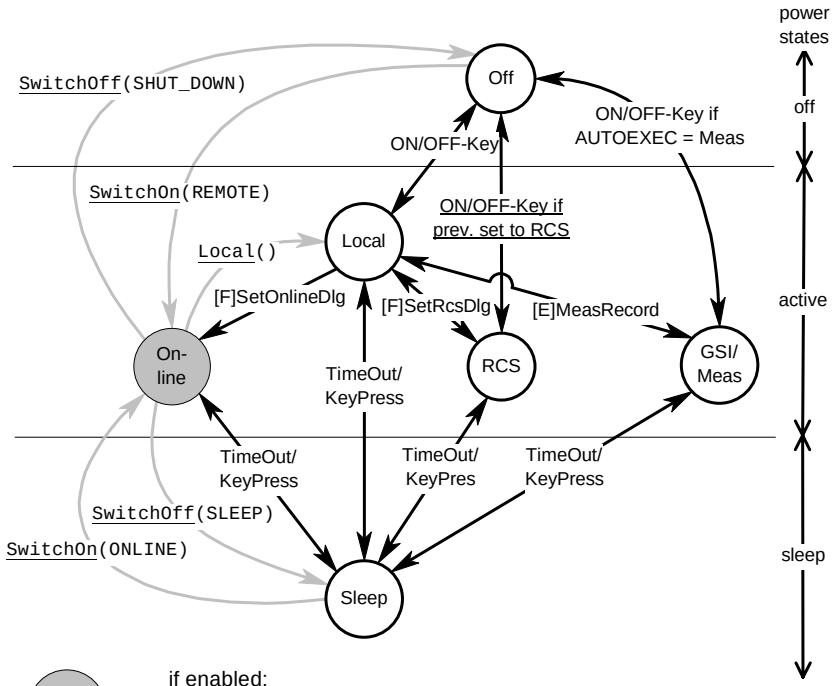
Sleep Either because of reaching the time out or by using the function `COM_SwitchOff(COM_TPS_SLEEP)` this state has been reached when starting from online mode. Only if the previous mode was online mode it can be switched back to it with `COM_SwitchOnTPS(COM_TPS_REMOTE)`.

3.5.1 Getting Mode of Operation Concerning Communication

This is available only when the application uses the function call protocol. Hence a DLL is used to automate RPC calls. That is why the current implementation does not log the current state per default. Logging can be switched on by using `COM_EnableSignOff(TRUE)`. After enabling state changes to and from online mode will be logged and can be requested with `COM_GetTPSState`.

Note: The mode can be determined only if GeoCOM is active and the sign-off message is activated. In any other situation, `COM_GetTPSState` will yield the state `COM_TPS_UNKNOWN`.

The following picture shows the dependencies graphically. Instead of the correctly defined identifiers, shortcuts have been used.



On-line	if enabled: sends SignOff message when leaving, sends SignOn message on arrival
RPC	which can change a state
[F]SetOnlineDlg	System function which shows a dialog for switching to online mode. Can be started in "Communication mode" menu on the default configuration.
[F]SetRcsDlg	System function which shows a dialog for switching to remote control mode. Can be started with the RCS key in the "PROG" menu on the default configuration.
[E]MeasRecord	System event which causes the starting of the measurement dialog. Can be generated by pressing the MEAS key in the "Main" menu on the default configuration.

Picture 3-1 TPS1100 modes of operation in respect to communication

3.6 COMMON COMMUNICATION ERRORS

GeoCOM is based on calling functions remotely. Because of the additional communication layer the set of return codes increases with return codes based on communication errors. Since all of these codes may be returned by any RPC we will explain them here and omit them in the descriptions of the RPC's. See also 20-Appendix

Return Codes.

Return Code	Val	Description
RC_OK	0	Successful termination, implies also no communication error.
RC_COM_CANT_ENCODE	3073	Can't encode arguments in client. Returned by the client to the calling application directly, i.e. without anything being sent to the transport layer and beyond.
RC_COM_CANT_DECODE	3074	Can't decode results in client. Once an RPC has been sent to the server and a reply has been sent back, this return code states that the encoded reply could not be decoded in the client. This is usually the result of using different versions of GeoCOM on client and server.
RC_COM_CANT_SEND	3075	Failure in sending calls. If the resources at the transmitting port have been allocated previously, i.e. GeoCOM does not have exclusive rights to the port, or if the exception or similar routine has experienced a failure, this error code is returned.
RC_COM_CANT_RECV	3076	Failure in receiving result. A failure has occurred during reception of a packet at the data link layer. This could be due to incorrect parameter settings or noise on the line, etc..

Return Code	Val	Description
RC_COM_TIMEDOUT	3077	Call timed out. The client has sent an RPC to the server but it has not replied within the current time-out period as set for the current transaction. This could be because: the server has not received the request; the server has taken too long to execute the request; the client has not received the reply; the communication line (physical layer is no longer there; or, the time-out is too short (especially true when communicating over noisy or radio links at low baud rates).
RC_COM_WRONG_FORMAT	3078	The request and receive formats are different. Something got mixed up along the way or the application tried to send using a format which has not been implemented on both client and server.
RC_COM_VER_MISMATCH	3079	RPC protocol mismatch error. An RPC protocol has been requested which does not exist. This error will indicate incompatible client and server protocols.
RC_COM_CANT_DECODE_REQ	3080	Can't decode request in server. If the client sends the server an RPC but one which cannot be decoded in the server, the server replies with this error. It could be that the GeoCOM versions running on the client and server are different or the packet was not correctly sent over a noisy or unreliable line.
RC_COM_PROC_UNAVAIL	3081	The requested procedure is unavailable in the server. An attempt has been made to call an RPC, which does not exist. This is usually caused when calling RPC's which have been inserted, appended, deleted, or altered between the differing versions of GeoCOM on client and server. To be on the safe side, always use the same GeoCOM version whenever possible on both sides.

Return Code	Val	Description
RC_COM_CANT_ENCODE_REP	3082	Can't encode reply in server. The server has attempted to encode the reply but has failed. This can be caused by the calling procedure trying to pass too much data back to the client and in so doing has exceeded the maximum packet length.
RC_COM_SYSTEM_ERR	3083	Communication hardware error
RC_COM_FAILED	3085	Mess into communication itself. Should be OK once the node has been recycled, i.e. powered-down and -up again.
RC_COM_NO_BINARY	3086	Unknown protocol. An unknown (or not yet supported) Transport or Network protocol has been used. Could appear when using differing GeoCOM versions on client and server.
RC_COM_INTR	3087	Call interrupted. Something has happened outside of the scope of GeoCOM, which has forced the current RPC to abort itself.
RC_COM_REQUIRES_8DBITS	3090	This error indicates desired protocol requires 8 data bits
RC_COM_TR_ID_MISMATCH	3093	Request and reply transaction ids do not match. Somewhere along the line a packet (usually a reply) has been lost or delayed. GeoCOM tries to bring everything back to order but if this error continues during the session it may be wise to inspect the line and, at least, to restart the session. The immediately following RPC may be lost.
RC_COM_NOT_GEOCOM	3094	Parse failed; data package not recognised as GeoCOM communication package
RC_COM_UNKNOWN_PORT	3095	Tried to access an unknown hardware port. The application has not taken the physical resources of the machine on which it is running into account.

Return Code	Val	Description
RC_COM_OVERRUN	3100	Overruns during receive. A packet has been received which has exceeded the maximum packet length. It will be discarded! This can be caused by a noisy line during GeoCOM Binary format transmissions.
RC_COM_SRVR_RX_ CHECKSUM_ERROR	3101	Checksum received at server is wrong. The checksum belonging to the current packet is wrong - no attempt is made at decoding the packet.
RC_COM_CLNT_RX_ CHECKSUM_ERROR	3102	Checksum received at client is wrong. The checksum belonging to the current packet is wrong - no attempt is made at decoding the packet.
RC_COM_PORT_NOT_ AVAILABLE	3103	COM port not available. This can be caused by attempting to open a port for unique use by GeoCOM, which has already been allocated to another application.
RC_COM_PORT_NOT_ OPEN	3104	COM port not opened / initialised. The application has attempted to use a COM port to which it has no unique rights.
RC_COM_NO_PARTNER	3105	No communications partner on other end. The connection to the partner could not be made or has been lost. Check that the line is there and try again.
RC_COM_ERO_NOT_ STARTED	3106	The client, after calling an ERO has decided not to confirm the start of the ERO and has instead called another RPC.
RC_COM_CONS_REQ	3107	Attention to send consecutive requests. The application has attempted to send another request before it has received a reply to its original request. Although GeoCOM does not return control to the app until a reply is received, this error is still possible with event-driven applications, i.e., the user pushing a button yields control back to the application code which can then call GeoCOM again.

Return Code	Val	Description
RC_COM_SRVR_IS_SLEEPING	3108	TPS has gone to sleep. Wait and try again.
RC_COM_SRVR_IS_OFF	3109	TPS has shut down. Wait and try again

4 REMARKS ON THE DESCRIPTION

This chapter contains some remarks on the description of RPC's and on the structure of the descriptions.

4.1 STRUCTURE OF DESCRIPTIONS

The whole reference part is subdivided into sections. Each section contains descriptions of a set of functions, which build up a subsystem. A subsystem gathers all functions, which are related to a specific functionality of a TPS1100 instrument, e.g. MOT describes all functions, which relate to motorization. Each subsystem is subdivided into the descriptions of RPC's.

4.1.1 Structure of a Subsystem

A subsystem consists of the following parts:

1. **Usage**
   This part gives some hints about the usage of the subsystem and general information of its functionality.
2. **Constants and Types**
   All subsystem specific constants and data types are listed here. Also their meanings are described if they are not obvious.
3. **Functions**
   All RPC's of this subsystems are listed here and described in detail.

Note: To reduce redundancy the VB declarations of data types and constants have been omitted. Please refer to chapter 3.3 to get more information about this subject.

4.1.2 Structure of a RPC Description

One RPC description contains the following parts:

Title

Contains the name of the RPC and a short description of the function.

C-Declaration

Contains the C declaration of the function (excluding the return type).

VB-Declaration

Declares the function in VB (excluding the return type).

ASCII-Request

Describes the composition, inclusive the base types, of the ASCII request.

ASCII-Reply

Describes the composition, inclusive the base types, of the corresponding reply.

Remarks

Gives additional information on the usage and possible side effects of the function.

Parameters

Explains the parameters, their data types and their meaning.

Return-Codes

Gives the meaning of the return codes related to this RPC. General and communication return codes will be omitted in explanations. They are explained in 3.6.

See Also

Cross references shows other RPC's which relate to this one.

Example

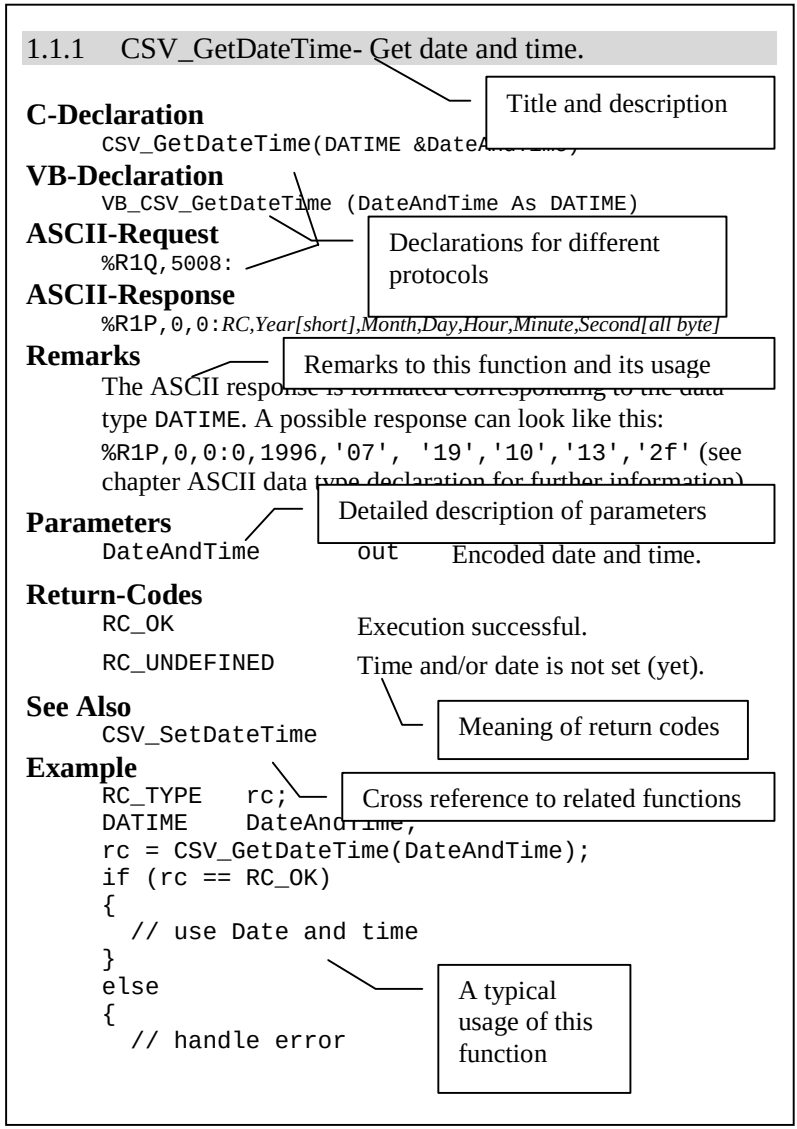
Gives an example of how this RPC could be used.

Note: To reduce redundancy the return type has been omitted from the C- and VB-declarations of the RPC's.

ASCII-Request and Reply do not explain the whole data structures. Instead the corresponding base types will be given. Please refer to chapter 2.2 to get more information on this topic.

Also because of redundancy the necessary CR/LF at the end has been omitted from ASCII-Request and Reply.

4.1.3 Sample of a RPC Description



1.1.1 CSV_GetDateTime- Get date and time.

Title and description

C-Declaration

```
CSV_GetDateTime(DATIME &DateAndTime)
```

VB-Declaration

```
VB_CSV_GetDateTime (DateAndTime As DATIME)
```

ASCII-Request

```
%R1Q,5008:
```

Declarations for different protocols

ASCII-Response

```
%R1P,0,0:RC,Year[short],Month,Day,Hour,Minute,Second[all byte]
```

Remarks

Remarks to this function and its usage

The ASCII response is formated corresponding to the data type `DATIME`. A possible response can look like this: `%R1P,0,0:0,1996,'07', '19','10','13','2f'` (see chapter ASCII data type declaration for further information)

Parameters

Detailed description of parameters

`DateAndTime`	`out`	Encoded date and time.

Return-Codes

`RC_OK`	Execution successful.
`RC_UNDEFINED`	Time and/or date is not set (yet).

Meaning of return codes

See Also

`CSV_SetDateTime`

Example

Cross reference to related functions

```
RC_TYPE   rc;
DATIME    DateAndTime;
rc = CSV_GetDateTime(DateAndTime);
if (rc == RC_OK)
{
  // use Date and time
}
else
{
  // handle error
```

A typical usage of this function

5 COMMUNICATION SETTINGS

This subsystem provides functions which influences GeoCOM as a whole and functions, which relate to the client side only.

If a function influences the client side only then there is no ASCII request defined.

5.1 CONSTANTS AND TYPES

Serial Port Selector

This enumeration type denotes the hardware serial port.

```
enum COM_PORT
{
        COM_1   = 0,            // port 1
        COM_2   = 1,            // port 2
        COM_3   = 2,            // port 3
        COM_4   = 3             // port 4
  };
```

Transmission Data Format

This value tells if the transmission takes place in a readable ASCII data format or in a data size optimised binary data format.

```
enum COM_FORMAT
{
        COM_ASCII  = 0,         // Force ASCII comm.
        COM_BINARY = 1          // Enable binary comm.
};
```

Baud Rate

```
enum COM_BAUD_RATE
{
        COM_BAUD_38400 = 0,
        COM_BAUD_19200 = 1,     // default baud rate
        COM_BAUD_9600  = 2,
        COM_BAUD_4800  = 3,
        COM_BAUD_2400  = 4
};
```

TPS1100 Operation Status

```
enum COM_TPS_STATUS
{
        COM_TPS_OFF         = 0,    // switched off
        COM_TPS_SLEEPING    = 1,    // sleep mode
        COM_TPS_ONLINE      = 2,    // online mode
        COM_TPS_LOCAL       = 3,    // local mode
        COM_TPS_UNKNOWN     = 4     // unknown or not initialised
};
```

MS-Windows Data Types

One of the described functions uses the predefined type `HWND` of MS-Windows. Please refer to the documentation of MS-Windows development environment for this data type.

Note: `HWND` depends on whether the pre-processor symbol `STRICT` is defined. When MFC libraries are used, `STRICT` is automatically defined. Otherwise the user must `#define STRICT` or he will get unresolved externals.

5.2 GENERAL GEOCOM FUNCTIONS

5.2.1 COM_GetDoublePrecision - Get Double Precision Setting

C-Declaration

```
COM_GetDoublePrecision( short &nDigits )
```

VB-Declaration

```
VB_COM_GetDoublePrecision( nDigits As Integer )
```

ASCII-Request

```
%R1Q,108:
```

ASCII-Response

```
%R1P,0,0:RC, nDigits[short]
```

Remarks

This function returns the precision - number of digits to the right of the decimal point - when double floating-point values are transmitted. The usage of this function is only meaningful if the communication is set to ASCII transmission mode. Precision is equal in both transmission directions. In the case of an ASCII request, the precision of the server side will be returned.

Parameters

NDigits	Out	Number of digits to the right of the decimal point.

Return Codes

RC_OK	On successful completion.

See Also

COM_SetDoublePrecision

Example

```
RC_TYPE         rc;
short           nDigits, nOldDigits;
TMC_HEIGT       height;

 (void) COM_GetDoublePrecision(nOldDigits);
rc = COM_SetDoublePrecision(nDigits);

// nDigits > 15, nDigits < 0 -> RC_IVPARAM
if (rc == RC_IVPARAM)
{
  rc = COM_SetDoublePrecision(7);
}

// measure height of reflector ...

// the result is precisely calculated and
// returned with nDigits to the right of the
// decimal point

(void) TMC_GetHeight(height);     // ignore return code
print(„height: %d\n“, height.dHr);

// reset server accuracy to the old value
rc = COM_SetDoublePrecision(nOldDigits);

// no error handling, because nOldDigits must be valid
```

5.2.2 COM_SetDoublePrecision - Set Double Precision Setting

C-Declaration

```
COM_SetDoublePrecision( short nDigits )
```

VB-Declaration

```
VB_COM_SetDoublePrecision( ByVal nDigits As Integer )
```

ASCII-Request

`%R1Q,107:`*nDigits[short]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function sets the precision - number of digits to the right of the decimal - when double floating-point values are transmitted. The TPS' system software always calculates with highest possible precision. The default precision is fifteen digits. However, if this precision is not needed then transmission of double data (ASCII transmission) can be speeded up by choosing a lower precision. Especially when many double values are transmitted this may enhance the operational speed. The usage of this function is only meaningful if the communication is set to ASCII transmission mode. In the case of an ASCII request, the precision of the server side will be set. Notice that trailing Zeros will not be sent by the server and values may be rounded. E.g. if precision is set to 3 and the exact value is 1.99975 the resulting value will be 2.0

Note: With this function one can decrease the accuracy of the delivered values.

Parameters

`nDigits`	`In`	Number of digits right to the comma.

Return Codes

`RC_OK`	On successful completion.
`RC_IVPARAM`	`0 > nDigits > 15`

See Also

`COM_GetDoublePrecision`

Example

see `COM_GetDoublePrecision`

5.3 CLIENT SPECIFIC GEOCOM FUNCTIONS

The following functions are not applicable to the ASCII protocol, because these functions influence the behaviour of the client application only.

5.3.1 COM_Init - Initialize GeoCOM

C-Declaration

```
COM_Init ( void )
```

VB-Declaration

```
VB_COM_Init ()
```

ASCII-Request

-

ASCII-Response

-

Remarks

`COM_Init` has to be called to initialise internal buffers and variables. It does not change the TPS' state.

Note: No other GeoCOM function can be called successfully without having initialised GeoCOM before.

Parameters

-

Return Codes

`RC_OK`	On successful completion.

See Also

`COM_End`

Example

See appendix C-2 for an example program frame.

5.3.2 COM_End - Quit GeoCOM

C-Declaration

```
COM_End( void )
```

VB-Declaration

```
VB_COM_End()
```

ASCII-Request

-

ASCII-Response

-

Remarks

COM_End has to be called to finish up all open GeoCOM transactions. It closes an open port and does whatever is necessary to shutdown GeoCOM. The TPS' state will not be changed.

Parameters

-

Return Codes

RC_OK	On successful completion.

See Also

COM_Init

Example

see COM_Init

5.3.3 COM_OpenConnection - Open a Port for Communication

C-Declaration

```
COM_OpenConnection( COM_PORT        ePort,
                    COM_BAUD_RATE  &eRate,
                    Short           nRetries )
```

VB-Declaration

```
VB_COM_OpenConnection( ByVal Port     As Integer,
                       ByVal Baud     As Integer,
                       ByVal Retries  As Integer )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function opens a PC serial port and attempts to detect a theodolite based on the given baud rate. If a TPS is well connected to the PC then GeoCOM tries to establish a connection to it.

If no connection could be established and the connection dialog flag is set to TRUE then a dialog appears which asks the user if all possible baud rate settings should be tried. See also `COM_SetConnDlgFlag`. If the flag is cleared, all possible settings will be tried automatically without notification of the user.

To be successful the TPS must be in online mode.

If the TPS is switched off it will be switched on remotely and set into online mode automatically.

The default transmission data format is set to `COM_BINARY`, if the TPSRelease is equal or higher than 2.00.

This function will fail if the TPS is in local-mode or if the serial-port is locked or in use. It will also fail if no TPS is connected to the serial port.

If the call cannot be finished successfully then the port will be freed and closed.

The successful completion of this may take more than a minute, if all possible settings have to be tried out.

`nRetries` denotes the number of retries of subsequent RPCs if the first request has not been fulfilled successfully. Especially for radio data links this is of interest if the link is not reliable. We recommend not using a value higher than two since this may slow down communication significantly.

Note: In the current implementation, GeoCOM does not support two open connections at the same time. A second attempt to open a second port at once will be denied by GeoCOM.

Parameters - C-Declaration

`EPort`	`In`	Serial port.
`eBaud`	`InOut`	Baud rate.
`nRetries`	`In`	Number of retries.

Return Codes

RC_OK	On successful completion.
RC_COM_PORT_NOT_AVAILABLE	Port is in use or does not exist
RC_COM_NO_PARTNER	GeoCOM failed to detect a TPS.
RC_IVPARAM	Illegal parameter.

See Also

COM_CloseConnection
COM_SetConnDlgFlag

Example

see COM_Init

5.3.4 COM_CloseConnection - Close the Open Port

C-Declaration

```
COM_CloseConnection( void )
```

VB-Declaration

```
VB_COM_CloseConnection( )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function closes the (current) open port and releases an established connection. It will not change the TPS' state.

Parameters

-

Return Codes

RC_OK	On successful completion.

See Also

COM_OpenConnection

Example

See appendix C-2 for an example program frame.

5.3.5 COM_GetBaudRate - Get Current Baud Rate

C-Declaration

```
COM_GetBaudRate ( COM_BAUD_RATE &eRate )
```

VB-Declaration

```
VB_COM_GetBautRate( eRate As Long )
```

ASCII-Request

-

ASCII-Response

-

Remarks

Get the current baud rate of the serial line. It should be the setting of both client and server. In ASCII protocol, this RPC is not available.

Parameters

eRate	Out	Baud rate of serial line.

Return Codes

RC_OK	On successful completion.

See Also

COM_SetBaudRate

Example

```
void main()
{
     RC_TYPE              rc;
     COM_BAUD_RATE        eRate;

     // init GeoCOM
     ...

     // get baud rate of active connection
       rc = COM_GetBaudRate(eRate);
       if (rc != RC_OK)
       {
         COM_ViewError(rc, "Setup baud rate");
       }
       else
       {
       printf("Baudrate is %d Baud = " );
       switch (eRate )
       {
```

```
            case COM_BAUD_38400:
              printf("38400\n");
              break ;
            case COM_BAUD_19200:
              printf("19200\n");
              break ;
            case COM_BAUD_9600:
              printf("9600\n ");
              break ;
            case COM_BAUD_4800:
              printf("4800\n ");
              break ;
            case COM_BAUD_2400:
              printf("2400\n ");
              break ;
            default:
              printf("illegal\n ");
              break ;
          }
        }

        ...
        // shutdown GeoCOM

    }    // end of main
```

5.3.6 COM_SetBaudRate - Set Baud Rate

C-Declaration

```
COM_SetBaudRate( COM_BAUD_RATE  eRate )
```

VB-Declaration

```
VB_COM_SetBaudRate( ByVal eRate As Long )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function sets the baud rate of the serial line, hence on both client and server side. A port must have been opened successfully with `COM_OpenConnection`.

Parameters

eRate	In	Baud rate.

Return Codes

RC_OK	On successful completion.

See Also

COM_GetBaudRate

Example

```
RC_TYPE         rc;


// set baud rate
rc = COM_SetBaudRate(COM_BAUD_9600);
if (rc == RC_IVPARAM)
{
  // handle errors
}
```

5.3.7 COM_GetTimeOut - Get Current Timeout Value

C-Declaration

```
COM_GetTimeOut( short &nTimeOut )
```

VB-Declaration

```
VB_COM_GetTimeOut( nTimeOut As Integer )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function retrieves the current timeout value for a request in seconds. The timeout value is the delay GeoCOM will wait for completion before it signals an error to the calling application.

Parameters

nTimeOut	Out	Timeout value in seconds, default value is 5 sec.

Return Codes

RC_OK	On successful completion.

See Also

COM_SetTimeOut

Example

```
RC_TYPE    rc;
short nTimeOut;

COM_GetTimeOut(nTimeOut);

if (nTimeOut <= 5)
{
  COM_SetTimeOut(7);
}
```

5.3.8 COM_SetTimeOut - Set Current Timeout Value

C-Declaration

```
COM_SetTimeOut( short nTimeOut )
```

VB-Declaration

```
VB_COM_SetTimeOut( nTimeOut As Integer )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function sets the current timeout value in seconds. The timeout value is the delay GeoCOM will wait for completion of the last RPC before it signals an error to the calling application.

A zero timeout value indicates no wait. This can be used for polling the input queue. But be aware of that this will yield into a RC_COM_TIMEDOUT return code.

Note: A negative timeout value indicates an infinite waiting period and may block the client application.

Parameters

nTimeOut	In	timeout value in seconds

Return Codes

RC_OK	On successful completion.

See Also

COM_GetTimeOut

Example

see COM_GetTimeOut

5.3.9 COM_GetComFormat - Get Transmission Data Format

C-Declaration

```
COM_GetComFormat( COM_FORMAT &eComFormat )
```

VB-Declaration

```
VB_COM_GetComFormat( eComFormat As Long )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function gets the actual transmission data format. GeoCOM uses COM_BINARY as a default with Firmware Releases 2.00 and higher. But if the TPS' Firmware does not support binary transmission data format then ASCII data format will be used instead.

Parameters

eComFormat	Out	COM_ASCII or COM_BINARY

Return Codes

RC_OK	On successful completion.

See Also

COM_SetComFormat

Example

```
RC_TYPE          rc;
COM_FORMAT    eComFormat;
```

```
COM_GetComFormat(eComFormat);
if (eComFormat == COM_ASCII)
{
   printf(„ASCII mode in use.\n“);
}
else
{
   printf(„BINARY mode in use.\n“);
}
```

5.3.10 COM_SetComFormat - Set Transmission Data Format

C-Declaration

```
COM_SetComFormat( COM_FORMAT eComFormat )
```

VB-Declaration

```
VB_COM_SetComFormat( ByVal eComFormat As Long )
```

ASCII-Request

-

ASCII-Response

-

Remarks

GeoCOM chooses during start-up the default transmission data-format. If the TPS Firmware Release is 2.00 or higher, then, this will be binary mode. Nevertheless, one can force ASCII data format for special purposes, e.g. debugging.
In the case of an ASCII request, the server side will be set only. The server always replies in the data-format that it has received the request. In this context this RPC can be used to deny binary data format.

Parameters

EComFormat	Out	COM_ASCII or COM_BINARY

Return Codes

RC_OK	On successful completion.
RC_COM_PORT_NOT_OPEN	Port not open for transmission.
RC_COM_NO_BINARY	TPS Firmware does not support binary data transmission format.

See Also

COM_SetComFormat

Example

```
RC_TYPE     rc;
COM_FORMAT  eFormat;

// change coding method
// eFormat is COM_ASCII or COM_BINARY
eFormat = COM_BINARY;
rc = COM_SetComFormat(eFormat);
if (rc == RC_COM_PORT_NOT_OPEN)
{
  rc = COM_SetComFormat(eFormat);
}

switch (rc)
{
  case RC_COM_PORT_NOT_OPEN:
    printf("Port not open\n");
    return (RC_FATAL);
    break;

  case RC_COM_NO_BINARY:
    printf("Binary format not available "
           "for this version.");
    // continue in ASII-format
    break;

}  // end of switch (rc)

// continue in program
```

5.3.11 COM_UseWindow - Declare Parent Window Handle

C-Declaration

```
COM_UseWindow( HWND handle )
```

VB-Declaration

```
VB_COM_UseWindow( handle As HWND )
```

ASCII-Request

-

ASCII-Response

-

Remarks

The function sets the parent window-handle that GeoCOM uses when it creates a dialog or message box. If this function is not called, GeoCOM will use the `NULL` window as default.

> **Note:** `HWND` depends on whether the pre-processor symbol `STRICT` is defined. When MFC libraries are used, `STRICT` is automatically defined. Otherwise the user must `#define STRICT` or he will get unresolved externals.

Parameters

`handle`	`In`	Parent window handle.

Return Codes

`RC_OK`	On successful completion.

See Also

-

Example

```
RC_TYE  rc;
HWND    hWnd;

rc = COM_UseWindow(hWnd);
```

5.3.12 COM_SetConnDlgFlag - Set Connection Dialog Flag

C-Declaration

```
COM_SetConnDlgFlag( BOOLE bShow )
```

VB-Declaration

```
VB_COM_SetConnDlgFlag( ByVal bShow As Long )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function sets or clears the connection dialog flag. If cleared, then the user will not be asked by a dialog box if all possible settings should be tried

to establish a connection. The flag is set to TRUE by default. Usually this function will be called beforehand of `COM_OpenConnection`.

Parameters

`bShow`	`In`	show dialog

Return Codes

`RC_OK`	On successful completion.

See Also

`COM_OpenConnection`

Example

```
Const short  RETRIES = 2;
RC_TYPE      rc;

// disable dialog box
COM_SetConnDlgFlag(FALSE);
// establish connection without showing dialog box
rc = COM_OpenConnection(PORT_1, COM_BAUD_9600, Retries);
```

5.3.13 COM_ViewError - Pop Up Error Message Box

C-Declaration

```
COM_ViewError( RC_TYPE  Result,
               char    *szMsgTitle )
```

VB-Declaration

```
VB_COM_ViewError( ByVal Result      As Integer,
                  ByVal szMsgTitle  As String)
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function checks the value of Result and if it is not equal to `RC_OK` then it pops up a message box containing the specific error text.

Note: This function yields a valid error text only if GeoCOM has been initialised successfully.

Parameters

`Result`	`In`	Error result code.
`szMsgTitle`	`In`	Title of the displayed dialog box.

Return Codes

`RC_OK` Always.

See Also

`COM_GetErrorText`

Example

```
RC_TYPE    rc;

// initialize GeoCOM
rc = COM_SetBaudRate(COM_BAUD_19200);

if (rc != RC_OK)
{
  COM_ViewError(rc, "Set up connection");
  // handle error
}
```

5.3.14 COM_GetErrorText - Get Error Text

C-Declaration

```
COM_GetErrorText( RC_TYPE Result,
                  char    *szErrText)
```

VB-Declaration

```
VB_COM_GetErrorText(ByVal Result     As Integer,
                          szErrText  As String)
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function checks the value of Result and returns an error text if the value is not equal to RC_OK. The function yields an empty string if the

value is RC_OK. The maximum length of such an error text is 255 characters.

Parameters

`Result`	`In`	Error code of a function called before this code will be checked.
`szErrText`	`Out`	Error text if not equal to `RC_OK`.

Return Codes

RC_OK	On successful completion.

See Also

`COM_ViewError`

5.3.15 COM_GetWinSWVersion - Retrieve Client Version Information

C-Declaration

```
COM_GetWinSWVersion( short &nRel,
                     short &nVer,
                     short &nSubVer )
```

VB-Declaration

```
VB_COM_GetWinSWVersion( nRel    As Integer,
                        nVer    As Integer,
                        nSubVer As Integer )
```

ASCII-Request

-

ASCII-Response

-

Remarks

This function retrieves the actual software Release (Release, version and subversion) of GeoCOM on the client side.

Parameters

`nRel`	`Out`	Software Release.
`nVer`	`Out`	Software version.
`nSubVer`	`Out`	Software subversion.

Return Codes

`RC_OK`	On successful completion.

See Also

COM_GetSWVersion

Example

```
RC_TYPE    rc;
short      nRel, nSubVer, nVer;

 (void) COM_GetWinSWVersion(nRel, nVer, nSubVer);

printf(„Windows GeoCOM:\n“);

printf(„Release %2d.%02d.%02d\n“, nRel, nVer, nSubVer);
```

5.3.16 COM_GetTPSState - Get Current TPS Operation Mode

C-Declaration

COM_GetTPSState(COM_TPS_STATUS &eMode)

VB-Declaration

VB_COM_ GetTPSState(eMode As Long)

ASCII-Request

-

ASCII-Response

-

Remarks

This function retrieves the current operation mode.

> **Note:** This function returns a valid value only if the sign-off message is enabled. In any other situation, it returns COM_TPS_UNKNOWN.
> It is important to enable the sign-off message again if the instrument has been shut down.

Parameters

eMode	Out	Current operation mode.

Return Codes

RC_OK	On successful completion.

See Also

COM_EnableSignOff
CTL_GetUpCounter

Example

```
RC_TYPE          rc;
COM_TPS_STATUS   eMode;

rc = COM_EnableSignOff(TRUE);
if (rc == RC_OK)
{
  rc = COM_GetTPSSTate(eMode);
  if (rc == RC_OK)
  {
    switch (eMode)
    {
       case COM_TPS_OFF:
         printf("TPS is switched off\n");
         break;

       ...

       case COM_TPS_UNKNOWN:
       default:
         printf("TPS state unknown\n");
    }
  }
}
```

6 ALT USER - AUS

6.1 USAGE

The subsystem 'Alt User' mainly contains functions behind the "FNC" button.

6.2 CONSTANTS AND TYPES

On/Off switch

```
enum ON_OFF_TYPE
{
  OFF,                    // 0
  ON                      // 1
};
```

6.3 FUNCTIONS

6.3.1 AUS_GetUserAtrState - Get the status of the ATR mode

C-Declaration

```
AUS_GetUserAtrState(ON_OFF_TYPE &OnOff)
```

VB-Declaration

```
VB_AUS_GetUserAtrState (OnOff As Long)
```

ASCII-Request

```
%R1Q,18006:
```

ASCII-Response

```
%R1P,0,0:RC,OnOff[long]
```

Remarks

Get the current status of the ATR mode on TCA instruments. This command does not indicate whether the ATR has currently acquired a prism. It replaces the function `AUT_GetAtrStatus`.

Parameters

`OnOff`	`out`	State of the ATR mode

Return-Codes

`RC_OK`	Execution always successful.

RC_NOT_IMPL	ATR not available; no TCA instrument.

See Also

AUS_SetUserAtrState

Example

```
RC_TYPE         rc;
ON_OFF_TYPE     OnOff;

// look for ATR state and set On if it is Off

rc = AUS_GetUserAtrState(OnOff);
if (OnOff == OFF)
{
  rc = AUS_SetUserAtrState(ON);
  if (rc == RC_OK)
  {
    // set of ATR status successful
  }
  else
  {
    // no TCA instrument
  }
}
```

6.3.2 AUS_SetUserAtrState - Set the status of the ATR mode

C-Declaration

```
AUS_SetUserAtrState(ON_OFF_TYPE OnOff)
```

VB-Declaration

```
VB_AUS_SetUserAtrState(OnOff As Long)
```

ASCII-Request

%R1Q,18005:*OnOff[long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

Activate respectively deactivate the ATR mode.

Activate ATR mode:
The ATR mode is activated and the LOCK mode (if sets) will be reset automatically also.

Deactivate ATR mode:
The ATR mode is deactivated and the LOCK mode keep unchanged.

This command is valid for TCA instruments only. It replaces the function `AUT_GetAtrStatus`.

Parameters

`OnOff`	`in`	State of the ATR mode

Return-Codes

`RC_OK`	Execution successful.
`RC_NOT_IMPL`	ATR not available; no TCA instrument.

See Also

`AUS_GetUserAtrState`
`AUS_GetUserLockState`
`AUS_SetUserLockState`

Example

`see AUS_GetUserAtrState`

6.3.3 AUS_GetUserLockState - Get the status of the lock switch

C-Declaration

```
AUS_GetUserLockState(ON_OFF_TYPE &OnOff)
```

VB-Declaration

```
VB_AUS_GetUserLockState(OnOff As Long)
```

ASCII-Request

`%R1Q,18008:`

ASCII-Response

`%R1P,0,0:`*RC, OnOff[long]*

Remarks

This command gets the current LOCK switch. This command is valid for TCA instruments only and does not indicate whether the ATR has a prism in lock or not.
With the function `MOT_ReadLockStatus` you can find out whether a target is locked or not.

This command is valid for TCA instruments only. It replaces the function `AUT_GetLockStatus`.

Parameters

OnOff	Out	State of the ATR lock switch

Return-Codes

RC_OK	Execution always successful.
RC_NOT_IMPL	ATR not available; no TCA instrument.

See Also

AUS_SetUserLockState
MOT_ReadLockStatus

Example

```
RC_TYPE        rc;
ON_OFF_TYPE    OnOff, OldAtrStatus;

rc = AUS_GetUserAtrState(OldAtrStatus); // save old mode
rc = AUS_GetUserLockState(OnOff);

if (OnOff == OFF)
{// ------ enable target tracking ------------------
  rc = AUS_SetUserLockState(ON); //set the ATR mode
                                 //automatically also!
  if (rc == RC_OK)
  {// set of Lock state successful
    rc = AUT_LockIn();// activate the real target
                      // tracking
    if(rc != RC_OK)
    {
       // error handling
    }
  }
  else
  {
    // no TCA instrument
  }
}
else
{// ------ disable target tracking ---------------
    rc = AUS_SetUserLockState(OFF);// reset the ATR
                                  // mode not
                                  // automatically
    if(rc == RC_OK)
    {// reset of Lock state successful
       if(OldAtrStatus==OFF)
       {// set old ATR mode
       rc == AUS_SetUserAtrState(OFF);
```

```
        }
    }
}
```

6.3.4 AUS_SetUserLockState - Set of the ATR lock switch

C-Declaration

`AUS_SetUserLockState(ON_OFF_TYPE OnOff)`

VB-Declaration

`VB_AUS_SetUserLockState(OnOff As Long)`

ASCII-Request

`%R1Q,18007:`*OnOff[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Set the lock status.

Status ON:
The target tracking functionality is available but not activated. In order to activate target tracking, see the function `AUT_LockIn`. The ATR mode will be set automatically.

Status OFF:
A running target tracking will be aborted and the manual driving wheel is activated. The ATR mode will be not reset automatically respectively keep unchanged.

This command is valid for TCA instruments only. It replaces the function `AUT_SetLockStatus`.

Parameters

`OnOff`	`in`	State of the ATR lock switch

Return-Codes

`RC_OK`	Execution successful.
`RC_NOT_IMPL`	ATR not available; no TCA instrument.

See Also

`AUS_GetUserLockState`
`AUS_SetUserAtrState`
`AUT_LockIn`

Example

see AUS_GetUserLockState

6.3.5 AUS_GetRcsSearchSwitch - Get RCS-Searching mode switch

C-Declaration

AUS_GetRcsSearchSwitch(ON_OFF_TYPE &OnOff)

VB-Declaration

VB_AUS_GetRcsSearchSwitch(OnOff As Long)

ASCII-Request

%R1Q,18010:

ASCII-Response

%R1P,0,0:*RC, OnOff[long]*

Remarks

This command gets the current RCS-Searching mode switch.

If RCS style searching is enabled, then the extended searching for BAP_SearchTarget or after a loss of lock is activated.

This command is valid for TCA instruments only.

Parameters

OnOff	Out	State of the RCS searching switch

Return-Codes

RC_OK	Execution always successful.
RC_NOT_IMPL	ATR not available; no TCA instrument.

See Also

AUS_SwitchRcsSearch

Example

-

6.3.6 AUS_SwitchRcsSearch - Set RCS-Searching mode switch

C-Declaration

AUS_SwitchRcsSearch(ON_OFF_TYPE OnOff)

VB-Declaration

```
VB_AUS_SwitchRcsSearch(OnOff As Long)
```

ASCII-Request

`%R1Q,18009:`*OnOff[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Set the RCS searching mode switch.

If the RCS style searching is enabled, then the extended for `BAP_SearchTarget` or after a loss of lock is activated.

This command is valid for TCA instruments only.

Parameters

`OnOff`	`in`	State of the RCS searching mode switch

Return-Codes

`RC_OK`	Execution successful.
`RC_NOT_IMPL`	ATR not available; no TCA instrument.

See Also

`AUS_SwitchRcsSearch`

Example

-

7 AUTOMATION - AUT

7.1 USAGE

The subsystem ‘Automation’ mainly performs the dynamic application ‘absolute positioning’. This operation positions the axes of the instrument within a given tolerance to the system’s angle measurement unit.

In combination with the Automatic Target Recognition System (ATR) other functionality such as automatic target position or target search are supported.

Some of the functions of this subsystem can take a undefined time for execution (for example the position operation takes the more time the more precision is required).

7.2 CONSTANTS AND TYPES

Number of axis

```
const short MOT_AXES = 2;
```

Positioning Tolerance

```
struct AUT_POSTOL
{
  double  adPosTol[MOT_AXES];
      // positioning tolerance for Hz and V [rad]
};
```

Maximum Position Time [s]

```
struct AUT_TIMEOUT
{
 double adPosTimeout[MOT_AXES];// max. positioning time [sec]
};
```

Position Precision

```
enum AUT_POSMODE
{
          AUT_NORMAL = 0,       // fast positioning mode
          AUT_PRECISE = 1       // exact positioning mode
                                // note: can distinctly claim more time
```

```
                                    //      for the positioning
}
```

Fine-adjust Position Mode

```
enum AUT_ADJMODE                    // Possible settings of the positioning
                                    // tolerance relating the angle- or the
                                    // point- accuracy at the fine adjust.
{
        AUT_NORM_MODE = 0           // Angle tolerance
        AUT_POINT_MODE  = 1         // Point tolerance
        AUT_DEFINE_MODE = 2         // System independent positioning
                                    // tolerance. Set with AUT_SetTol
}
```

Automatic Target Recognition Mode

```
enum AUT_ATRMODE                    // Possible modes of the target
                                    // recognition
{
        AUT_POSITION = 0,           // Positioning to the hz- and v-angle
        AUT_TARGET = 1              // Positioning to a target in the
                                    // environment of the hz- and v-angle.
}
```

Automatic Detent Mode

```
struct AUT_DETENT                   // Detent data
{
  double dPositiveDetent;           // Detent in positive direction
  double dNegativeDetent;           // Detent in negative direction
  BOOLE bActive;                    // Is detent active
}
```

Search Spiral

```
struct AUT_SEARCH_SPIRAL
{
  double dRangeHz;          // width of spiral [rad]
  double dRangeV;           // maximal height of spiral [rad]
}
```

Search Area

```
struct AUT_SEARCH_AREA
{
  double dCenterHz;         // Hz angle of spiral - center
  double dCenterV;          // V angle of spiral - center
  double dRangeHz;          // width of spiral [rad]
  double dRangeV;           // maximal height of spiral [rad]
  BOOLE  bEnabled;          // TRUE: user defined spiral is active
}
```

7.3 FUNCTIONS

7.3.1 AUT_ReadTol - Read current setting for the positioning tolerances

C-Declaration

```
AUT_ReadTol(AUT_POSTOL &TolPar)
```

VB-Declaration

```
VB_AUT_ReadTol(TolPar As AUT_POSTOL)
```

ASCII-Request

```
%R1Q,9008:
```

ASCII-Response

```
%R1P,0,0:RC,Tolerance Hz[double],Tolerance V[double]
```

Remarks

This command reads the current setting for the positioning tolerances of the Hz- and V- instrument axis.

This command is valid for TCM and TCA instruments only.

Parameters

TolPar	out	The values for the positioning tolerances in Hz and V direction [rad].

Return-Codes

RC_OK	Execution always successful.

See Also

AUT_SetTol

Example

```
const double MIN_TOL=3.141592654e-05;

RC_TYPE         rc;
AUT_POSTOL      TolPar;

// read tolerance and set to a minimum of
// 3.141592654e-05

rc = AUT_ReadTol(TolPar);

if ((TolPar.adPosTol[MOT_HZ_AXLE] > MIN_TOL) ||
   (TolPar.adPosTol[MOT_V_AXLE] > MIN_TOL))
{
  TolPar.adPosTol[MOT_HZ_AXLE] = MIN_TOL;
  TolPar.adPosTol[MOT_HZ_AXLE] = MIN_TOL;
  rc = AUT_SetTol(TolPar);
  switch (rc)
  {
    case (RC_OK):
      // set of Lock tolerance successful
      break;
    case (RC_IVPARAM):
      // invalid parameter
      break;
    case (MOT_RC_UNREADY):
      // subsystem not ready
      break;
  }
}
```

7.3.2 AUT_SetTol - Set the positioning tolerances

C-Declaration

```
AUT_SetTol(AUT_POSTOL  TolPar)
```

VB-Declaration

```
VB_AUT_SetTol(TolPar As AUT_POSTOL)
```

ASCII-Request

%R1Q,9007:*ToleranceHz[double], Tolerance V[double]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This command stops every movement and sets new values for the positioning tolerances of the Hz- and V- instrument axes. This command is valid for TCM and TCA instruments only.

The tolerances must be in the range of 1[cc] (=1.57079 E-06[rad]) to 100[cc] (=1.57079 E-04[rad]).

Note: The max. Resolution of the angle measurement system depends on the instrument accuracy class. If smaller positioning tolerances are required, the positioning time can increase drastically.

Parameters

TolPar	in	The values for the positioning tolerances in Hz and V direction [rad].

Return-Codes

RC_OK	Execution successful.
RC_IVPARAM	One or both tolerance values not within the boundaries (1.57079E-06[rad] =1[cc] to 1.57079E-04[rad] =100[cc]).
MOT_RC_UNREADY	Instrument has no motorization

See Also

AUT_ReadTol

Example

see AUT_ReadTol

7.3.3 AUT_ReadTimeout - Read current timeout setting for positioning

C-Declaration

AUT_ReadTimeout(AUT_TIMEOUT &TimeoutPar)

VB-Declaration

```
VB_AUT_ReadTimeout(TimeoutPar As AUT_TIMEOUT)
```

ASCII-Request

```
%R1Q,9012:
```

ASCII-Response

```
%R1P,0,0:RC, TimeoutHz[double], TimeoutV[double]
```

Remarks

This command reads the current setting for the positioning time out (maximum time to perform positioning).

Parameters

TimeoutPar	Out	The values for the positioning time out in Hz and V direction [sec].

Return-Codes

RC_OK	Execution always successful.

See Also

AUT_SetTimeout

Example

```
RC_TYPE         rc;
AUT_TIMEOUT     TimeoutPar;

// read timeout and set to a minimum of 10 [s]

rc = AUT_ReadTimeout(TimeoutPar);

if ((TimeoutPar.adPosTimeout[0] < 10) ||
   (TimeoutPar.adPosTimeout[1] < 10))
{
  TimeoutPar.adPosTimeout[0] = 10;
  TimeoutPar.adPosTimeout[1] = 10;
  rc = AUT_SetTimeout(TimeoutPar);
  switch (rc)
  {
    case (RC_OK):
      // set of timeout successful
      break;
```

```
      case (RC_IVPARAM):
        // invalid parameter
        break;
    }
}
```

7.3.4 AUT_SetTimeout - Set timeout for positioning

C-Declaration

`AUT_SetTimeout(AUT_TIMEOUT TimeoutPar)`

VB-Declaration

`VB_AUT_SetTimeout(TimeoutPar As AUT_TIMEOUT)`

ASCII-Request

`%R1Q,9011:`*TimeoutHz[double],TimeoutV[double]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This command set the positioning timeout (set maximum time to perform a positioning). The timeout is reset on 10[sec] after each power on

Parameters

`TimeoutPar`	`in`	The values for the positioning timeout in Hz and V direction [s]. Valid values are between 1 [sec] and 60 [sec].

Return-Codes

`RC_OK`	Execution successful. Maximum positioning time is set.
`RC_IVPARAM`	One or both time out values not within the boundaries (1[sec] to 60[sec]).

See Also

`AUT_ReadTimeout`

Example

see `AUT_ReadTimeout`

7.3.5 AUT_MakePositioning - Turns telescope to specified position

C-Declaration

```
AUT_MakePositioning(double Hz,
                    double V,
                    AUT_POSMODE POSMode,
                    AUT_ATRMODE ATRMode,
                    BOOLE bDummy)
```

VB-Declaration

```
VB_AUT_MakePositioning4(Hz As Double,
                        V As Double,
                        POSMode As Long,
                        ATRMode As Long,
                        bDummy As Boolean)
```

ASCII-Request

%R1Q,9027:*Hz,V,PosMode,ATRMode,0*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This procedure turns the telescope absolute to the in *Hz* and *V* specified position, taking tolerance settings for positioning (see AUT_POSTOL) into account. Any active control function is terminated by this function call.

If the position mode is set to normal (PosMode = AUT_NORMAL) it is assumed that the current value of the compensator measurement is valid. Positioning precise (PosMode = AUT_PRECISE) forces a new compensator measurement at the specified position and includes this information for positioning.

If ATR is possible and activated and the ATR mode is set to AUT_TARGET, the instrument tries to position onto a target in the destination area. In addition, the target is locked after positioning if the LockIn status is set. If the Lock status not set, the manual driving wheel is activated after the positioning.

Parameters

Hz	In	Horizontal (telescope) position [rad].
V	In	Vertical (instrument) position [rad].

POSMode	In	Position mode: AUT_NORMAL: (default) uses the current value of the compensator (no compensator measurement while positioning). For values >25GON positioning might tend to inaccuracy. AUT_PRECISE: tries to measure exact inclination of target. Tend to longer position time (check AUT_TIMEOUT and/or COM-time out if necessary).
ATRMode	In	Mode of ATR: AUT_POSITION: (default) conventional position using values Hz and V. AUT_TARGET: tries to position onto a target in the destination area. This mode is only possible if ATR exists and is activated.
bDummy	In	It's reserved for future use, set bDummy always to FALSE

Return-Codes

RC_OK	Execution successful.
RC_IVPARAM	Invalid parameter (e.g. no valid position).
AUT_RC_DETENT_ERROR	Destination angle lies in the illegal sector, collision is occurred. Set new Destination angle or free illegal sectors.
AUT_RC_TIMEOUT	Time out while positioning of one or both axes. (perhaps increase AUT time out, see AUT_SetTimeout)
AUT_RC_MOTOR_ERROR	Instrument has no 'motorization'.
AUT_RC_ANGLE_ERROR	Error within angle measurement. Maybe instrument is not levelled and not stable. Try to correct instrument's position or switch automatic incline correction to OFF (see TMC_SetInclineSwitch).

AUT_RC_INCACC	In the position mode AUT_PRECISE a valid measurement of target's inclination was not possible: position inexact.
RC_ABORT	Function aborted.
RC_COM_TIMEDOUT	Communication timeout. (perhaps increase COM timeout, see COM_SetTimeout)

Additionally with position mode AUT_TARGET.

AUT_RC_NO_TARGET	No target found
AUT_RC_MULTIPLE_TARGETS	Multiple targets found.
AUT_RC_BAD_ENVIRONMENT	Inadequate environment conditions.
AUT_RC_ACCURACY	Inexact fine position, repeat positioning
AUT_RC_DEV_ERROR	During the determination of the angle deviation error detected, repeat positioning
AUT_RC_NOT_ENABLED	ATR mode not enabled, enable ATR mode

See Also

AUS_GetUserAtrState, AUS_SetUserAtrState
AUS_GetUserLockState, AUS_SetUserLockState
AUT_ReadTol, AUT_SetTol
AUT_ReadTimeout, AUT_SetTimeout
COM_GetTimeOut, COM_SetTimeOut

Example

The example program tries to positioning to the given position. If a time out occurred, the time out values are increased and the position procedure starts again. If a measurement error occurred, the automatic inclination correction is switched off and the position procedure starts again.

```
RC_TYPE         rc, hrc;
short           i;
BOOL            TryAgain = TRUE;
AUT_TIMEOUT     TimeoutPar;
AUT_POSMODE     POSMode = AUT_PRECISE;
short           nComTimeOut, nOldComTimeOut;
```

```
rc=RC_IVRESULT;
hrc = COM_GetTimeOut(nOldComTimeOut);
hrc = AUS_SetUserAtrState(ON); // for the ATR mode
                              // AUT_TARGET necessary,
                              // otherwise not necessary

while(rc!=RC_OK || TryAgain)
{
  rc = AUT_MakePositioning(1.3, 1.6, POSMode,
                              AUT_TARGET, FALSE );
  switch (rc)
  {
    case (RC_OK):
      //Positioning successful and precise
      break;
    case (AUT_RC_TIMEOUT):
      // measure timeout fault: increase timeout
      hrc = AUT_ReadTimeout(TimeoutPar);
      TimeoutPar.adPosTimeout[0]
        = __min(TimeoutPar.adPosTimeout[0]+=5,60);
      TimeoutPar.adPosTimeout[1]
        = __min(TimeoutPar.adPosTimeout[1]+=5,60);
      hrc = AUT_SetTimeout(TimeoutPar);
      break;
    case RC_COM_TIMEDOUT:
      //increase timeout
      hrc = COM_GetTimeOut(nComTimeOut);
      nComTimeOut=__min(nComTimeOut+=5, 60);
      hrc = COM_SetTimeOut(nComTimeOut);
      break;
    case AUT_RC_ANGLE_ERROR:
      // error within angle measurement:
      // switch inclination correction off
      hrc = TMC_SetInclineSwitch(OFF);
      break;
    default:
      // precise position not possible
      TryAgain = FALSE;
      if (rc == AUT_RC_INCACC)
      {
        //Position successful but not precise
      }
      else
      {
        // Positioning not successful
        // here further error analyse possible
```

```
        }
        break;
    }
}
rc = AUS_SetUserAtrState(OFF); // Note: LOCK mode will
                               // be automatically
                               // reseted !
hrc = COM_SetTimeOut(nOldComTimeOut);// Set old time-
                                     // out
```

7.3.6 AUT_ChangeFace - Turns telescope to other face

C-Declaration

```
AUT_ChangeFace(AUT_POSMODE PosMode,
               AUT_ATRMODE ATRMode,
               BOOLE bDummy)
```

VB-Declaration

```
VB_AUT_ChangeFace4(PosMode As Long,
                   ATRMode As Long,
                   bDummy As Boolean)
```

ASCII-Request

`%R1Q,9028:`*PosMode,ATRMode,0*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This procedure turns the telescope to the other face.
Is in the moment of the function calling an other control function active it will be terminated before.

The start angle is automatically measured before the position starts.

If the position mode is set to normal (`PosMode = AUT_NORMAL`) it is allowed that the current value of the compensator measurement is inexact. Positioning precise (`PosMode = AUT_PRECISE`) forces a new compensator measurement. If this measurement is not possible, the position does not take place.

If ATR is possible and activated and the ATR mode is set `to AUT_TARGET` the instrument tries to position onto a target in the destination area. In addition, the target is locked after positioning if the `LockIn` status is set.

Parameters

POSMode	In	Position mode: AUT_NORMAL: uses the current value of the compensator. For values >25GON positioning might tend to inexact. AUT_PRECISE: tries to measure exact inclination of target. Tends to long position time (check AUT_TIMEOUT and/or COM-time out if necessary).
ATRMode	In	Mode of ATR: AUT_POSITION: conventional position to other face. AUT_TARGET: tries to position onto a target in the destination area. This set is only possible if ATR exists and is activated.
bDummy	In	It's reserved for future use, set bDummy always to FALSE

Return-Codes

General:

RC_OK	Execution successful.
RC_IVPARAM	Invalid parameter.
AUT_RC_DETENT_ERROR	Destination angle lies in the illegal sector, collision is occurred. Set new Destination angle.
AUT_RC_TIMEOUT	Timeout while positioning of one or both axes. (perhaps increase AUT timeout, see AUT_SetTimeout)
AUT_RC_MOTOR_ERROR	Instrument has no 'motorization'.
AUT_RC_ANGLE_ERROR	Error within angle measurement. Maybe instrument is not levelled and not stable. Try to correct instrument's position or switch automatic incline correction to OFF (see TMC_SetInclineSwitch).

AUT_RC_INCACC	In the position mode AUT_PRECISE a valid measurement of target's inclination was not possible. The positioning is inexact.
RC_FATAL	Fatal error.
RC_ABORT	Function aborted.
RC_COM_TIMEDOUT	Communication timeout. (perhaps increase COM timeout, see COM_SetTimeout)

Additionally with position mode AUT_TARGET.

AUT_RC_NO_TARGET	No target found
AUT_RC_MULTIPLE_TARGETS	Multiple targets found.
AUT_RC_BAD_ENVIRONMENT	Inadequate environment conditions.
AUT_RC_ACCURACY	Inexact fine position, repeat positioning
AUT_RC_DEV_ERROR	During the determination of the angle deviation error detected, repeat change face
AUT_RC_NOT_ENABLED	ATR mode not enabled, enable ATR mode

See Also

AUS_GetUserAtrState, AUS_SetUserAtrState
AUS_GetUserLockState, AUS_SetUserLockState
AUT_ReadTol, AUT_SetTol
AUT_ReadTimeout, AUT_SetTimeout
COM_GetTimeOut, COM_SetTimeOut
TMC_GetFace

Example

The example program performs a change face. If a measurement error occurs, the automatic inclination correction is switched off and the change face starts again.

```
RC_TYPE         rc, rch;
BOOL            TryAgain = TRUE;
AUT_POSMODE     POSMode = AUT_PRECISE;

rc=RC_IVRESULT;
```

```
while(rc!=RC_OK && TryAgain)
{
  rc = AUT_ChangeFace(POSMode,
                      AUT_POSITION,
                      FALSE);
  switch (rc)
  {
  case (RC_OK): // position successful
    //change face successful and precise
    break;
  case (AUT_RC_ANGLE_ERROR):
    //error within angle measurement:
    //switch inclination correction off
    rch = TMC_SetInclineSwitch(OFF);
    break;
  case (RC_COM_TIMEDOUT):
    //communication timed out while change face
    TryAgain = FALSE;
    break;
  default:
    //precise position not possible
    TryAgain = FALSE;
    if (rc == AUT_RC_INCACC)
    {
      //change face successful but not precise
    }
    else
    {
      // change face not successful
      // here further error analyse possible
    }
    break;
  }
}
```

7.3.7 AUT_FineAdjust - Automatic target positioning

C-Declaration

```
AUT_FineAdjust(       Double dSrchHz,
                      double dSrchV ,
                      BOOLE  bDummy)
```

VB-Declaration

```
VB_AUT_FineAdjust3(       DSrchHz As Double,
```

```
                        dSrchV  As Double,
                        bDummy As Boolean)
```

ASCII-Request

`%R1Q,9037:` `dSrchHz`*[double],* `dSrchV`*[double],0*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This procedure performs a positioning of the Theodolite axis onto a destination target. If the target is not within the sensor measure region a target search will be executed. The target search range is limited by the parameter `dSrchV` in V- direction and by parameter `dSrchHz` in Hz - direction. If no target found the instrument turns back to the initial start position. The ATR mode must be enabled for this functionality, see `AUS_SetUserAtrState` and `AUS_GetUserAtrState`.

Any actual target lock is terminated by this procedure call. After position, the target is not locked again.

The timeout of this operation is set to 5s, regardless of the general position timeout settings. The positioning tolerance is depends on the previously set up the fine adjust mode (see `AUT_SetFineAdjustMoed` and `AUT_GetFineAdjustMode`).

Tolerance settings (with `AUT_SetTol` and `AUT_ReadTol`) have no influence to this operation. The tolerance settings as well as the ATR measure precision depends on the instrument's class and the used EDM measure mode (The EDM measure modes are handled by the subsystem TMC).

Parameters

`DSrchHz`	In	Search range Hz-
`DSrchV`	In	Search range V-axis
`bDummy`	In	It's reserved for future use, set `bDummy` always to FALSE

Return-Codes

`RC_OK`	Execution successful.
`AUT_RC_TIMEOUT`	Timeout while positioning of one or both axes. The position fault lies above 100[cc]. (perhaps increase AUT timeout, see `AUT_SetTimeout`)

AUT_RC_MOTOR_ERROR	Instrument has no ‘motorization’.
RC_FATAL	Fatal error.
RC_ABORT	Function aborted.
AUT_RC_NO_TARGET	No target found.
AUT_RC_MULTIPLE_TARGETS	Multiple targets found.
AUT_RC_BAD_ENVIRONMENT	Inadequate environment conditions.
AUT_RC_DEV_ERROR	During the determination of the angle deviation error detected, repeat fine positioning
AUT_RC_NOT_ENABLED	ATR mode not enabled, enable ATR mode
AUT_RC_DETECTOR_ERROR	AZE error, at repeated occur call service
RC_COM_TIMEDOUT	Communication time out. (perhaps increase COM timeout, see COM_SetTimeout)

See Also

AUS_SetUserAtrState
AUS_GetUserAtrState
AUT_SetFineAdjustMode
AUT_GetFineAdjustMode

Example

```
RC_TYPE     Result;
ON_OFF_TYPE ATRState;
double  dHzSearchRange, dVSearchRange

dHzSearchRange=0.08;// search range in [rad]
dVSearchRange=0.08; // search range in [rad]

Result = AUS_GetUserAtrState(ATRState);// The ATR-
                                       // Status must
                                       // be set for
                                       // the fine
                                       // adjust
                                       // functionality
if(ATRState==ON)
{
     // performs a fine position with a max. target
     // search range of 0.08rad (5gon) in Hz and V
     // direction
```

```
        Result = AUT_FineAdjust(dHzSearchRange,
                                dVSearchRange,
                                FALSE);
        switch (Result) // function return code
        {
          case (RC_OK):
            //fine adjust successful and precise
            break;
          case (AUT_RC_NO_TARGET):
            //no target found.
            break;
          case (AUT_RC_MULTIPLE_TARGETS):
            //multiple targets found.
            break;
          case (AUT_RC_BAD_ENVIRONMENT):
            //inadequate environment conditions.
            break;
          default:
            //fine adjust not successful
            //here further error analyse possible
            break;
        }
    }
```

7.3.8 AUT_Search - Performs an automatically target search

C-Declaration

```
AUT_Search(double Hz_Area,
           double V_Area,
           BOOLE  bDummy)
```

VB-Declaration

```
VB_AUT_Search2(Hz_Area As Double,
               V_Area As Double,
               bDummy As Boolean)
```

ASCII-Request

`%R1Q,9029:`*Hz_Area,V_Area,0*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This procedure performs an automatically target search within a given area. The search area has an elliptical shape where the input parameters

determine the axis in horizontal and vertical direction. If the search was successful, the telescope will position to the target in a exactness of the field of vision (±1.25[GON]), otherwise the instrument turns back to the initial start position. With the ESC key a running search process will be aborted. The ATR mode must be enabled for this functionality, see `AUS_SetUserAtrState()` and `AUS_GetUserAtrState`. For a exact positioning use fine adjust (see `AUT_FineAdjust`) afterwards.

Note: If you expand the search range of the function `AUT_FineAdjust`, then you have a target search and a fine positioning in one function.

Parameters

`Hz_Area`	`In`	Horizontal search region [`rad`].
`V_Area`	`In`	Vertical search region [`rad`].
`bDummy`	`In`	It's reserved for future use, set `bDummy` always to `FALSE`

Return-Codes

`RC_OK`	Execution successful.
`RC_IVPARAM`	Invalid parameter.
`AUT_RC_MOTOR_ERROR`	Instrument has no 'motorization'.
`RC_FATAL`	Fatal error.
`RC_ABORT`	Function aborted.
`AUT_RC_NO_TARGET`	No target found.
`AUT_RC_BAD_ENVIRONMENT`	Inadequate environment conditions.
`AUT_RC_NOT_ENABLED`	ATR mode not enabled, enable ATR mode
`AUT_RC_DETECTOR_ERROR`	AZE error, at repeated occur call service
`RC_COM_TIMEDOUT`	Communication timeout. (perhaps increase COM timeout, see `COM_SetTimeout`)

See Also

`AUS_SetUserAtrState`
`AUS_GetUserAtrState`
`AUT_FineAdjust`

Example

The example program performs a search in the given area. If no target is found, the area is increased until 1[rad]. If a communication timeout occurs, the value for the communication timeout is increased until 30[s] (Note that a search over a big area takes a long time often results in an error).

```
RC_TYPE    rc, hrc;
BOOL       TryAgain = TRUE;
double     Hz_Area,V_Area;
short      nComTimeOut, nOldComTimeOut;

Hz_Area = 0.1;
V_Area  = 0.1;
rc = RC_IVRESULT;

hrc = COM_GetTimeOut(nOldComTimeOut);
hrc = AUS_SetUserAtrState(ON);    // activate ATR mode

while(rc!=RC_OK && TryAgain && hrc==RC_OK)
{
  rc = AUT_Search(Hz_Area,V_Area,FALSE);
  switch (rc)
  {
  case (RC_OK):
    // execution successful
    // Target found
    break;
  case (AUT_RC_NO_TARGET):
    //no target found.
    //increase search area
    Hz_Area += 0.1;
    V_Area += 0.1;
    if (Hz_Area > 1)
    {
      TryAgain = FALSE;
    }
    break;
  case (RC_COM_TIMEDOUT):
    //communication timeout
    //increase timeout until 30s
    hrc = COM_GetTimeOut(nComTimeOut);
    nComTimeOut=(short)__min(nComTimeOut+=5, 60);
    hrc = COM_SetTimeOut(nComTimeOut);
    //abort if timeout >= 30s
    if (nComTimeOut >= 30)
```

```
        {
          TryAgain = FALSE;
        }
        break;
      default:
        //error: search not possible
        //here further error analyse possible
        break;
      }
    }

    hrc = COM_GetTimeOut(nOldComTimeOut);// Set old time
                                         // out back
    hrc = AUS_SetUserAtrState(OFF);// Note: LOCK mode will
                                  // be automatically also
                                 // reseted!
```

7.3.9 AUT_GetFineAdjustMode - Get fine adjust positioning mode

C-Declaration

`AUT_GetFineAdjustMode(AUT_ADJMODE& rAdjMode)`

VB-Declaration

`VB_AUT_GetFineAdjustMode(AdjMode As Long)`

ASCII-Request

`%R1Q,9030:`

ASCII-Response

`%R1P,0,0:`*RC,AdjMode[integer]*

Remarks

This function returns the current activated fine adjust positioning mode. This command is valid for all instruments, but has only effects for TCA instruments.

Parameters

`RAdjMode`	Out	current fine adjust positioning mode

Return-Codes

`RC_OK`	Execution successful (always)

See Also

AUT_SetFineAdjustMode

Example

see AUT_SetFineAdjustMode

7.3.10 AUT_SetFineAdjustMode - Set the fine adjustment mode

C-Declaration

```
AUT_SetFineAdjustMode(AUT_ADJMODE AdjMode)
```

VB-Declaration

```
VB_AUT_SetFineAdjustMode(AdjMode As Long)
```

ASCII-Request

%R1Q,9031:*AdjMode[long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This function sets the positioning tolerances (default values for both modes) relating the angle accuracy or the point accuracy for the fine adjust. This command is valid for all instruments, but has only effects for TCA instruments. If a target is very near or held by hand, it's recommended to set the adjust-mode to AUT_POINT_MODE.

Parameters

AdjMode	in	AUT_NORM_MODE: Fine positioning with angle tolerance AUT_POINT_MODE: Fine positioning with point tolerance

Return-Codes

RC_OK	Execution successful
RC_IVPARAM	Invalid mode

See Also

AUS_GetUserAtrState

Example

```
RC_TYPE         Result;
AUT_ADJMODE     AdjMode;
```

```
Result=AUT_GetFineAdjustMode(AdjMode);
if(AdjMode!=AUT_MODE_POINT && Result==RC_OK)
{// change the finepositioning mode to AUT_MODE_POINT
     Result=AUT_SetFineAdjustMode(AUT_MODE_POINT);
     if(Result!=RC_OK)
     { // Error handling
     }
}
```

7.3.11 AUT_LockIn - Starts the target tracking

C-Declaration

`AUT_LockIn()`

VB-Declaration

`VB_AUT_LockIn()`

ASCII-Request

`%R1Q,9013:`

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Function starts the target tracking. Is at this time another ATR-configuration active, this configuration will be aborted before. The function can be called several times. If the target is already locked, the command will be ignored. The LOCK mode must be enabled for this functionality, see `AUS_SetUserLockState` and `AUS_GetUserLockState`. The ATR can only lock the target, if it is in the field of view (FoV).

Parameters

-

Return-Codes

`RC_OK`	LOCK-IN configuration is now active or already active.
`AUT_RC_NOT_ENABLED`	Target acquisition not enabled
`AUT_RC_MOTOR_ERROR`	Instrument has no ‘motorization’.
`AUT_RC_DETECTOR_ERROR`	Error in target acquisition, at repeated occur call service
`AUT_RC_NO_TARGET`	No target detected

AUT_RC_BAD_ENVIRONMENT		Bad environment conditions

See Also

AUS_SetUserLockState
AUS_GetUserLockState
MOT_ReadLockStatus

Example

```
RC_TYPE    result;

result = AUS_SetUserLockState(ON);// enable lock mode
if(result==RC_OK)
{
     result = AUT_LockIn(); // activate target tracking
     if(result != RC_OK)
     {
          // Error handling
     }
}
```

7.3.12 AUT_GetSearchArea - Get user searching area

C-Declaration

```
AUT_GetSearchArea( AUT_SEARCH_AREA &Area )
```

VB-Declaration

```
VB_AUT_GetSearchArea(Area As AUT_SEARCH_AREA)
```

ASCII-Request

```
%R1Q,9042:
```

ASCII-Response

%R1P,0,0:*RC,dCenterHz [double],dCenterV [double], dRangeHz[double],dRangeV [double],bEnabled[Boolean]*

Remarks

This function returns the current user searching area. This command is valid for all instruments, but has only effects for TCA instruments.

Parameters

Area	Out	user defined searching area

Return-Codes

RC_OK	Execution successful (always)

See Also

AUT_SetSearchArea, BAP_SearchTarget

Example

see AUT_SetSearchArea

7.3.13 AUT_SetSearchArea - Set user searching area

C-Declaration

```
AUT_SetSearchArea( AUT_SEARCH_AREA Area )
```

VB-Declaration

```
VB_AUT_SetSearchArea(byval Area As AUT_SEARCH_AREA)
```

ASCII-Request

%R1Q,9043:*dCenterHz,dCenterV,dRangeHz,dRangeV,bEnabled*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This function defines the user definable searching area. This command is valid for all instruments, but has only effects for TCA instruments.

Parameters

Area	In	user defined searching area

Return-Codes

RC_OK	Execution successful (always)

See Also

AUT_GetSearchArea, BAP_SearchTarget

Example

```
AUT_SEARCH_AREA   SearchArea;
SearchArea.dCenterHz  =  0.5;
SearchArea.dCenterV   =  1.5708;  // 100 gon
SearchArea.dRangeHz   =  0.4;
SearchArea.dRangeV    =  0.2;
SearchArea.bEnabled = TRUE;       // activate it
RetCode = AUT_SetSearchArea(SearchArea);
```

7.3.14 AUT_GetUserSpiral - Get user searching spiral

C-Declaration

```
AUT_GetUserSpiral( AUT_SEARCH_SPIRAL &SpiralDim )
```

VB-Declaration

```
VB_AUT_GetUserSpiral(SpiralDim As AUT_SEARCH_SPIRAL)
```

ASCII-Request

```
%R1Q,9040:
```

ASCII-Response

```
%R1P,0,0:RC,dRangeHz[double],dRangeV [double]
```

Remarks

This function returns the current dimension of the searching spiral. This command is valid for all instruments, but has only effects for TCA instruments.

Parameters

`SpiralDim`	Out	searching spiral dimension

Return-Codes

`RC_OK`	Execution successful (always)

See Also

`AUT_SetUserSpiral, BAP_SearchTarget`

Example

see `AUT_SetUserSpiral`

7.3.15 AUT_SetUserSpiral - Set user searching spiral

C-Declaration

```
AUT_SetUserSpiral( AUT_SEARCH_SPIRAL SpiralDim)
```

VB-Declaration

```
VB_AUT_SetUserSpiral(byval SpiralDim As
AUT_SEARCH_SPIRAL)
```

ASCII-Request

```
%R1Q,9041:dRangeHz,dRangeV
```

ASCII-Response

```
%R1P,0,0:RC
```

Remarks

This function sets the dimension of the searching spiral. This command is valid for all instruments, but has only effects for TCA instruments.

Parameters

`SpiralDim`	In	searching spiral dimension

Return-Codes

`RC_OK`	Execution successful (always)

See Also

`AUT_GetUserSpiral, BAP_SearchTarget`

Example

```
AUT_SEARCH_SPIRAL    SearchSpiral;
RC_TYPE              result;

SearchSpiral.dRangeHz = 0.4;
SearchSpiral.dRangeV = 0.2;
result = AUT_SetUserSpiral(SearchSpiral);
```

8 BASIC APPLICATIONS - BAP

The subsystem basic applications (BAP) contain high level functions for application programs.

8.1 CONSTANTS AND TYPES

Measurement Modes

```
enum BAP_MEASURE_PRG
  {
  BAP_NO_MEAS      = 0 // no measurements, take last one
  BAP_NO_DIST      = 1 // no dist. measurement,
                       // angles only
  BAP_DEF_DIST     = 2 // default distance measurements,
                       // pre-defined using
                       // BAP_SetMeasPrg
  BAP_CLEAR_DIST   = 5 // clear distances
  BAP_STOP_TRK     = 6 // stop tracking
                       // laser
  };
```

Distance measurement programs

```
enum BAP_USER_MEASPRG   {
  BAP_SINGLE_REF_STANDARD = 0,
     // standard single IR distance with reflector
  BAP_SINGLE_REF_FAST  = 1,
     // fast single IR distance with reflector
  BAP_SINGLE_REF_VISIBLE  = 2
     // long range distance with reflector (red laser)
  BAP_SINGLE_RLESS_VISIBLE   = 3,
     // single RL distance, reflector free (red laser)
  BAP_CONT_REF_STANDARD = 4,
     // tracking IR distance with reflector
  BAP_CONT_REF_FAST = 5,
     // fast tracking IR distance with reflector
  BAP_CONT_RLESS_VISIBLE = 6,
     // fast tracking RL distance, reflector free (red)
  BAP_AVG_REF_STANDARD = 7,
     // Average IR distance with reflector
  BAP_AVG_REF_VISIBLE = 8,
     // Average long range dist. with reflector (red)
  BAP_AVG_RLESS_VISIBLE = 9
     // Average RL distance, reflector free (red laser)
};
```

Number of prism types

```
BAP_NO_PRISMTYPES   = 7; // see BAP_PRISMTYPE
```

Prism type definition

```
enum BAP_PRISMTYPE
  {
  BAP_PRISM_ROUND = 0, // prism type: round
  BAP_PRISM_MINI  = 1, // prism type: mini
  BAP_PRISM_TAPE  = 2, // prism type: tape
  BAP_PRISM_360   = 3, // prism type: 360
  BAP_PRISM_USER1 = 4, // prism type: user1
  BAP_PRISM_USER2 = 5, // prism type: user2
  BAP_PRISM_USER3 = 6  // prism type: user3
  };
```

Reflector type definition

```
enum BAP_REFLTYPE
  {
  BAP_REFL_UNDEF = 0, // reflector not defined
  BAP_REFL_PRISM = 1, // reflector prism
  BAP_REFL_TAPE  = 2  // reflector tape
  };
```

Prism name length

```
BAP_PRISMNAME_LEN   = 16;  // prism name string
```

Prism definition

```
struct BAP_PRISMDEF
  {
  char         szName[BAP_PRISMNAME_LEN+1];
  double       dAddConst;  // prism correction
  BAP_REFLTYPE eReflType;  // reflector type
  }
```

Target type definition

```
enum BAP_TARGET_TYPE
  {
  BAP_REFL_USE  = 0 // with reflector
  BAP_REFL_LESS = 1 // without reflector
  };
```

8.2 FUNCTIONS

8.2.1 BAP_GetLastDisplayedError - Get last TPS system error number

C-Declaration

```
BAP_GetLastDisplayedError(short &nError,
                          short &nGSIError)
```

VB-Declaration

```
VB_BAP_GetLastDisplayedError(nError As Integer,
                             nGSIError As Integer)
```

ASCII-Request

```
%R1Q,17003:
```

ASCII-Response

`%R1P,0,0:`*RC, nError[short], nGSIError[short]*

Remarks

This function returns the last displayed error and clears it in the TPS system. So a second `GetLastDisplayedError` call will result in `RC_IVRESULT`.

Parameters

`nError`	out	Last displayed error-, info- or warning-number
`nGSIError`	out	Corresponding GSI error number

Return Codes

`RC_OK`	An error has been displayed before last call
`RC_IVRESULT`	No error has been displayed before last call

See Also

-

Example

```
RC_TYPE  rc;
short    nError, nGSIError;

rc = BAP_GetLastDisplayedError(nError, nGSIError);
if (rc == RC_OK)
  {
  printf("Error number: %d\n", nError);
  printf("GSI error number: %d\n", nGSIError);
```

```
  }
else if (rc == RC_IVRESULT)
  {
  printf("No error displayed before last call!");
  }
```

8.2.2 BAP_GetTargetType - Get actual target type

C-Declaration

```
BAP_GetTargetType( BAP_TARGET_TYPE &eTargetType )
```

VB-Declaration

```
VB_BAP_GetTargetType(eTargetType As Long)
```

ASCII-Request

```
%R1Q,17022:
```

ASCII-Response

%R1Q,0,0:*RC*, *eTargetType*[*long*]

Remarks

Gets the current target type for distance measurements (with reflector or without reflector).

Parameters

eTargetType	out	actual target type

Return Codes

RC_OK	Always

See Also

BAP_SetTargetType()
BAP_SetMeasPrg()

Example

-

8.2.3 BAP_SetTargetType – Sets the target type

C-Declaration

```
BAP_SetTargetType( BAP_TARGET_TYPE eTargetType )
```

VB-Declaration

```
VB_BAP_SetTargetType(byVal eTargetType As Long)
```

ASCII-Request

`%R1Q,17021:` *eTargetTyp*e *[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Defines the target type, with reflector or reflector-free

If the actual distance measurement not valid for the set target type, then the measurement program will be changed to the last used one for this type.

BAP_SetMeasPrg can also change the target type.

Reflector-free measurement programs are not available on all instrument types.

Parameters

eTargetType	in	target type

Return Codes

RC_OK	Target type is set
RC_IVPARAM	Target type is not available

See Also

BAP_GetTargetType()
BAP_SetMeasPrg()

Example

-

8.2.4 BAP_GetPrismType - Get actual prism type

C-Declaration

```
BAP_GetPrismType( BAP_PRISMTYPE &ePrismType )
```

VB-Declaration

```
VB_BAP_GetPrismType (ePrismType As Long)
```

ASCII-Request

`%R1Q,17009:`

ASCII-Response

`%R1Q,0,0:`*RC, ePrismTyp*e*[long]*

Remarks

Gets the current prism type.

Parameters

ePrismType	out	actual prism type

Return Codes

RC_OK	Always

See Also

BAP_SetPrismType()

Example

-

8.2.5 BAP_SetPrismType – Sets the prism type

C-Declaration

```
BAP_SetPrismType( BAP_PRISMTYPE ePrismType )
```

VB-Declaration

```
VB_BAP_SetPrismType(byVal ePrismType As Long)
```

ASCII-Request

%R1Q,17008: *ePrismTyp*e *[long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

Sets the prism type for measurements with a reflector. It overwrites the prism constant, set by TMC_SetPrismCorr.

Parameters

ePrismType	in	Prism type

Return Codes

RC_OK	Prism type is set
RC_IVPARAM	Prism type is not available, i.e. a user prism is not defined

See Also

BAP_GetPrismType()
TMC_SetPrismCorr()

Example

-

8.2.6 BAP_GetPrismDef - Get a prism definition

C-Declaration

```
BAP_GetPrismDef( BAP_PRISMTYPE ePrismType,
                 BAP_PRISMDEF &PrismDef)
```

VB-Declaration

```
VB_BAP_GetPrismDef(byval ePrism As Long,
                   PrismDef As BAP_PRISMDEF )
```

ASCII-Request

`%R1Q,17023:` *ePrismType[long]*

ASCII-Response

`%R1Q,0,0:`*RC, Name[String], dAddConst[double], eReflType[long]*

Remarks

Get the definition of a prism.

Parameters

`ePrismType`	`in`	actual prism type
`PrismDef`	`out`	definition of the prism

Return Codes

`RC_OK`	successful
`RC_IVPARAM`	invalid prism type

See Also

`BAP_SetPrismDef()`

Example

-

8.2.7 BAP_SetPrismDef – Sets a user prism definition

C-Declaration

```
BAP_SetPrismDef( BAP_PRISMTYPE ePrismType,
                 BAP_PRISMDEF PrismDef)
```

VB-Declaration

```
VB_BAP_SetPrismDef(byVal ePrismType As Long,
                   PrismDef As BAP_PRISMDEF)
```

ASCII-Request

`%R1Q,17024:` *ePrismTyp*e *[long] , Name[String], dAddConst[double], eReflType[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Defines an user prism.

Parameters

`ePrismType`	`in`	`Prism type. Valid: BAP_PRISM_USER1.. BAP_PRISM_USER3`
`PrismDef`	`in`	Definition of the prism

Return Codes

`RC_OK`	Prism type is set
`RC_IVPARAM`	Invalid prism type

See Also

`BAP_SetPrismType()`
`BAP_GetPrismDef()`

Example

-

8.2.8 BAP_GetMeasPrg - Get actual distance measurement program

C-Declaration

```
BAP_GetMeasPrg( BAP_USER_MEASPRG &eMeasPrg )
```

VB-Declaration

```
VB_BAP_GetMeasPrg(eMeasPrg As Long)
```

ASCII-Request

`%R1Q,17018:`

ASCII-Response

`%R1Q,0,0:`*RC*, *eMeasPrg[long]*

Remarks

-

Parameters

eMeasPrg	out	actual measurement program

Return Codes

RC_OK	Always

See Also

BAP_SetMeasPrg()

Example

-

8.2.9 BAP_SetMeasPrg - Set the distance measurement program

C-Declaration

```
BAP_SetMeasPrg( BAP_USER_MEASPRG eMeasPrg )
```

VB-Declaration

```
VB_BAP_SetMeasPrg(byVal eMeasPrg As Long)
```

ASCII-Request

%R1Q,17019:*eMeasPrg [long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

Defines the distance measurement program i.e. for BAP_MeasDistanceAngle

Reflector-free measurement programs are not available on all instrument types.

Changing the measurement programs may change the target type too (with reflector / reflector-free)

Parameters

eMeasPrg	in	Measurement program

Return Codes

RC_OK	Measurement program is set
RC_IVPARAM	Measurement program is not available

See Also

BAP_GetMeasPrg()
BAP_SetTargetType()

Example

-

8.2.10 BAP_MeasDistanceAngle - Measure distance and angles

C-Declaration

```
BAP_MeasDistanceAngle(BAP_MEASURE_PRG &DistMode,
                      double &dHz, double &dV,
                      double &dDist)
```

VB-Declaration

```
VB_BAP_MeasDistAng(DistMode As Long,
                   dHz As Double, dV As Double
                   dDist As Double)
```

ASCII-Request

%R1Q,17017:*DistMode[long]*

ASCII-Response

%R1P,0,0:*RC, dHz[double], dV[double], dDist[double],DistMode[long]*

Remarks

This function measures distances and angles depending on the mode DistMode and updates the internal data pool after correct measurements. It controls the special beep (sector or lost lock), maintains measurement icons and disables the "FNC"-key during tracking.

Parameters

DistMode	in	BAP_DEF_DIST, pre-defined using BAP_SetMeasPrg
DistMode	out	actual distance measurement mode
dHz	out	Horizontal angle [rad]x, depends on DistMode
dV	out	Vertical angle [rad]x, depends on DistMode
dDist	out	Slopedistance [m]x, depends on DistMode

Return Codes

BAP_MeasDistanceAngle may additionally return AUT- and TMC-return codes.

RC_OK	Measurement executed successfully
AUT_RC_ANGLE_ERROR	Angle measurement error
AUT_RC_BAD_ENVIRONMENT	Bad Environment conditions
AUT_RC_CALACC	ATR-calibration failed
AUT_RC_DETECTOR_ERROR	Error in target acquisition
AUT_RC_DETENT_ERROR	Positioning not possible due to mounted EDM
AUT_RC_DEV_ERROR	Deviation measurement error
AUT_RC_INCACC	Position not exactly reached
AUT_RC_MOTOR_ERROR	Motorization error
AUT_RC_MULTIPLE_TARGETS	Multiple targets detected
AUT_RC_NO_TARGET	No target detected
AUT_RC_TIMEOUT	Position not reached
BAP_CHANGE_ALL_TO_DIST	No prism has been found during distance measurement with ATR, command changed from "All" to "Dist"
TMC_ACCURACY_GUARANTEE	Info, accuracy cannot be guaranteed
TMC_ANGLE_ACCURACY_ GUARANTEE	Info, only angle measurement valid, accuracy cannot be guaranteed
TMC_ANGLE_ERROR	Error, no valid angle measurement
TMC_ANGLE_NO_FULL_ CORRECTION	Warning, only angle measurement valid, accuracy cannot be guaranteed
TMC_ANGLE_OK	Warning, only angle measurement valid
TMC_BUSY	Error, TMC submodule already in use by another subsystem, command not processed
TMC_DIST_ERROR	An error occurred during distance measurement.
TMC_DIST_PPM	Error, wrong setting of PPM or MM on EDM
TMC_NO_FULL_CORRECTION	Warning, measurement without full correction

TMC_SIGNAL_ERROR	Error, no signal on EDM (only in signal mode)
RC_ABORT	Error, measurement aborted
RC_COM_TIMEDOUT	Error, communication timeout. (possibly increase COM timeout, see COM_SetTimeout)
RC_IVPARAM	Error, invalid DistMode
RC_SHUT_DOWN	Error, system stopped

See Also

-

Example

```
void MyMeasurement(BAP_MEASURE_PRG DistMode)
  {
  RC_TYPE          Result;
  BAP_MEASURE_PRG  DistMode;
  double           dHz, dV, dDist;

  DistMode = BAP_DEF_DIST
  Result = BAP_MeasDistanceAngle(DistMode,
                                 dHz, dV, dDist);
  if (rc != RC_OK)
    { // error-handling
    switch (rc)
      {
      case RC_IVPARAM:
      printf("Wrong value for DistMode!");
      break;

      case RC_ABORT:
      printf("Measurement aborted!");
      break;

      case RC_SHUT_DOWN:
      printf("System has been stopped!");
      break;

      case TMC_DIST_PPM:
      printf("PPM or MM should be switched off");
      printf(" when EDM is on -> no results!“);
      break;

      case TMC_DIST_ERROR:
```

```
        printf("Error occured during");
        printf(" distance measurement!");
        break;

        case TMC_ANGLE_ERROR:
        printf("Error occured while slope");
        printf(" was measured!");
        break;

        case TMC_BUSY:
        printf("TMC is busy!");
        break;

        case TMC_ANGLE_OK:
        printf("Angle without coordinates!")
        break;
        } // end of switch (rc)
      } // end of error handling
    else
      { // use results
      printf("horizontal angel [rad]: %d\n", dHz);
      printf("vertical angel [rad]  : %d\n", dV);
      printf("slopedistance [rad]   : %d\n", dDist);
      }
    } //end of MyMeasurement
```

8.2.11 BAP_SearchTarget - Search the target

C-Declaration

```
BAP_SearchTarget(BOOLE bDummy)
```

VB-Declaration

```
VB_BAP_SearchTarget(bDummy As Boolean)
```

ASCII-Request

```
%R1Q,17020:0
```

ASCII-Response

```
%R1P,0,0:RC
```

Remarks

Function searches a target. The used searching range is dependent of the set searching area and whether the additional user area is enabled or not. The functionality is only available by ATR instruments.

Parameters

bDummy	in	It's reserved for future use, set bDummy always to FALSE

Return Codes

RC_OK	executed successfully
AUT_RC_BAD_ENVIRONMENT	Bad Environment conditions
AUT_RC_DEV_ERROR	Deviation measurement error
AUT_RC_ACCURACY	Position not exactly reached
AUT_RC_MOTOR_ERROR	Motorization error
AUT_RC_MULTIPLE_TARGETS	Multiple targets detected
AUT_RC_NO_TARGET	No target detected
AUT_RC_TIMEOUT	Time out, no target found
RC_ABORT	Error, searching aborted
RC_FATAL	Fatal Error

See Also

AUT_GetUserSpiral, AUT_SetUserSpiral, AUT_GetSearchArea, AUT_SetSearchArea

9 BASIC MAN MACHINE INTERFACE - BMM

The subsystem BMM (Basic Man Machine Interface) implements the low-level functions for the MMI. These are also functions which are relevant for controlling the display, keyboard, character sets and the beeper (signalling device). In GeoCOM only the beep control functions are supported. The description of the IOS beep control functions is also in this chapter, because there is a very close relationship to the BMM functions.

9.1 CONSTANTS AND TYPES

Constants for the signal-device

```
const short IOS_BEEP_STDINTENS = 100;
    // standard intensity of beep expressed as
    //a percentage
```

9.2 FUNCTIONS

9.2.1 BMM_BeepAlarm - Output of an alarm-signal

C-Declaration

```
BMM_BeepAlarm(void)
```

VB-Declaration

```
VB_BMM_BeepAlarm()
```

ASCII-Request

```
%R1Q,11004:
```

ASCII-Response

```
%R1P,0,0:RC
```

Remarks

This function produces a triple beep with the configured intensity and frequency, which cannot be changed. If there is a continuous signal active, it will be stopped before.

Parameters

-

Return Codes

`RC_OK` always

See Also

`BMM_BeepNormal, IOS_BeepOn, IOS_BeepOff`

9.2.2 BMM_BeepNormal - A single beep-signal

C-Declaration

```
BMM_BeepNormal(void)
```

VB-Declaration

```
VB_BMM_BeepNormal()
```

ASCII-Request

```
%R1Q,11003:
```

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function produces a single beep with the configured intensity and frequency, which cannot be changed. If a continuous signal is active, it will be stopped first.

Parameters

-

Return Codes

`RC_OK` always

See Also

`BMM_BeepAlarm, IOS_BeepOn, IOS_BeepOff`

9.2.3 IOS_BeepOn - Start a beep-signal

C-Declaration

```
IOS_BeepOn(short nIntens = IOS_BEEP_STDINTENS)
```

VB-Declaration

```
VB_IOS_BeepOn(ByVal nIntens As Integer)
```

ASCII-Request

`%R1Q,20001:`*Volume[short]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function switches on the beep-signal with the intensity `nIntens`. If a continuous signal is active, it will be stopped first. Turn off the beeping device with `IOS_BeepOff`.

Parameters

`nIntens`	`in`	Intensity of the beep-signal (volume) expressed as a percentage. Default value is 100 %

Return Codes

`RC_OK`	always

See Also

`IOS_BeepOff, BMM_BeepAlarm, BMM_BeepNormal`

Example

```
IOS_BeepOn(IOS_BEEP_STDINTENS)
// wait for a second

IOS_BeepOff();
```

9.2.4 IOS_BeepOff - Stop active beep-signal

C-Declaration

`IOS_BeepOff(void)`

VB-Declaration

`VB_IOS_BeepOff()`

ASCII-Request

`%R1Q,20000:`

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function switches off the beep-signal.

Parameters

-

Return Codes

RC_OK	always

See Also

IOS_BeepOn, BMM_BeepAlarm, BMM_BeepNormal

Example

see IOS_BeepOn

10 COMMUNICATIONS - COM

This subsystem contains those functions, which are subsystem COM related, but will be executed as RPC's on the TPS1100 instrument. It provides a function to check communication between the computer and the TPS1100 and also some functions to get and set communication relevant parameters on the server side. Furthermore, it implements functions to switch on or off (sleep mode, shut down) the TPS1100 instrument.

10.1 CONSTANTS AND TYPES

Stop Mode

```
enum COM_TPS_STOP_MODE
{
  COM_TPS_STOP_SHUT_DOWN =0,  // power down instrument
  COM_TPS_STOP_SLEEP     =1   // puts instrument into sleep state
};
```

Start Mode

```
enum COM_TPS_STARTUP_MODE
{
  COM_TPS_STARTUP_LOCAL =0  // RPC's enabled, local mode
  COM_TPS_STARTUP_REMOTE=1  // RPC's enabled, online mode
};
```

10.2 FUNCTIONS

10.2.1 COM_GetSWVersion - Retrieve Server Release Information

C-Declaration

```
COM_GetSWVersion(    short &nRel,
                     short &nVer,
                     short &nSubVer )
```

VB-Declaration

```
VB_COM_GetSWVersion( nRel     As Integer,
                     nVer     As Integer,
                     nSubVer  As Integer)
```

ASCII-Request

```
%R1Q,110:
```

ASCII-Response

`%R1P,0,0:`*RC, nDigits[short]*

Remarks

This function retrieves the current GeoCOM release (release, version and subversion) of the server.

Parameters

`nRel`	`out`	Software release.
`nVer`	`out`	Software version.
`nSubVer`	`out`	Software subversion (reserved).

Return Codes

`RC_OK`	On successful termination.

See Also

`CSV_GetSWVersion`
`COM_GetWinSWVersion`

Example

```
RC_TYPE    rc;
short      nRel, nSubVer, nVer;

COM_GetSWVersion(nRel, nVer, nSubVer);

printf(„TPS1100 GeoCOM Release:\n“);
printf(„Release      %02d\n“, nRel);
printf(„Version      %02d\n“, nVer);
printf(„Subversion   %02d\n“, nSubVer);
```

10.2.2 COM_SetSendDelay - Set Reply Delay

C-Declaration

```
COM_SetSendDelay(short nSendDelay)
```

VB-Declaration

```
VB_COM_SetSendDelay(ByVal nSendDelay As Integer)
```

ASCII-Request

`%R1Q,109:`*nSendDelay[short]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

The GeoCOM implementation of the server has been optimised for speed. If the server reacts to fast, then it may happen, that the client is not able to receive the reply (complete and) correctly. This RPC inserts a delay before the server responds to a request. This might be of interest especially for radio data links. Reset to no delay can be done with `nSendDelay = 0`.

Parameters

`nSendDelay`	`In`	Time of transmission delay in milliseconds.

Return Codes

`RC_OK`	On successful termination.

See Also

-

Example

```
RC_TYPE    rc;

rc = COM_SetSendDelay(5);
// do communication, long term RPC-calls
```

10.2.3 COM_Local - Switch TPS1100 into Local Mode

C-Declaration

```
COM_Local(void)
```

VB-Declaration

```
VB_COM_Local()
```

ASCII-Request

```
%R1Q,1:
```

ASCII-Response

```
%R1P,0,0:RC
```

Remarks

Leaves on-line mode and switches TPS1100 into local mode. If in local mode, no communication will take place. Any attempt of sending data will be ignored. Changing local into online mode can be done manually only.

Parameters

-

Return Codes

RC_OK	On successful termination.

See Also

COM_SwitchOffTPS
COM_SwitchOnTPS

Example

-

10.2.4 COM_SwitchOnTPS - Switch on TPS instrument

C-Declaration

```
COM_SwitchOnTPS(COM_TPS_STARTUP_MODE eOnMode)
```

VB-Declaration

```
VB_COM_SwitchOnTPS(ByVal eOnMode As Long)
```

ASCII-Request

%R1Q,111:*eOnMode[short]*

ASCII-Response

```
%N1,0,255,,0%T0,0,0,:%R1P,0,0:0
```

Remarks

This function switches on the TPS1100 instrument and put it into remote mode. It can also be used to switch from sleep into remote mode.

Parameters

eOnMode	In	Run mode - use COM_TPS_REMOTE only! COM_TPS_LOCAL will yield to erroneous behaviour.

Return Codes

RC_OK	On successful termination.

See Also

COM_SwitchOffTPS

Example

-

10.2.5 COM_SwitchOffTPS - Switch off TPS1100 or Set Sleep Mode

C-Declaration

```
COM_SwitchOffTPS(COM_TPS_STOP_MODE eOffMode)
```

VB-Declaration

```
VB_COM_SwitchOffTPS(ByVal eOffMode As Long)
```

ASCII-Request

`%R1Q,112:`*eOffMode[short]*

ASCII-Response

if RPC was successful and sign-off is enabled then

either `%N1,0,255,,0%T0,0,0,:%R1P,1,0:0,1` (sleep)

or `%N1,0,255,,0%T0,0,0,:%R1P,1,0:0,0` (shut down)

optionally followed by

`%R1P,0,0:`*RC*

else

`%R1P,0,0:`*RC*

Remarks

This function switches off the TPS1100 instrument or put it into sleep mode.

Parameters

`eOffMode`	`In`	Stop mode - see Constants and Types.

Return Codes

`RC_OK`	On successful termination.
`RC_COM_SRVR_IS_SLEEPING`	TPS has gone to sleep. Wait and try again. Only if called repetitive with `eOffMode == COM_TPS_SLEEP` and sign-off is enabled.

See Also

`COM_SwitchOnTPS`

Example

-

10.2.6 COM_NullProc - Check Communication

C-Declaration

`COM_NullProc(void)`

VB-Declaration

`VB_COM_NullProc()`

ASCII-Request

`%R1Q,0:`

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function does not provide any functionality except of checking if the communication is up and running.

Parameters

-

Return Codes

`RC_OK`	On successful termination.

See Also

-

Example

-

10.2.7 COM_EnableSignOff - Enable Remote Mode Logging

C-Declaration

`COM_EnableSignOff(BOOLE bEnable)`

VB-Declaration

`VB_COM_EnableSignOff(ByVal bEnable As Long)`

ASCII-Request

`%R1Q,115:`*bEnable[Boolean]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function enables logging if the Remote mode changes. See also section 3.5 TPS1100 Instrument Modes of Operation for further explanations.

Note: The flag will be reset each time the instrument turns on.

Parameters

`bEnable`	`In`	`TRUE`: enable mode logging

Return Codes

`RC_OK`	On successful termination.

See Also

`COM_GetTPSState`

Example

-

10.2.8 COM_GetBinaryAvailable - Get Binary Attribute of Server

C-Declaration

```
COM_GetBinaryAvailable(BOOLE &bAvailable)
```

VB-Declaration

```
VB_COM_GetBinaryAvailable(bAvailable As Long)
```

ASCII-Request

```
%R1Q,113:
```

ASCII-Response

`%R1P,0,0:`*RC, bAvailable[Boolean]*

Remarks

This function gets the ability information about the server to handle binary communication. Since TPS1100 Release 2.00 the client may make requests in binary format which speeds up the communication by about 40-50%.

Parameters

`bAvailable`	`Out`	`TRUE`: binary operation enabled. `FALSE`: ASCII operation enabled.

Return Codes

`RC_OK`	On successful termination.

See Also

COM_SetBinaryAvailable
COM_SetFormat
COM_GetFormat

10.2.9 COM_SetBinaryAvailable - Set Binary Attribute of Server

C-Declaration

COM_SetBinaryAvailable(BOOLE bAvailable)

VB-Declaration

VB_COM_SetBinaryAvailable(ByVal bAvailable As Long)

ASCII-Request

%R1Q,114:*bAvailable[Boolean]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This function sets the ability of the server to handle binary communication. With this function, one can force to communicate in ASCII only. During initialisation, the client checks if binary communication is enabled / possible or not which depends on this flag. Binary data format is not supported yet in GeoCOM Versions below 2.0.

Parameters

bAvailable	In	TRUE: enable binary operation. FALSE: enable ASCII operation only.

Return Codes

RC_OK	On successful termination.

See Also

COM_GetBinaryAvailable
COM_SetFormat

Example

-

11 CENTRAL SERVICES - CSV

The subsystem Central Services implements some centralised functions to maintain global data of the TPS system software. Examples are date and time or the instrument's name.

11.1 USAGE

These functions do not depend on other subsystems. Since this part is responsible for global data, any function can be called at any time.

11.2 CONSTANTS AND TYPES

TPS Device Configuration Type

```
struct  TPS_DEVICE
{
  TPS_DEVICE_CLASS Class; // device precision class
  TPS_DEVICE_TYPE  Type;  // device configuration type
};
```

TPS Device Precision Class

```
enum TPS_DEVICE_CLASS
{
  TPS_CLASS_1100 = 0,      // TPS1000 family member,
                           // 1 mgon, 3"
  TPS_CLASS_1700 = 1,      // TPS1000 family member,
                           // 0.5 mgon, 1.5"
  TPS_CLASS_1800 = 2,      // TPS1000 family member,
                           // 0.3 mgon, 1"
  TPS_CLASS_5000 = 3,      // TPS2000 family member
  TPS_CLASS_6000 = 4,      // TPS2000 family member
  TPS_CLASS_1500 = 5,      // TPS1000 family member
  TPS_CLASS_2003 = 6,      // TPS2000 family member
  TPS_CLASS_5005 = 7,      // TPS5000 family member
  TPS_CLASS_5100 = 8,      // TPS5000 family member
  TPS_CLASS_1102 = 100,    // TPS1100 family member, 2"
  TPS_CLASS_1103 = 101,    // TPS1100 family member, 3"
  TPS_CLASS_1105 = 102     // TPS1100 family member, 5"
  TPS_CLASS_1101 = 103     // TPS1100 family member, 1"
};
```

TPS Device Configuration Type

```
enum TPS_DEVICE_TYPE
{
  TPS_DEVICE_T   = 0x00000,  // theodolite without
                             // built-in EDM
  TPS_DEVICE_TC1 = 0x00001,  // tachymeter built-in
  TPS_DEVICE_TC2 = 0x00002,  // tachymeter with red
                             // red laser built-in
  TPS_DEVICE_MOT = 0x00004,  // motorized device
  TPS_DEVICE_ATR = 0x00008,  // automatic target
                             // recognition
  TPS_DEVICE_EGL = 0x00010,  // electronic guide light
  TPS_DEVICE_DB  = 0x00020,  // reserved
  TPS_DEVICE_DL  = 0x00040,  // diode laser
  TPS_DEVICE_LP  = 0x00080,  // laser plummet
  TPS_DEVICE_ATC = 0x00100,  // autocollimination lamp
  TPS_DEVICE_LPNT= 0x00200,  // Laserpointer
  TPS_DEVICE_RL_EXT = 0x00400,// Red laser with
                              //   extended range
  TPS_DEVICE_SIM = 0x04000   // runs on simulation,
                             // not on hardware
};
```

General Date and Time

```
struct  DATIME {
  DATE_TYPE   Date;
  TIME_TYPE   Time;
};
```

General Date

```
struct DATE_TYPE {
  short   Year;           // year
  BYTE    Month;          // month in year 1..12
  BYTE    Day;            // day in month 1..31
};
```

General Time

```
struct TIME_TYPE {
  BYTE    Hour;           // 24 hour per day  0..23
  BYTE    Minute;         // minute 0..59
  BYTE    Second;         // seconds 0..59
};
```

Power sources

```
struct CSV_POWER_PATH{
  CSV_CURRENT_POWER = 0,  // actual power source
  CSV_EXTERNAL_POWER = 1, // power source is external
  CSV_INTERNAL_POWER = 2  // power source is the
                          // internal battery
};
```

11.3 FUNCTIONS

11.3.1 CSV_GetInstrumentNo - Get factory defined instrument number

C-Declaration

```
CSV_GetInstrumentNo(long &SerialNo)
```

VB-Declaration

```
VB_CSV_GetInstrumentNo(SerialNo As Long)
```

ASCII-Request

```
%R1Q,5003:
```

ASCII-Response

```
%R1P,0,0:RC, SerialNo[long]
```

Remarks

-

Parameters

SerialNo	out	The serial number.

Return-Codes

RC_OK	Execution successful.
RC_UNDEFINED	Instrument number not yet set.

Example

```
RC_TYPE   rc;
long      SerialNo;

rc = CSV_GetInstrumentNo(SerialNo);
if (rc == RC_OK)
{
     // use SerialNo
}
else
```

```
{
      // instrument number not yet set
}
```

11.3.2 CSV_GetInstrumentName - Get Leica specific instrument name

C-Declaration

```
CSV_GetInstrumentName(char *Name)
```

VB-Declaration

```
VB_CSV_GetInstrumentName(Name As String)
```

ASCII-Request

```
%R1Q,5004:
```

ASCII-Response

```
%R1P,0,0:RC,Name[string]
```

Remarks

-

Parameters

Name	out	The instrument name

Return-Codes

RC_OK	Execution successful.
RC_UNDEFINED	Instrument name not set yet.

Example

```
RC_TYPE   rc;

rc = CSV_GetInstrumentName(szName);
if (rc == RC_OK)
{
  // use instrument name
}
else
{
  // instrument name not set yet
  // (incomplete calibration data)
}
```

11.3.3 CSV_GetDeviceConfig - Get instrument configuration

C-Declaration

```
CSV_GetDeviceConfig(TPS_DEVICE &Device);
```

VB-Declaration

```
VB_CSV_GetDeviceConfig(Device As TPS_DEVICE)
```

ASCII-Request

```
%R1Q,5035:
```

ASCII-Response

```
%R1P,0,0:RC, DevicePrecisionClass[long], DeviceConfigurationType[long]
```

Remarks

This function returns information about the class and the configuration type of the instrument.

Parameters

`Device`	out	System information (see data type description for further information).

Return-Codes

`RC_OK`	Well configured.
`RC_UNDEFINED`	Instrument precision class undefined.

Example

```
RC_TYPE      rc;
TPS_DEVICE   Device;

rc = CSV_GetDeviceConfig(Device);
if (rc == RC_OK)
{
    // Use system information
}
else
{
    // Intrument precision class undefined
    // (incomplete calibration data)
}
```

11.3.4 CSV_GetDateTime - Get date and time.

C-Declaration

```
CSV_GetDateTime(DATIME &DateAndTime)
```

VB-Declaration

```
VB_CSV_GetDateTime (DateAndTime As DATIME)
```

ASCII-Request

```
%R1Q,5008:
```

ASCII-Response

`%R1P,0,0:`*RC,Year[short],Month,Day,Hour,Minute,Second[all byte]*

Remarks

The ASCII response is formatted corresponding to the data type `DATIME`. A possible response can look like this: `%R1P,0,0:0,1996,'07','19','10','13','2f'` (see chapter ASCII data type declaration for further information)

Parameters

`DateAndTime`	out	Encoded date and time.

Return-Codes

`RC_OK`	Execution successful.
`RC_UNDEFINED`	Time and/or date are not set (yet).

See Also

`CSV_SetDateTime`

Example

```
RC_TYPE   rc;
DATIME    DateAndTime;

rc = CSV_GetDateTime(DateAndTime);
if (rc == RC_OK)
{
    // use Date and time
}
else
{
    // time and/or date is not set (yet)
    // use CSV_SetDateTime to set date and time
    // (March 25 1997, 10:20)
    DateAndTime.Date.Year   = 1997;
    DateAndTime.Date.Month  = 3;
```

```
        DateAndTime.Date.Day    = 25;
        DateAndTime.Time.Hour   = 10;
        DateAndTime.Time.Minute = 20;
        DateAndTime.Time.Second = 0;
        rc = CSV_SetDateTime(DateAndTime);
    }
```

11.3.5 CSV_SetDateTime - Set date and time

C-Declaration

`CSV_SetDateTime(DATIME DateAndTime)`

VB-Declaration

`VB_CSV_SetDateTime(ByVal DateAndTime As DATIME)`

ASCII-Request

`%R1Q,5007:`*Year[short],Month,Day,Hour,Minute,Second[all byte]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

It is not possible to set invalid date or time. See data type description of `DATIME` for valid date and time.

Parameters

`DateAndTime`	in	Encoded date and time.

Return-Codes

`RC_OK`	Execution always successful.

See Also

`CSV_GetDateTime`

Example

See `CSV_GetDateTime`.

11.3.6 CSV_GetSWVersion - Get Software Version

C-Declaration

```
CSV_GetSWVersion2(short &nRelease, short &nVersion,
                  short &nSubVersion)
```

VB-Declaration

```
VB_CSV_GetSWVersion2(nRelease As Integer,
                     nVersion As Integer,
                     nSubVersion As Integer)
```

ASCII-Request

```
%R1Q,5034:
```

ASCII-Response

```
%R1P,0,0:RC,nRelease,nVersion,nSubVersion[all short]
```

Remarks

Returns the system software version.

Parameters

nRelease	out	Release
nVersion	out	Version
nSubVersion	out	Sub Version

Return-Codes

RC_OK	Execution always successful.

Example

```
RC_TYPE  rc;
short    nRel, nVers, nSubVers;
char     szBuffer[17]

rc = CSV_GetSWVersion(nRel, nVers, nSubVers)
sprintf(szBuffer, "Version %02d.%02d.%02d",
        nRel, nVers, nSubVers);
```

Returns: `nRel = 2, nVers = 20, nSubVers = 0`
`szBuffer = “Version 02.20.00”`

11.3.7 CSV_CheckPower – check the available power

C-Declaration

```
CSV_CheckPower( unsigned short   &unCapacity,
                CSV_POWER_PATH   &eActivePower,
                CSV_POWER_PATH   &ePowerSuggest)
```

VB-Declaration

```
VB_CSV_CheckPower( unCapacity    As integer,
                   eActivePower  As long,
                   ePowerSuggest As long)
```

ASCII-Request

`%R1Q,5039:`

ASCII-Response

`%R1P,0,0:`*RC, unCapacity [long], eActivePower[long], ePowerSuggest[long]*

Remarks

This routine returns the capacity of the current power source and its source (internal or external).

Parameters

unCapacity	out	Actual capacity [%]
`eActivePower`	out	Actual power source
`ePowerSuggest`	out	Recommend power source

Return-Codes

`RC_OK`	Power ok
`RC_LOW_POWER`	Power is low
`RC_BATT_EMPTY`	Battery is nearly empty

Example

```
RC_TYPE  rc;
CSV_POWER_PATH eActivePower;
CSV_POWER_PATH ePowerSuggest;
unsigned short unCapacity;

rc = CSV_CheckPower(unCapacity, eActivePower,
                    ePowerSuggest)
```

11.3.8 CSV_GetVMem - Get value of the memory backup voltage supply

C-Declaration

`CSV_GetVMem(double &VMem)`

VB-Declaration

`VB_CSV_GetVMem(VMem As double)`

ASCII-Request

`%R1Q,5010:`

ASCII-Response

`%R1P,0,0:`*RC,VMem[double]*

Remarks

A value of Vmem > 3.1 V means OK.

Parameters

Vmem	out	Voltage of the memory backup system [V]. The voltage has an exactness of ±0.1V.

Return-Codes

RC_OK	Execution always successful.

Example

```
RC_TYPE   rc;
double    VMem;

rc = CSV_GetVMem(VMem);
if (Vbat > 3.1)
{
    // Message: memory backup voltage OK
}
else
{
    // Warning: memory backup voltage not OK
    // exchange battery it in Leica service
}
```

11.3.9 CSV_GetIntTemp - Get the temperature

C-Declaration

```
CSV_GetIntTemp(double &Temp)
```

VB-Declaration

```
VB_CSV_GetIntTemp(Temp As double)
```

ASCII-Request

```
%R1Q,5011:
```

ASCII-Response

```
%R1P,0,0:RC,Temp[long]
```

Remarks

Get the internal temperature of the instrument, measured on the Mainboard side. Values are reported in degrees Celsius.

Parameters

`Temp`	out	Instrument temperature.

Return-Codes

`RC_OK`	Execution always successful.

Example

```
RC_TYPE   rc;
double    Temp;

rc = CSV_GetIntTemp(Temp);
// use temperature information
```

12 CONTROLLER TASK - CTL

This chapter describes one RPC only, which shows how often the TPS1100 instrument has been switched on and how often the instrument fell asleep.

12.1 FUNCTIONS

12.1.1 CTL_GetUpCounter - Get Up Counter

C-Declaration

```
CTL_GetUpCounter( short &nPowerOn,
                  short &nWakeUp )
```

VB-Declaration

```
VB_CTL_GetUpCounter( nPowerOn As Integer,
                     nWakeUp  As Integer )
```

ASCII-Request

```
%R1Q,12003:
```

ASCII-Response

```
%R1P,0,0:RC, nPowerOn[short] , nWakeUp[short]
```

Remarks

This function retrieves how often, since the last call of this function, the TPS1100 instrument has been switched on and how often it has been awakened from sleep mode. Both counters are unique and will be reset to Zero once the function has been called.

Parameters

`nPowerOn`	`out`	Switch on counter.
`nWakeUp`	`out`	Waken up counter.

Return Codes

`RC_OK`	On successful termination.

See Also

```
COM_GetTPSState
COM_SwitchOffTPS
```

Example

```
RC_TYPE    RetCode;
short      nPowerOn, nWakeUp;

// do some stuff

RetCode = CTL_GetUpCounter(nPowerOn, nWakeUp);
if (RetCode != RC_OK)
{
  // handle error
}

if (nPowerOn > 0)
{
  // instrument has been switched off in between
}

if (nWakeUp > 0)
{
  // instrument has been fallen asleep in between
}
...
```

13 ELECTRONIC DISTANCE MEASUREMENT - EDM

The subsystem electronic distance measurement (EDM) is the adaptation of the distance measurement devices on the Theodolite.

With the functionality of EDM one can switch on or off the Laserpointer and the Electronic Guide Light respectively. Additionally the same functions as to switch on or off the Electronic Guide Light (`EDM_SetEGLIntensity` and `EDM_GetEGLIntensity`) make it possible to change the intensity (brightness).

13.1 USAGE

In order to use the functions concerning the Laserpointer and the Electronic Guide Light, make sure these devices are available. If not, these functions returns error messages.

13.2 CONSTANTS AND TYPES

On/off switch

```
enum ON_OFF_TYPE // on/off switch type
    {
    OFF = 0,
    ON  = 1
    };
```

Intensity of Electronic Guidelight

```
typedef enum EDM_EGLINTENSITY_TYPE
    {
    EDM_EGLINTEN_OFF      = 0,
    EDM_EGLINTEN_LOW      = 1,
    EDM_EGLINTEN_MID      = 2,
    EDM_EGLINTEN_HIGH     = 3
    };
```

13.3 FUNCTIONS

13.3.1 EDM_Laserpointer - Switch on/off laserpointer

C-Declaration

```
EDM_Laserpointer(ON_OFF_TYPE eLaser)
```

VB-Declaration

```
VB_EDM_Laserpointer(ByVal eLaser As Long)
```

ASCII-Request

%R1Q,1004:*eLaser[long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

Laserpointer is only available in theodolites which supports distance measurement without reflector.

Parameters

eOn	in	ON - switch Laserpointer on OFF - switch Laserpointer off

Return Codes

RC_OK	Laserpointer switched on/off
EDM_DEV_NOT_INSTALLED	Laserpointer is not implemented
EDM_RC_HWFAILURE	hardware error in EDM occurred
EDM_COMERR	error on communication with EDM
RC_TIME_OUT	process time out
RC_ABORT	function interrupted
RC_SYSBUSY	EDM already busy
RC_IVPARAM	parameter value must be ON or OFF
RC_UNDEFINED	instrument name could not be read

See Also

-

Example

```
RC_TYPE    rc;

// switch on laserpointer
rc = EDM_Laserpointer(ON);

if (rc != RC_OK)
  { // error-handling
  switch (rc)
    {
    case EDM_DEV_NOT_INSTALLED:
      printf("Laserpointer is not implemented.
              Laserpointer is only available in
              theodolites which supports distance
              measurement without reflector.");
      break;

    case EDM_RC_HWFAILURE:
      printf("Hardware error occured in EDM!");
      break;

    case EDM_COMERR:
      printf("Error on communication with EDM!");
      break;

    case RC_TIME_OUT:
      printf("Process time out");
      break;

    case RC_ABORT:
      printf("Function aborted!");
      break;

    case RC_UNDEFINED:
      printf("Instrument name is not set!");
      printf("Fatal error: call service!");
      break;

    case RC_SYSBUSY:
      printf("EDM is already busy!");
      break;

    case RC_IVPARAM:
      printf("Parameter of EDM_Laserpointer
              must be of ON_OFF_TYPE!");
      break;
```

```
    } // end of switch (rc)
  }  // end of error handling
else if (rc == RC_OK)
  {
  // use laserpointer
  }
```

13.3.2 EDM_GetEglIntensity - Get value of intensity of guide light

C-Declaration

```
EDM_GetEglIntensity(EDM_EGLINTENSITY_TYPE
                          &eIntensity)
```

VB-Declaration

```
VB_EDM_GetEglIntensity (eIntensity As Long)
```

ASCII-Request

```
%R1Q,1058:
```

ASCII-Response

`%R1Q,0,0:`*`RC`*`,`*eIntensity[long]*

Remarks

The Electronic Guide Light must be implemented in the theodolite.

Parameters

`eBrightness`	`out`	`EDM_EGLINTEN_OFF`
		`EDM_EGLINTEN_LOW`
		`EDM_EGLINTEN_MID`
		`EDM_EGLINTEN_HIGH`

Return Codes

`RC_OK`	value of the Electronic Guide Light intensity returned
`EDM_DEV_NOT_INSTALLED`	Electronic Guide Light not implemented

See Also

```
EDM_ SetEglIntensity ()
```

Example

See `EDM_SetEglIntensity`.

13.3.3 EDM_SetEglIntensity - Change intensity of guide light

C-Declaration

```
EDM_SetEglIntensity (EDM_EGLINTENSITY_TYPE
                     eIntensity)
```

VB-Declaration

```
VB_EDM_SetEglIntensity (ByVal eIntensity As
                        Long)
```

ASCII-Request

`%R1Q,1059:`*eIntensity [long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

The Electronic Guide Light must be implemented in the theodolite.

Parameters

`eBrightness`	in	`EDM_EGLINTEN_OFF`
		`EDM_EGLINTEN_LOW`
		`EDM_EGLINTEN_MID`
		`EDM_EGLINTEN_HIGH`

Return Codes

`RC_OK`	Electronic Guide Light intensity is set
`RC_SYSBUSY`	EDM already busy
`EDM_DEV_NOT_INSTALLED`	Electronic Guide Light not implemented

See Also

`EDM_GetEglIntensity ()`

Example

```
RC-TYPE rc;
EDM_EGLINTENSITY_TYPE eIntensity, eNewIntensity;

// Get actual EGL intensity
rc = EDM_GetEglIntensity(eIntensity);

if (rc == RC_OK)
{
  // switch EGL intensity one level up
  switch (eIntensity)
  {
```

```
        case EDM_EGLINTENSITY_OFF:
          eIntensityNew = EDM_EGLINTENSITY_LOW; break;

        case EDM_EGLINTENSITY_LOW:
          eIntensityNew = EDM_EGLINTENSITY_MID; break;

        case EDM_EGLINTENSITY_MID:
          eIntensityNew = EDM_EGLINTENSITY_HIGH;break;

        case EDM_EGLINTENSITY_HIGH:
          break; // Allready highest intensity

        default:
          eIntensityNew = EDM_EGLINTENSITY_LOW;
      }
    //Set new EGL intensity
    rc = SetEglIntensity(eIntensityNew);

    // Handle errors
  }
```

14 MOTORISATION - MOT

The subsystem ‘Motorisation’ controls the motorised drive of the axis.

14.1 USAGE

Within the subsystem, there exist three different types of functions:

"Open-End" functions: These functions start a motorisation control task and continue execution until cancellation. Special control functions are used to cancel such functions. An example for this type of function is the speed control function `MOT_SetVelocity`.

"Terminating" functions: These functions start control tasks, which terminate automatically. Examples for this type are positioning functions for example `MOT_StartController` and `MOT_StopController`.

Functions for the parameter handling: These functions manage system parameters. Examples are control parameter, motion parameter, tolerance and system configuration parameters (Example: `MOT_ReadLockStatus`).

14.2 CONSTANTS AND TYPES

Lock Conditions

```
enum MOT_LOCK_STATUS
{
  MOT_LOCKED_OUT = 0,    // locked out
  MOT_LOCKED_IN  = 1,    // locked in
  MOT_PREDICTION = 2     // prediction mode
};
```

Controller Stop Mode

```
enum MOT_STOPMODE
{
  MOT_NORMAL   = 0,      // slow down with current acceleration
  MOT_SHUTDOWN = 1       // slow down by switch off power supply
};
```

Values for Horizontal (instrument) and Vertical (telescope) Speed

```
struct MOT_COM_PAIR
{
  double  adValue[MOT_AXES];
};
```

Controller Configuration

```
enum MOT_MODE
{
  MOT_POSIT     = 0,   // configured for relative postioning
  MOT_OCONST    = 1,   // configured for constant speed
  MOT_MANUPOS   = 2,   // configured for manual positioning
                       // default setting
  MOT_LOCK      = 3,   // configured as "Lock-In"-controller
  MOT_BREAK     = 4,   // configured as "Brake"-controller
                       // do not use 5 and 6
  MOT_TERM      = 7    // terminates the controller task
};
```

Number of axis

```
const short MOT_AXES = 2;
```

14.3 FUNCTIONS

14.3.1 MOT_ReadLockStatus - Return condition of LockIn control

C-Declaration

```
MOT_ReadLockStatus(MOT_LOCK_STATUS &Status)
```

VB-Declaration

```
VB_MOT_ReadLockStatus(Status As Long)
```

ASCII-Request

```
%R1Q,6021:
```

ASCII-Response

```
%R1P,0,0:RC,Status[long]
```

Remarks

This function returns the current condition of the LockIn control (see subsystem AUT for further information). This command is valid for TCA instruments only.

Parameters

`Status`	`out`	lock information

Return-Codes

`RC_OK`	Execution successful.
`RC_NOT_IMPL`	No motorisation available (no TCA instrument).

Example

```
RC_TYPE          rc;
MOT_LOCK_STATUS  Status;

rc = MOT_ReadLockStatus(Status)
if (rc == RC_OK)
{
  // use lock status information
}
else
{
  // this is no TCA instrument
}
```

14.3.2 MOT_StartController - Start motor controller

C-Declaration

```
MOT_StartController(MOT_MODE ControlMode)
```

VB-Declaration

```
VB_MOT_StartController(ControlMode As Long)
```

ASCII-Request

`%R1Q,6001:`*ControlMode[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This command is used to enable remote or user interaction to the motor controller.

Parameters

ControlMode	in	Controller mode. If used together with MOT_SetVelocity the control mode has to be MOT_OCONST.

Return-Codes

RC_OK	Execution successful.
RC_IVPARAM	The value of ControlMode is not MOT_OCONST.
RC_NOT_IMPL	No motorization available (no TCA instrument).
MOT_RC_BUSY	Subsystem is busy (e.g. controller already started).
MOT_RC_UNREADY	Subsystem is not initialised.

See Also

MOT_SetVelocity
MOT_StopController

Example

see MOT_SetVelocity

14.3.3 MOT_StopController - Stop motor controller

C-Declaration

```
MOT_StopController(MOT_STOPMODE Mode)
```

VB-Declaration

```
VB_MOT_StopController(Mode As Long)
```

ASCII-Request

%R1Q,6002:*Mode[long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This command is used to stop movement and to stop the motor controller operation.

Parameters

Mode	in	Stop mode

Return-Codes

RC_OK	Execution successful.

MOT_RC_NOT_BUSY	No movement in progress (e.g. stop without start).

See Also

MOT_SetVelocity
MOT_StartController
AUS_SetUserLockState

Example

see MOT_SetVelocity

14.3.4 MOT_SetVelocity - Drive Instrument with visual control

C-Declaration

```
MOT_SetVelocity(MOT_COM_PAIR RefOmega)
```

VB-Declaration

```
VB_MOT_SetVelocity(RefOmega As MOT_COM_PAIR)
```

ASCII-Request

%R1Q,6004:*HZ-Speed[double],V-Speed[double]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This command is used to set up the velocity of motorization. This function is valid only if MOT_StartController(MOT_OCONST) has been called previously. RefOmega[0] denotes the horizontal and RefOmega[1] denotes the vertical velocity setting.

Parameters

RefOmega	in	The speed in horizontal and vertical direction in rad/s. The maximum speed is +/- 0.79 rad/s each.

Return-Codes

RC_OK	Execution successful
RC_IVPARAM	RefOmega.adValue[HZ] and/or RefOmega.adValue[V] values are not within the boundaries.
MOT_RC_NOT_CONFIG	System is not in state MOT_CONFIG or MOT_BUSY_OPEN_END (e.g. missing ‘start controller’).

MOT_RC_NOT_OCONST	Drive is not in mode MOT_OCONST (set by MOT_StartController).
RC_NOT_IMPL	No motorization available (no TCA instrument).

See Also

MOT_StartController
MOT_StopController
AUS_SetUserLockState

Example

```
RC_TYPE         rc;
MOT_COM_PAIR    RefOmega;

// set parameter
RefOmega.adValue[0] = 0.05;
RefOmega.adValue[1] = 0.05;

// stop controller and any possible movements
(void) MOT_StopController(MOT_NORMAL);
// wait at least 5 sec.
wait(5);

// start controller; the only valid mode
// for SetVelocity is MOD_OCONST
rc = MOT_StartController(MOT_OCONST);
if (rc == RC_OK)
{
  rc = MOT_SetVelocity(RefOmega);
  // insert here a time delay or a wait for user
  // action; the movement stops by calling
  // MOT_StopController
}
// stop controller and movements abruptly
rc = MOT_StopController(MOT_SHUTDOWN);

// restart controller with default setting
rc = MOT_StartController(MOT_MANUPOS);
if (rc != RC_OK)
{
  // handle error
}
```

15 SUPERVISOR - SUP

The subsystem 'Supervisor' performs the continuous control of the system (e.g. battery voltage, temperature) and allows to display automatically status information (e.g. system time, battery-, position-, Memory-Card-, and inclination measurement icons as well as local-remote display). It also controls the automatic shutdown mechanism.

15.1 CONSTANTS AND TYPES

On/Off Switch

```
enum ON_OFF_TYPE
{
  OFF = 0,
  ON  = 1
};
```

Automatic Shutdown Mechanism for the System

```
enum SUP_AUTO_POWER
{
  AUTO_POWER_DISABLED = 0,  // deactivate the mechanism
  AUTO_POWER_SLEEP    = 1,  // activate sleep mechanism
  AUTO_POWER_OFF      = 2   // activate shut down
                               mechanism
};
```

System Time

```
typedef long SYSTIME;       // [ms]
```

15.2 FUNCTIONS

15.2.1 SUP_GetConfig - Get power management configuration status

C-Declaration

```
SUP_GetConfig(ON_OFF_TYPE &LowTempOnOff,
              SUP_AUTO_POWER &AutoPower,
              SYSTIME &Timeout)
```

VB-Declaration

```
VB_SUP_GetConfig(LowTempOnOff As Long,
                 AutoPower As Long,
                 Timeout As Long)
```

ASCII-Request

```
%R1Q,14001:
```

ASCII-Response

```
%R1P,0,0:RC,LowTempOnOff[long], AutoPower[long], Timeout[long]
```

Remarks

The returned settings are power off configuration and timing.

Parameters

`LowTempOnOff`	`out`	Current state of the low temperature control flag. If the `state=ON` the device automatically turns off when internal temperature fall short of -24°C.
`AutoPower`	`out`	Current activated shut down mechanism
`Timeout`	`out`	The timeout in ms. After this time the device switches in the mode defined by the value of `AutoPower` when no user activity (press a key, turn the device or communication via GeoCOM) occurs.

Return-Codes

`RC_OK`	Execution always successful.

See Also

`SUP_SetConfig`
`SUP_SwitchLowTempControl`

Example

see `SUP_SetConfig`

15.2.2 SUP_SetConfig - Set power management configuration

C-Declaration

```
SUP_SetConfig(ON_OFF_TYPE LowTempOnOff,
              SUP_AUTO_POWER AutoPower,
              SYSTIME Timeout)
```

VB-Declaration

```
VB_SUP_SetConfig(LowTempOnOff As Long,
                 AutoPower As Long,
                 Timeout As Long)
```

ASCII-Request

`%R1Q,14002:`*LowTempOnOff[long], AutoPower[long], Timeout[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Set the configuration for the low temperature control (`ON|OFF`), the auto power off automatic (`AUTO_POWER_DISABLED|..._SLEEP|..._OFF`) and the corresponding timeout for the auto power off automatic.

Parameters

`LowTempOnOff`	`in`	Switch for the low temperature control. Per default the device automatically turns off when internal temperature fall short of -24°C (`LowTempOnOff = On`).
`AutoPower`	`in`	Defines the behaviour of the power off automatic (default: `AutoPower = AUTO_POWER_SLEEP`).
`Timeout`	`in`	The timeout in ms. After this time the device switches in the mode defined by the value of `AutoPower` when no user activity (press a key, turn the device or communication via GeoCOM) occurs. The default value for `Timeout` is 900000ms = 15 Min.

Return-Codes

`RC_OK`	Execution always successful.

See Also

`SUP_GetConfig`

```
SUP_SwitchLowTempControl
```

Example

```
RC_TYPE                 rc;
ON_OFF_TYPE             LowTempOnOff;
SUP_AUTO_POWER          AutoPower;
SYSTIME                 Timeout;

// get parameter values
rc = SUP_GetConfig (LowTempOnOff,
                    AutoPower,
                    Timeout);


// set new values for parameter
LowTempOnOff   = OFF;
AutoPower      = AUTO_POWER_DISABLED;
Timeout        = 600000;  // =10min

rc = SUP_SetConfig (LowTempOnOff,
                    AutoPower,
                    Timeout);
```

15.2.3 SUP_SwitchLowTempControl - Set low temperature control

C-Declaration

```
SUP_SwitchLowTempControl(ON_OFF_TYPE LowTempOnOff)
```

VB-Declaration

```
VB_SUP_SwitchLowTempControl(LowTempOnOff As Long)
```

ASCII-Request

`%R1Q,14003:`*LowTempOnOff[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

Activate (`ON`) respectively deactivate (`OFF`) the low temperature control.

Parameters

`LowTempOnOff`	`in`	Switch for the low temperature control. Per defaults the device automatically turns off when internal temperature fall short of -24°C.

Return-Codes

RC_OK	Execution always successful.

See Also

SUP_GetConfig
SUP_SetConfig

Example

```
RC_TYPE          rc;

// deactivate the low temperature control

rc = SUP_SwitchLowTempControl(OFF)
```

16 THEODOLITE MEASUREMENT AND CALCULATION - TMC

This module is the central measurement, calculation and geodetic control module of the TPS1100 instrument family. All sensors (angle, distance and compensator) deliver their respective data to this module. All sensor information is used to continuously calculate corrected or uncorrected values for angles, distance and position co-ordinates.

The functions handled by the TMC module are:

Measurement Functions

These functions deliver measurement results. Angle- and inclination measurement are started by system functions directly, other measurement operations needs activating the corresponding sensor (e.g. distance measurement)

Measurement Control Functions

These functions control measurement behaviour (activate/deactivate sensors) and basic data for the calculation of measurement results.

Data Set-up Functions

These functions allow sending destination data, location data and section data to the Theodolite.

Information Functions

These functions return additional information about measurement results, sensors, Theodolite states, etc.

Configuration Functions

These functions control the Theodolite behaviour in general.

The measurement functions of this subsystem generally can generate three types of return codes:

System Return Codes are of general use (`RC_OK` means result is okay,...)

Informative Return code indicates that the function was terminated successfully. But some restrictions apply (e.g. it can be reported that the angle values are okay, the distance is invalid).

Error Return Codes signal a non-successful termination of the function call.

16.1 USAGE

16.1.1 Inclination measurement/correction

The TMC module handles the inclination sensor data and correction. To get exact results (co-ordinates, angles, distances) the inclination of the instrument must be taken into account. In general, there are two ways how this can be done:

Measuring the inclination

Calculating the inclination
For a limited time of several seconds and a limited horizontal angle between 10 and 40 degrees (depending on instrument type) an inclination model is generated to speed up measurement. The model for the inclination is based on the last exact inclination measurement and is maintained within the TMC as a calculated inclination plane.

To control the kind of generating the results, all measurement functions have a parameter (of type `TMC_INCLINE_PRG`), where the inclination mode can be selected. The different measurement modes are:

`TMC_MEA_INC`:
Measures the inclination (in any case). Use this mode by unstable conditions like e.g. the instrument has been moved or walking around the instrument may influence on an unstable underground (e.g. field grass). The disadvantage of this mode is that it is about half a second slower than `TMC_PLANE_INC`.

`TMC_PLANE_INC`:
Calculates the inclination (assumes that the instrument has not been moved). This mode gives an almost immediate result (some milliseconds).

`TMC_AUTO_INC`:
The system decides which method should be used (either `TMC_MEA_INC` or `TMC_PLANE_INC`). You get the best performance regarding measure rate and accuracy with this mode, the instrument checks the conditions around the station. We recommend taking this mode any time.

Note that the results depend on the system's configuration, too. That means that the compensator must be switched on in order to get a result with inclination correction (see `TMC_SetInclineSwitch`). The return code of the measurement functions holds information about the quality of the result. E.g. it is reported, if the compensation of inclination could not be done.

16.1.2 Sensor measurement programs

The instrument supports different measurement programs, which activates or deactivates the sensors in different manner. The programs can be selected by the control function `TMC_DoMeasure` (via the parameter of the type `TMC_MEASURE_PRG`).

Additionally the setting of the EDM measurement mode is set with the function `TMC_SetEdmMode` and influences the measurement. Here a choice between single measurement and continues measurement is possible (each is different in speed and precision).

General measurement programs:

`TMC_DEF_DIST`:
Starts the distance measurement with the set distance measurement program.

`TMC_TRK_DIST`:
Starts the distance measurement in tracking mode.

`TMC_RTRK_DIST`:
Starts the distance measurement in rapid tracking mode.

`TMC_STOP`:
Stops measurement.

`TMC_CLEAR`:
Stops the measurement and clears the data.

`TMC_SIGNAL`:
Help mode for signal intensity measurement (use together with function `TMC_GetSignal`)

`TMC_RED_TRK_DIST:`

Starts the distance tracking measurement with red laser. This mode can be used for reflectorless short distance measurement or long distance measurement with reflector.

16.2 CONSTANTS AND TYPES

On / Off switches

```
enum ON_OFF_TYPE        // on/off switch type
    {
    OFF         = 0,       // Switch is off
    ON          = 1        // Switch is on
    };
```

Inclination Sensor Measurement Program

(see Chapter 16.1.1 for further information)

```
enum TMC_INCLINE_PRG {
  TMC_MEA_INC     = 0,   // Use sensor (apriori sigma)
  TMC_AUTO_INC    = 1,   // Automatic mode (sensor/plane)
  TMC_PLANE_INC   = 2,   // Use plane (apriori sigma)
};
```

TMC Measurement Mode

(see Chapter 16.1.2 for further information)

```
enum TMC_MEASURE_PRG {
  TMC_STOP              = 0,   // Stop measurement program
  TMC_DEF_DIST          = 1,   // Default DIST-measurement
                               // program
  TMC_TRK_DIST          = 2,   // Distance-TRK measurement
                               // program
  TMC_CLEAR             = 3,   // TMC_STOP and clear data
  TMC_SIGNAL            = 4,   // Signal measurement (test
                               // function)
  TMC_DO_MEASURE        = 6,   // (Re)start measurement  task
  TMC_RTRK_DIST         = 8,   // Distance-TRK measurement
                               // program
  TMC_RED_TRK_DIST      = 10,  // Red laser tracking
  TMC_FREQUENCY         = 11   // Frequency measurement (test)
};
```

EDM Measurement Mode

```
enum EDM_MODE {
  EDM_MODE_NOT_USED    = 0, // Init value
  EDM_SINGLE_TAPE      = 1, // Single measurement with tape
  EDM_SINGLE_STANDARD  = 2, // Standard single measurement
  EDM_SINGLE_FAST      = 3, // Fast single measurement
  EDM_SINGLE_LRANGE    = 4, // Long range single measurement
  EDM_SINGLE_SRANGE    = 5, // Short range single  measurement
  EDM_CONT_STANDARD    = 6, // Standard repeated measurement
  EDM_CONT_DYNAMIC     = 7, // Dynamic repeated measurement
  EDM_CONT_REFLESS     = 8, // Reflectorless repeated measurement
  EDM_CONT_FAST        = 9, // Fast repeated measurement
  EDM_AVERAGE_IR       = 10,// Standard average measurement
  EDM_AVERAGE_SR       = 11,// Short range average measurement
  EDM_AVERAGE_LR       = 12  // Long range average measurent
};
```

EDM Frequency

```
typedef struct TMC_EDM_FREQUENCY {
  double  dFrequency; // EDM's frequency in Hz
  SYSTIME Time;       // Time of last measurement
 };
```

Calculated Co-ordinates based on a Distance Measurement

```
struct TMC_COORDINATE {
  double dE;               // E-Coordinate
  double dN;               // N-Coordinate
  double dH;               // H-Coordinate
  SYSTIME CoordTime;       // Moment of dist. measurement
  double dE_Cont;          // E-Coordinate (continuously)
  double dN_Cont;          // N-Coordinate (continuously)
  double dH_Cont;          // H-Coordinate (continuously)
  SYSTIME CoordContTime;   // Moment of measurement [ms]
};
```

Corrected Angle Data

```
struct TMC_HZ_V_ANG {
  double dHz;                // Horizontal angle [rad]
  double dV;                 // Vertical angle [rad]
};
```

Corrected Angle Data with Inclination Data

```
struct TMC_ANGLE {
  double dHz;              // Horizontal angle [rad]
  double dV;               // Vertical angle [rad]
  double dAngleAccuracy;   // Accuracy of angles [rad]
  SYSTIME AngleTime;       // Moment of measurement [ms]
  TMC_INCLINE Incline;     // Corresponding  inclination
  TMC_FACE eFace;          // Face position of telescope
};
```

Offset Values for Correction

```
struct TMC_OFFSETDIST {
  double dLengthVal; // Aim offset length
  double dCrossVal;  // Aim offset cross
  double dHeightVal; // Aim offset height
};
```

Inclination Data

```
struct TMC_INCLINE {
  double dCrossIncline;     // Transverse axis incl. [rad]
  double dLengthIncline;    // Longitud. axis inclination [rad]
  double dAccuracyIncline;  // Inclination accuracy [rad]
  SYSTIME InclineTime;      // Moment of measurement [ms]
};
```

System Time

```
typedef long SYSTIME;         // time since poweron [ms]
```

Face Position

```
enum TMC_FACE_DEF {
  TMC_FACE_NORMAL,          // Face in normal position
  TMC_FACE_TURN             // Face turned
};
```

Actual Face

```
enum TMC_FACE {
  TMC_FACE_1,               // Pos 1 of telescope
  TMC_FACE_2                // Pos 2 of telescope
};
```

Reflector Height

```
struct TMC_HEIGHT {
  double dHr;                // Reflector height
};
```

Atmospheric Correction Data

```
struct TMC_ATMOS_TEMPERATURE {
  double  dLambda;           // Wave length of the EDM transmitter
  double  dPressure;         // Atmospheric pressure
  double  dDryTemperature;   // Dry temperature
```

```
  double  dWetTemperature;  // Wet temperature
};
```

Refraction Control Data

```
struct TMC_REFRACTION {
  ON_OFF_TYPE eRefOn          // Refraction correction On/Off
  double  dEarthRadius;       // Radius of the earth
  double  dRefractiveScale;   // Refractive coefficient
};
```

Instrument Station Co-ordinates

```
struct TMC_STATION {
  double dE0;               // Station easting coordinate
  double dN0;               // Station northing coordinate
  double dH0;               // Station height coordinate
  double dHi;               // Instrument height
};
```

EDM Signal Information

```
struct TMC_EDM_SIGNAL  {
  double  dSignalIntensity; // Signal intensity of EDM in %
  SYSTIME Time;             // Time when measurement was taken
};
```

Correction Switches

```
struct TMC_ANG_SWITCH {
  ON_OFF_TYPE eInclineCorr;        // Inclination correction
  ON_OFF_TYPE eStandAxisCorr;      // Standing axis corr.
  ON_OFF_TYPE eCollimationCorr;    // Collimation error corr.
  ON_OFF_TYPE eTiltAxisCorr;       // Tilting axis corr.
};
```

16.3 MEASUREMENT FUNCTIONS

16.3.1 TMC_GetCoordinate - Gets the coordinates of a measured point

C-Declaration

```
TMC_GetCoordinate(SYSTIME WaitTime,
```

```
                    TMC_COORDINATE &Coordinate,
                    TMC_INCLINE_PRG Mode)
```

VB-Declaration

```
VB_TMC_GetCoordinate1(ByVal WaitTime As Long,
                      Coordinate As TMC_COORDINATE,
                      ByVal Mode As Long)
```

ASCII-Request

`%R1Q,2082:`*WaitTime[long],Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC,E[double],N[double],H[double],CoordTime[long],*
E-Cont[double],N-Cont[double],H-Cont[double],CoordContTime[long]

Remarks

This function issues an angle measurement and, in dependence of the selected `Mode`, an inclination measurement and calculates the co-ordinates of the measured point with an already measured distance. The `WaitTime` is a delay to wait for the distance measurement to finish. Single and tracking measurements are supported. Information about a missing distance measurement and other information about the quality of the result is returned in the return- code.

Parameters

`WaitTime`	`in`	The delay to wait for the distance measurement to finish [ms].
`Coordinate`	`out`	Calculated Cartesian co-ordinates.
`Mode`	`in`	Inclination sensor measurement mode

Return Codes

`RC_OK`	Execution successful.
`TMC_ACCURACY_GUARANTEE`	Accuracy is not guaranteed, because the result are consist of measuring data which accuracy could not be verified by the system. Co-ordinates are available.
`TMC_NO_FULL_CORRECTION`	The results are not corrected by all active sensors. Co-ordinates are available. In order to secure witch correction is missing use the both functions `TMC_IfDataAzeCorrError` and `TMC_IfDataIncCorrError`

TMC_ANGLE_OK	Angle values okay, but no valid distance. Co-ordinates are not available.
TMC_ANGLE_ACCURACY_ GUARANTEE	No distance data available but angle data are valid. The return code is equivalent to the TMC_ACCURACY_GUARANTEE and relates to the angle data. Co-ordinates are not available.
TMC_ANGLE_NO_FULL_ CORRECTION	No distance data available but angle data are valid. The return code is equivalent to the TMC_NO_FULL_CORRECTION and relates to the angle data. Co-ordinates are not available.
	Perform a distance measurement first before you call this function.
TMC_DIST_ERROR	No measuring, because of missing target point, co-ordinates are not available.
	Aim target point and try it again
TMC_DIST_PPM	No distance measurement respectively no distance data because of wrong EDM settings. The co-ordinates are not available. Set EDM -ppm and -mm to 0.
TMC_ANGLE_ERROR	Problems with angle res. incline sensor. A valid angle could not be measured.
	At repeated occur call service.
TMC_BUSY	TMC resource is locked respectively TMC task is busy.
	Repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

```
TMC_DoMeasure
TMC_IfDataAzeCorrError
TMC_IfDataIncCorrError
```

Example

```
RC_TYPE                 Result;
TMC_COORDINATE          Coordinate;
```

```
// make a single distance measurement first
Result=TMC_DoMeasure(TMC_DEF_DIST, TMC_AUTO_INC);

if(Result==RC_OK)
{// before you get the coordinates
     Result=TMC_GetCoordinate(1000,Coordinate,
                              TMC_AUTO_INC);
}

switch(Result)
{// result interpretation
case RC_OK:
     break;
     .
     .
// error handling
case ...:
     .
     .
default:
     break;
}
```

16.3.2 TMC_GetSimpleMea - Returns angle and distance measurement

C-Declaration

```
TMC_GetSimpleMea(SYSTIME WaitTime,
                 TMC_HZ_V_ANG &OnlyAngle,
                 double &SlopeDistance,
                 TMC_INCLINE_PRG Mode)
```

VB-Declaration

```
VB_TMC_GetSimpleMea(ByVal WaitTime As Long,
                    OnlyAngle As TMC_HZ_V_ANG,
                    SlopeDistance As Double,
                    ByVal Mode As Long)
```

ASCII-Request

`%R1Q,2108:`*WaitTime[long],Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC,Hz[double],V[double],SlopeDistance[double]*

Remarks

This function returns the angles and distance measurement data. The distance measurement will be set invalid afterwards. It is important to note that this command does not issue a new distance measurement.

If a distance measurement is valid the function ignores `WaitTime` and returns the results.

If no valid distance measurement is available and the distance measurement unit is not activated (by `TMC_DoMeasure` before the `TMC_GetSimpleMea` call) the `WaitTime` is also ignored and the angle measurement result is returned. So this function can be used instead of `TMC_GetAngle5`.

Information about distance measurement is returned in the return- code.

Parameters

`WaitTime`	`in`	The delay to wait for the distance measurement to finish [ms].
`OnlyAngle`	`out`	Result of the angle measurement.
`SlopeDistance`	`out`	Result of the distance measurement [m].
`Mode`	`in`	Inclination sensor measurement mode.

Return Codes

`RC_OK`	Execution successful.
`TMC_NO_FULL_CORRECTION`	The results are not corrected by all active sensors. Angle and distance data are available. In order to secure witch correction is missing use the both functions `TMC_IfDataAzeCorrError` and `TMC_IfDataIncCorrError` This message is to be considers as warning.
`TMC_ACCURACY_GUARANTEE`	Accuracy is not guaranteed, because the result consisting of measuring data which accuracy could not be verified by the system. Angle and distance data are available. You can a forced incline measurement perform or switch off the incline. This message is to be considers as info.

TMC_ANGLE_OK	Angle values okay, but no valid distance. Perform a distance measurement.
TMC_ANGLE_NO_FULL_CORRECTION	No distance data available but angle data are valid. The return code is equivalent to the TMC_NO_FULL_CORRECTION and relates to the angle data. Perform a distance measurement first before you call this function.
TMC_ANGLE_ACCURACY_GUARANTEE	No distance data available but angle data are valid. The return code is equivalent to the TMC_ACCURACY_GUARANTEE and relates to the angle data.
TMC_DIST_ERROR	No measuring, because of missing target point, angle data are available but distance data are not available. Aims target point and try it again.
TMC_DIST_PPM	No distance measurement respectively no distance data because of wrong EDM settings. Angle data are available but distance data are not available. Set EDM -ppm and -mm to 0.
TMC_ANGLE_ERROR	Problems with angle res. incline sensor. A valid angle could not be measured. Distance and angle data are not available. At repeated occur call service.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. Distance and angle data are not available. Repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

TMC_DoMeasure
TMC_GetAngle5

Example

```
RC_TYPE         rc;
TMC_HZ_V_ANG    OnlyAngle;
```

```
double          SlopeDistance;

// activate distance measurement
rc = TMC_DoMeasure(TMC_DEF_DIST, TMC_AUTO_INC);
if (rc == RC_OK)
{
  // distance measurement successful
  rc = TMC_GetSimpleMea(3000, OnlyAngle,
                        SlopeDistance, TMC_MEA_INC);

  if (rc == RC_OK)
  {
    // use distance and angle values
  else
  {
    // something with TMC_GetSimpleMea went wrong
  }
else
{
  // something with dist. measurement went wrong
}
```

16.3.3 TMC_GetAngle1 - Returns complete angle measurement

C-Declaration

```
TMC_GetAngle(TMC_ANGLE &Angle,
             TMC_INCLINE_PRG Mode)
```

VB-Declaration

```
VB_TMC_GetAngle1(Angle As TMC_ANGLE,
                 ByVal Mode As Long)
```

ASCII-Request

`%R1Q,2003:`*Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC,Hz[double],V[double],AngleAccuracy[double], AngleTime[long],CrossIncline[double],LengthIncline[double], AccuracyIncline[double],InclineTime[long],FaceDef[long]*

Remarks

This function carries out an angle measurement and, in dependence of configuration, inclination measurement and returns the results. As shown the result is very comprehensive. For simple angle measurements use `TMC_GetAngle5` or `TMC_GetSimpleMea` instead.

Information about measurement is returned in the return code.

Parameters

Angle	out	Result of the angle measurement.
Mode	in	Inclination sensor measurement mode.

Return Codes

RC_OK	Execution successful.
TMC_NO_FULL_CORRECTION	The results are not corrected by all active sensors. Angle data are available. In order to secure witch correction is missing use the both functions TMC_IfDataAzeCorrError and TMC_IfDataIncCorrError This message is to be considers as warning.
TMC_ACCURACY_GUARANTEE	Accuracy is not guaranteed, because the result consisting of measuring data which accuracy could not be verified by the system. Angle data are available. You can a forced incline measurement perform or switch off the incline. This message is to be considers as info.
TMC_ANGLE_ERROR	Problems with angle res. incline sensor. A valid angle could not be measured. Angle data are not available. At repeated occur call service.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. Angle data are not available. Repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

TMC_DoMeasure
TMC_GetAngle5
TMC_GetSimpleMea

Example

```
see TMC_GetAngle5
```

16.3.4 TMC_GetAngle5 - Returns simple angle measurement

C-Declaration

```
TMC_GetAngle(TMC_HZ_V_ANG &OnlyAngle,
             TMC_INCLINE_PRG Mode)
```

VB-Declaration

```
VB_TMC_GetAngle5(OnlyAngle As TMC_HZ_V_ANG,
                 ByVal Mode As Long)
```

ASCII-Request

`%R1Q,2107:`*Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC,Hz[double],V[double]*

Remarks

This function carries out an angle measurement and returns the results. In contrast to the function `TMC_GetAngle1` this function returns only the values of the angle. For simple angle measurements use or `TMC_GetSimpleMea` instead.

Information about measurement is returned in the return code.

Parameters

`Angle`	`out`	Result of the angle measurement.
`Mode`	`in`	Inclination sensor measurement mode.

Return Codes

`RC_OK`	Execution successful.
`TMC_NO_FULL_CORRECTION`	The results are not corrected by all active sensors. Angle data are available. In order to secure witch correction is missing use the both functions `TMC_IfDataAzeCorrError` and

	TMC_IfDataIncCorrError This message is to be considers as warning.
TMC_ACCURACY_GUARANTEE	Accuracy is not guaranteed, because the result consisting of measuring data which accuracy could not be verified by the system. Angle data are available. You can a forced incline measurement perform or switch off the incline. This message is to be considers as info.
TMC_ANGLE_ERROR	Problems with angle res. incline sensor. A valid angle could not be measured. Angle data are not available. At repeated occur call service.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. Angle data are not available. Repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

TMC_DoMeasure
TMC_GetAngle5
TMC_GetSimpleMea

Example

```
RC_TYPE         Result;
TMC_ANGLE       Angle;
BOOLE           bExit,
                bAzeCorrError,
                bIncCorrError;
short           nCnt;

nCnt=0;
do
{
bExit=TRUE;

// Gets the whole angle data
Result=TMC_GetAngle(Angle, TMC_AUTO_INC);
```

```
switch(Result)
{
case RC_OK:
     // Execution successful
     break;
case TMC_NO_FULL_CORRECTION:
     TMC_IfDataAzeCorrError(bAzeCorrError);
     TMC_IfDataIncCorrError(bIncCorrError);
     if(bAzeCorrError)
     {
     // coordinates are not corrected with the Aze-
     // deviation correction
     }
     if(bIncCorrError)
     {
     // coordinates are not corrected with the
     // incline correction
     }
     break;
case TMC_ACCURACY_GUARANTEE:
     // perform a forced incline measurement,
     // see example TMC_QuickDist
     break;

case TMC_BUSY:
     // repeat measurement
     bExit=FALSE;
case RC_ABORT:
case RC_SHUT_DOWN:
default:
     break;
}// end switch

nCnt++;
}while(!bExit && nCnt<3);
```

16.3.5 TMC_QuickDist - Returns slope-distance and hz-,v-angle

C-Declaration

```
TMC_QuickDist(   TMC_HZ_V_ANG &OnlyAngle,
                 double       &dSlopeDistance)
```

VB-Declaration

```
VB_TMC_QuickDist(       OnlyAngle       As
                        TMC_HZ_V_ANG,
                        dSlopeDistance  As Double)
```

ASCII- Request

```
%R1Q,2117:
```

ASCII-Response

```
%R1P,0,0:RC,dHz[double],dV[double],dSlopeDistance[double]
```

Remarks

The function waits until a new distance is measured and then it returns the angle and the slope-distance, but no co-ordinates. Is no distance available, then it returns the angle values (hz, v) and the corresponding return-code.

At the call of this function, a distance measurement will be started with the rapid-tracking measuring program. If the EDM is already active with the standard tracking measuring program, the measuring program will not changed to rapid tracking. Generally if the EDM is not active, then the rapid tracking measuring program will be started, otherwise the used measuring program will not be changed.

In order to abort the current measuring program use the function `TMC_DoMeasure`.

This function is very good suitable for target tracking, where high data transfers are required.

Note: Due to performance reasons the used inclination will be calculated (only if incline is activated), so the basic data for the incline calculation is exact, at least two forced incline measurements should be performed in between. The forced incline measurement is only necessary if the incline of the instrument because of measuring assembly has been changed.
Use the function `TMC_GetAngle(TMC_MEA_INC, Angle)` for the forced incline measurement. (For the forced incline measurement, the instrument must be in stable state for more than 3sec.).

Parameters

`OnlyAngle`	`out`	measured Hz- and V- angle
`dSlopeDistance`	`out`	measured slope-distance

Return-Codes

`RC_OK`	Measurement ok. Angle and distance data are available.

TMC_NO_FULL_ CORRECTION	The results are not corrected by all active sensors. Angle and distance data are available. In order to secure witch correction is missing use the both functions TMC_IfDataAzeCorrError and TMC_IfDataIncCorrError. This message is to be considers as warning.
TMC_ACCURACY_ GUARANTEE	Accuracy is not guaranteed, because the result consisting of measuring data which accuracy could not be verified by the system. Angle and distance data are available. You can perform a forced incline measurement or switch off the incline. This message is to be considers as info.
TMC_ANGLE_OK	Angle measuring data are valid, but no distance data are available. (Possible reasons are: -time out period to short -target out of view) This message is to be considers as warning.
TMC_ANGLE_NO_ FULL_ CORRECTION	Angle measuring data are valid, but not corrected by all active sensors. The distance data are not available. (Possible reasons are: -see return code TMC_ANGLE_OK) This message is to be considers as warning.
TMC_ANGLE_ ACCURACY_ GUARANTEE	Angle measuring data are valid, but the accuracy is not guarantee, because the result (angle) consisting of measuring data, which accuracy could not be verified by the system. The distance data are not available. (Possible reasons are: -see return code TMC_ANGLE_OK) This message is to be considers as info.
TMC_DIST_ERROR	Because of missing target point no distance data are available, but the angle data are valid respectively available. Aim target point and try it again.
TMC_DIST_PPM	No distance measurement respectively no distance

	data because of wrong EDM settings. The angle data are valid. Set EDM -ppm and -mm to 0.
TMC_ANGLE_ERROR	Problems with angle res. incline sensors. At repeated occur call service.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. Repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

TMC_GetAngle
TMC_DoMeasure
TMC_IfDataAzeCorrError
TMC_IfDataIncCorrError

Example

```
const short     MAX=100;// number of measurements
const double    STATIC_TIME=4.0;// in seconds
const double    MAX_DIFFERENCE=0.0002// in rad
RC_TYPE         Result;
TMC_ANG_SWITCH  SwCorr;
TMC_HZ_V_ANG    HzVAng;
TMC_ANGLE       AngleDummy;
BOOLE           bExit;
DATIME          Datime;
double          dSlopeDist,
                dLastHzAng,
                dhz_angle_diff,
                dact_time, dstart_time;
short           nNoMeasurements;


TMC_GetAngSwitch(SwCorr);

SwCorr.eInclineCorr=ON;    // measure rate will be
SwCorr.eStandAxisCorr=ON;  // reduced if angle and
SwCorr.eCollimationCorr=ON;// incline correction are
SwCorr.eTiltAxisCorr=ON;   // activated

TMC_DoMeasure(TMC_CLEAR);  // clear distance first
TMC_SetAngSwitch(SwCorr);  // before you can set the
```

```
                              // ANG switches, the
                              // distance must be
                              // cleared

CSV_GetDateTime(Datime);
dstart_time=Datime.Time.Minute*60+
              Datime.Time.Second;

// starts the rapid tracking dist. measurement program
TMC_QuickDist(HzVAng, dSlopeDist);

bExit=FALSE;
nNoMeasurements=0;
do
{
     dLastHzAng=HzVAng.dHz;
     Result=TMC_QuickDist(HzVAng, dSlopeDist);
     switch(Result)
     {
     // distance- and angles- data available
     case TMC_ACCURACY_GUARANTEE:
          // perform a forced incline measurement

          // caution: the calculation at zero rad is
          // not consider
          dhz_angle_diff=fabs(dLastHzAng-
                              HzVAng.dHz);

          if(dhz_angle_diff<MAX_DIFFERENCE)
          {// instrument is in static period
               CSV_GetDateTime(Datime);
               dact_time=Datime.Time.Minute*60+
                         Datime.Time.Second;

               if(dact_time-dstart_time > STATIC_TIME)
               {// static mode exceeding 3-4 sec
                         TMC_GetAngle(TMC_MEA_INC,
                                      AngleDummy);
                         TMC_GetAngle(TMC_MEA_INC,
                                      AngleDummy);
               }
          }
          else
          {// instrument is not in static period
               CSV_GetDateTime(Datime);
               dstart_time=Datime.Time.Minute*60+
```

```
                                    Datime.Time.Second;
            }

        case RC_OK:
        case TMC_NO_FULL_CORRECTION:
            break;

        // no distance data available
        case TMC_ANGLE_OK:
        case TMC_ANGLE_NO_FULL_CORRECTION:
        case TMC_ANGLE_ACCURACY_GUARANTEE:
        case TMC_DIST_ERROR:
        case TMC_DIST_PPM:
            break;

        // neither angle- nor distance- data available
        case TMC_ANGLE_ERROR:
        case RC_BUSY:
        case RC_ABORT:
        case RC_SHUT_DOWN:

        default:
            bExit=TRUE;
            break;
        }
    }
    while(!bExit && nNoMeasurements<MAX);

    TMC_DoMeasure(TMC_STOP);// stop measureprogram
```

16.4 MEASUREMENT CONTROL FUNCTIONS

16.4.1 TMC_DoMeasure - Carries out a distance measurement

C-Declaration

```
TMC_DoMeasure(TMC_MEASURE_PRG Command,
              TMC_INCLINE_PRG Mode)
```

VB-Declaration

```
VB_TMC_DoMeasure(ByVal Command As Long,
                 ByVal Mode As Long)
```

ASCII-Request

%R1Q,2008:*Command[long],Mode[long]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This function carries out a distance measurement in a variety of TMC measurement modes like single distance, rapid tracking,... . Please note that this command does not output any values (distances). In order to get the values you have to use other measurement functions such as TMC_GetCoordinate, TMC_GetSimpleMea or TMC_GetAngle.

The value of the distance measured is kept in the instrument up to the next TMC_DoMeasure command where a new distance is requested or the distance is clear by the measurement program TMC_CLEAR.

Note: If you perform a distance measurement with the measure program TMC_DEF_DIST, the distance sensor will be work with the set EDM mode, see TMC_SetEdmMode.

Parameters

Command	in	TMC measurement mode.
Mode	in	Inclination sensor measurement mode.

Return-Codes

RC_OK	Execution successful.

See Also

TMC_SetEdmMode
TMC_GetCoordinate
TMC_GetSimpleMea
TMC_GetAngle1
TMC_GetAngle5

Example

```
RC_TYPE Result;
short   nCnt;

// set average mode
Result=TMC_SetEdmMode(EDM_CONT_EXACT);
// perform a single distance measurement
Result=TMC_DoMeasure(TMC_DEF_DIST);

nCnt=0;
while(nCnt<100)
{// wait on the distance data max. 100x100ms
```

```
        Result=TMC_GetCoordinate(100,Coordinate,
                                          TMC_AUTO_INC);
        nCnt++;
    }

    // to complete the measurement, and clear data
    TMC_DoMeasure(TMC_CLEAR);
    // set standard mode
    TMC_SetEdmMode(EMD_SINGLE_STANDARD);
```

16.4.2 TMC_SetHandDist - Input slope distance and height offset

C-Declaration

```
TMC_SetHandDist(double SlopeDistance,
                double HgtOffset,
                TMC_INCLINE_PRG Mode)
```

VB-Declaration

```
VB_TMC_SetHandDist(ByVal SlopeDistance As Double,
                   ByVal HgtOffset As Double,
                   ByVal Mode As Long)
```

ASCII-Request

`%R1Q,2019:`*SlopeDistance[double],HgtOffset[double],Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function is used to input manually measured slope distance and height offset for a following measurement. Additionally an inclination measurement and an angle measurement are carried out to determine the co-ordinates of target. The V-angle is corrected to $\pi/2$ or $3\cdot\pi/2$ in dependence of the instrument's face because of the manual input.

After the function call the previous measured distance is cleared.

Parameters

`SlopeDistance`	`in`	Slope distance
`HgtOffset`	`in`	Height offset
`Mode`	`in`	Inclination sensor measurement mode.

Return Codes

`RC_OK`	Execution successful.

TMC_NO_FULL_CORRECTION	The results are not corrected by all active sensors. In order to secure witch correction is missing use the both functions TMC_IfDataAzeCorrError and TMC_IfDataIncCorrError This message is to be considers as warning.
TMC_ACCURACY_GUARANTEE	Accuracy is not guaranteed, because the result consisting of measuring data which accuracy could not be verified by the system. You can a forced incline measurement perform or switch off the incline. This message is to be considers as info.
TMC_ANGLE_ERROR	Problems with angle res. incline sensor. A valid angle could not be measured. At repeated occur call service.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. Repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.
RC_IVPARAM	Invalid parameter

See Also

TMC_IfDataAzeCorrError
TMC_IfDataIncCorrError

Example

```
RC_TYPE                    rc;
TMC_COORDINATE             Coordinate

rc = VB_TMC_SetHandDist(10, 1, TMC_AUTO_INC)
if (rc == RC_OK)
{
  // calculate coordinates
```

```
    rc=TMC_GetCoordinate(1000,Coordinate,TMC_AUTO_INC)
    if (rc == RC_OK)
    {
      // use coordinates
    else
    {
      // something went wrong
    }
  }
```

16.5 DATA SETUP FUNCTIONS

16.5.1 TMC_GetHeight - Returns the current reflector height

C-Declaration

```
TMC_GetHeight(TMC_HEIGHT &Height)
```

VB-Declaration

```
VB_TMC_GetHeight(Height As TMC_HEIGHT)
```

ASCII-Request

```
%R1Q,2011:
```

ASCII-Response

%R1P,0,0:*RC,Height[double]*

Remarks

This function returns the current reflector height.

Parameters

Height	out	current reflector height

Return Codes

RC_OK	Execution always successful.

See Also

```
TMC_SetHeight
```

Example

```
RC_TYPE         rc;
TMC_HEIGHT      Height, NewHeight;

// reset reflector height to 0
// if it is not already
```

```
rc = TMC_GetHeight(Height);
if (Height.dHr != 0)
{
  NewHeight.dHr = 0;
  rc = TMC_SetHeight(NewHeight);
  if (rc == RC_OK)
  {
    // set of height successful
  }
  else
  {
    // TMC is busy, no set possible
  }
}
```

16.5.2 TMC_SetHeight - Sets new reflector height

C-Declaration

`TMC_SetHeight(TMC_HEIGHT Height)`

VB-Declaration

`VB_TMC_SetHeight(ByVal Height As TMC_HEIGHT)`

ASCII-Request

`%R1Q,2012:`*Height[double]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function sets a new reflector height.

Parameters

`Height`	`in`	new reflector height

Return Codes

`RC_OK`	Execution successful (new height is set).
`TMC_BUSY`	TMC resource is locked respectively TMC task is busy. The reflector height is not set. Repeat measurement.

See Also

`TMC_GetHeight`

Example

see TMC_GetHeight

16.5.3 TMC_GetAtmCorr - Get atmospheric correction parameters

C-Declaration

```
TMC_GetAtmCorr
        (TMC_ATMOS_TEMPERATURE &AtmTemperature)
```

VB-Declaration

```
VB_TMC_GetAtmCorr
        (AtmTemperature As TMC_ATMOS_TEMPERATURE)
```

ASCII-Request

```
%R1Q,2029:
```

ASCII-Response

%R1P,0,0:*RC,Lambda[double],Pressure[double],*
DryTemperature[double],WetTemperature[double]

Remarks

This function is used to get the parameters for the atmospheric correction.

Parameters

AtmTemperature	out	Atmospheric Correction Data

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetAtmCorr

Example

see TMC_SetAtmCorr

16.5.4 TMC_SetAtmCorr - Set atmospheric correction parameters

C-Declaration

```
TMC_SetAtmCorr
        (TMC_ATMOS_TEMPERATURE AtmTemperature)
```

VB-Declaration

```
VB_TMC_SetAtmCorr
```

```
(ByVal AtmTemperature As TMC_ATMOS_TEMPERATURE)
```

ASCII-Request

%R1Q,2028:*Lambda[double],Pressure[double],*
DryTemperature[double],WetTemperature[double]

ASCII-Response

%R1P,0,0:*RC,*

Remarks

This function is used to set the parameters for the atmospheric correction.

Parameters

AtmTemperature	in	Atmospheric Correction Data

Return Codes

RC_OK	Execution successful (new atmospheric correction data are set).

See Also

TMC_GetAtmCorr

Example

```
TMC_ATMOS_TEMPERATURE AtmCorr;

TMC_GetAtmCorr(AtmCorr);

// set new wet and dry temperature
AtmCorr.dDryTemperature=60;
AtmCorr.dWetTemperature=80;

TMC_SetAtmCorr(AtmCorr);
```

16.5.5 TMC_SetOrientation - Orients the theodolite in Hz direction

C-Declaration

```
TMC_SetOrientation(double HzOrientation)
```

VB-Declaration

```
VB_TMC_SetOrientation(ByVal HzOrientation As Double)
```

ASCII-Request

%R1Q,2113:*HzOrientation[double]*

ASCII-Response

%R1P,0,0:*RC*

Remarks

This function is used to orientates the instrument in Hz direction. It is a combination of an angle measurement to get the Hz offset and afterwards setting the angle Hz offset in order to orientates onto a target. Before the new orientation can be set an existing distance must be cleared (use `TMC_DoMeasure` with the `command = TMC_CLEAR`).

Parameters

`HzOrientation`	`in`	Hz Orientation [rad]

Return Codes

`RC_OK`	Execution successful.
`TMC_NO_FULL_CORRECTION`	The orientation is set but not corrected by all active sensors. In order to secure witch correction is missing use the both functions `TMC_IfDataAzeCorrError` and `TMC_IfDataIncCorrError` This message is to be considers as warning.
`TMC_ACCURACY_GUARANTEE`	The orientation is set but the accuracy is not guarantee, because the result consisting of measuring data which accuracy could not be verified by the system You can a forced incline measurement perform or switch off the incline. This message is to be considers as info.
`TMC_ANGLE_ERROR`	Problems with angle res. incline sensor. A valid angle could not be measured. The orientation is not set. At repeated occur call service.
`TMC_BUSY`	TMC resource is locked respectively TMC task is busy or a distance is existing.

	The orientation is not set.
	Clear distance and repeat measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

TMC_IfDataAzeCorrError
TMC_IfDataIncCorrError
TMC_DoMeasure

Example

```
RC_TYPE Result;

// clear existing distance first
TMC_DoMeasure(TMC_CLEAR);
// set orientation to 0
Result=TMC_SetOrientation(0.0);
if(Result!=RC_OK)
{
// error or warning handling
}
```

16.5.6 TMC_GetPrismCorr - Get the prism constant

C-Declaration

```
TMC_GetPrismCorr(double &PrismCorr)
```

VB-Declaration

```
VB_TMC_GetPrismCorr(PrismCorr As Double)
```

ASCII-Request

```
%R1Q,2023:
```

ASCII-Response

```
%R1P,0,0:RC,PrismCorr[double]
```

Remarks

This function is used to get the prism constant.

Parameters

PrismCorr	out	Prism constant [mm]

Return Codes

RC_OK	Execution always successful.

See Also

`TMC_SetPrismCorr`

Example

```
const double Corr = 0.1;
RC_TYPE    rc;
double     PrismCorr;

// set the prism constant to
// 0.1 if not already set

rc = TMC_GetPrismCorr(PrismCorr);
if (PrismCorr != Corr)
{
  rc = TMC_SetPrismCorr(Corr);
  if (rc == RC_OK)
  {
    // set of prisma corr successful
  }
  else
  {
    // Invalid parameter
  }
}
```

16.5.7 TMC_SetPrismCorr - Set the prism constant

C-Declaration

`TMC_SetPrismCorr(double PrismCorr)`

VB-Declaration

`VB_TMC_SetPrismCorr(ByVal PrismCorr As Double)`

ASCII-Request

`%R1Q,2024:`*PrismCorr[double]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function is used to set the prism constant.

The high-level function `BAP_SetPrismType` overwrites this setting.

Parameters

PrismCorr	in	Prism constant [mm]

Return Codes

RC_OK	Execution successful.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. The prism constant is not set. Repeat measurement.

See Also

TMC_GetPrismCorr

Example

see TMC_GetPrismCorr

16.5.8 TMC_GetRefractiveCorr - Get the refraction factor

C-Declaration

```
TMC_GetRefractiveCorr(TMC_REFRACTION &Refractive)
```

VB-Declaration

```
VB_TMC_GetRefractiveCorr
                    (Refractive As TMC_REFRACTION)
```

ASCII-Request

```
%R1Q,2031:
```

ASCII-Response

```
%R1P,0,0:RC,RefOn[boolean],EarthRadius[double], RefractiveScale[double]
```

Remarks

This function is used to get the refraction distortion factor for correction of measured height difference.

Parameters

Refractive	out	Refraction distortion

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetRefractiveCorr

Example

```
const double              EarthRadius = 6378000;
RC_TYPE                   rc;
TMC_REFRACTION  Refractive;

// check the earth radius setting
// and reset if necessary
rc = TMC_GetRefractiveCorr(Refractive);
if (Refractive.dEarthRadius != EarthRadius)
{
  Refractive.dEarthRadius = EarthRadius;
  rc = TMC_SetRefractiveCorr(Refractive);
  if (rc == RC_OK)
  {
    // set of earth radius successful
  }
  else
  {
    // set not successful (subsystem busy)
  }
}
```

16.5.9 TMC_SetRefractiveCorr - Set the refraction factor

C-Declaration

```
TMC_SetRefractiveCorr(TMC_REFRACTION Refractive)
```

VB-Declaration

```
VB_TMC_SetRefractiveCorr
                  (ByVal Refractive As TMC_REFRACTION)
```

ASCII-Request

`%R1Q,2030:` *RefOn[boolean],EarthRadius[double], RefractiveScale[double]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function is used to set the refraction distortion factor for correction of measured height difference.

Parameters

`Refractive`	`in`	Refraction distortion

Return Codes

RC_OK	Execution successful.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. The refraction distortion factor is not set. Repeat measurement.

See Also

TMC_GetRefractiveCorr

Example

see TMC_GetRefractiveCorr

16.5.10 TMC_GetRefractiveMethod - Get the refraction model

C-Declaration

```
TMC_GetRefractiveMethod(unsigned short &Method)
```

VB-Declaration

```
VB_TMC_GetRefractiveMethod(Method As Integer)
```

ASCII-Request

```
%R1Q,2091:
```

ASCII-Response

```
%R1P,0,0:RC,Method[unsigned short]
```

Remarks

This function is used to get the current refraction model.

Parameters

Method	out	Refraction data: Method = 1 means method 1 (for Australia) Method = 2 means method 2 (for the rest of the world)

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetRefractiveMethod

Example

```
const unsigned short      RefractiveMethod = 1;
```

```
RC_TYPE                    rc;
unsigned short             Method;

// set the refractive methode to 1
// if it is not already

rc = TMC_GetRefractiveMethod(Method);
if (Method != RefractiveMethod)
{
  rc = TMC_SetRefractiveMethod(RefractiveMethod);
  if (rc == RC_OK)
  {
    // set of refractive methode successful
  }
  else
  {
    // set not successful (subsystem busy)
  }
}
```

16.5.11 TMC_SetRefractiveMethod - Set the refraction model

C-Declaration

`TMC_SetRefractiveMethod(unsigned short Method)`

VB-Declaration

`VB_TMC_SetRefractiveMethod(ByVal Method As Integer)`

ASCII-Request

`%R1Q,2090:`*Method[unsigned short]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function is used to set the refraction model.

Parameters

`Method`	`in`	Refraction data: `Method` = 1 means method 1 (for Australia) `Method` = 2 means method 2 (for the rest of the world)

Return Codes

RC_OK	Execution successful.
TMC_BUSY	TMC resource is locked respectively TMC task is busy. The refraction model is not set. Repeat measurement.

See Also

TMC_GetRefractiveMethod

Example

see TMC_GetRefractiveMethod

16.5.12 TMC_GetStation - Get the coordinates of the instrument station

C-Declaration

```
TMC_GetStation(TMC_STATION &Station)
```

VB-Declaration

```
VB_TMC_GetStation(Station As TMC_STATION)
```

ASCII-Request

```
%R1Q,2009:
```

ASCII-Response

```
%R1P,0,0:RC,E0[double],N0[double],H0[double],Hi[double]
```

Remarks

This function is used to get the co-ordinates of the instrument station.

Parameters

Station	out	Instrument station co-ordinates.

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetStation

Example

```
RC_TYPE         rc;
TMC_STATION Station, NullStation;
NullStation.dE0 = 0;
NullStation.dN0 = 0;
```

```
NullStation.dH0 = 0;
NullStation.dHi = 0;

// reset station coordinates to 0

rc = TMC_GetStation(Station);
if ((Station.dE0 != 0)||
    (Station.dN0 != 0)||
    (Station.dH0 != 0)||
    (Station.dHi != 0))
{
rc = TMC_SetStation(NullStation);
  if (rc == RC_OK)
  {
    // reset of station successful
  }
  else
  {
    // reset not successful (subsystem busy)
  }
}
```

16.5.13 TMC_SetStation - Set the coordinates of the instrument station

C-Declaration

`TMC_SetStation(TMC_STATION Station)`

VB-Declaration

`VB_TMC_SetStation(ByVal Station As TMC_STATION)`

ASCII-Request

`%R1Q,2010:`*E0[double],N0[double],H0[double],Hi[double]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function is used to set the co-ordinates of the instrument station.

Parameters

`Station`	`in`	Instrument station co-ordinates.

Return Codes

`RC_OK`	Execution successful.
`TMC_BUSY`	TMC resource is locked respectively

	TMC task is busy or a distance is existing. The instrument co-ordinates are not set.
	Clear distance and repeat measurement.

See Also

`TMC_GetStation`
`TMC_DoMeasure`

Example

see `TMC_GetStation`

16.6 INFORMATION FUNCTIONS

16.6.1 TMC_GetFace - Get face information of current telescope position

C-Declaration

```
TMC_GetFace(TMC_FACE &Face)
```

VB-Declaration

```
VB_TMC_GetFace(Face As Long)
```

ASCII-Request

```
%R1Q,2026:
```

ASCII-Response

`%R1P,0,0:`*RC,Face[long]*

Remarks

This function returns the face information of the current telescope position. The face information is only valid, if the instrument is in an active measurement state (that means a measurement function was called before the `TMC_GetFace` call, see example). Note that the instrument automatically turns into an inactive measurement state after a predefined timeout.

Parameters

`Face`	`out`	Face position.

Return Codes

`RC_OK`	Execution always successful.

See Also

`AUT_ChangeFace`

Example

```
RC_TYPE         rc;
TMC_FACE   Face;

// turn the face if not in normal position

// set active measurement state
rc = TMC_DoMeasure(TMC_DEF_DIST, TMC_AUTO_INC);
rc = TMC_GetFace(Face);
if (Face == TMC_FACE_TURN)
{
  rc = AUT_ChangeFace(AUT_NORMAL,
                      AUT_POSITION,
                      FALSE);
  if (rc == RC_OK)
  {
    // face successfuly turned
  }
  else
  {
    // change face problem: see AUT_ChangeFace
  }
}
// clear distance
rc = TMC_DoMeasure(TMC_CLEAR, TMC_AUTO_INC);
```

16.6.2 TMC_GetSignal - Get information about EDM's signal amplitude

C-Declaration

```
TMC_GetSignal(TMC_EDM_SIGNAL &Signal)
```

VB-Declaration

```
VB_TMC_GetSignal(Signal As TMC_EDM_SIGNAL)
```

ASCII-Request

```
%R1Q,2022:
```

ASCII-Response

```
%R1P,0,0:RC,SignalIntensity[double],Time[long]
```

Remarks

This function returns information about the amplitude of the EDM signal. The function only can perform measuring if the signal measurement program is activated. Start the signal measurement program with

TMC_DoMeasure where Command = TMC_SIGNAL. After the measurement the EDM must be switch off (use TMC_DoMeasure where Command = TMC_CLEAR).

Parameters

Face	out	Face position.

Return Codes

RC_OK	Execution successful.
TMC_SIGNAL_ERROR	Error within signal measurement. At repeated occur call service.
RC_IVPARAM	The signal measurement program is not activated. No signal measurement possible. Activate signal measurement.
RC_ABORT	Measurement through customer aborted.
RC_SHUT_DOWN	System power off through customer.

See Also

TMC_DoMeasure

Example

```
RC_TYPE Result;
TMC_SIGNAL Signal;

TMC_DoMeasure(TMC_SIGNAL);
do
{
     Result=TMC_GetSignal(Signal);
     if(Result==RC_OK)
     {
          .
          .
          .
     }
}while(Result==RC_OK);
```

16.7 CONFIGURATION FUNCTIONS

16.7.1 TMC_GetAngSwitch - Get angular correction's states

C-Declaration

```
TMC_GetAngSwitch(TMC_ANG_SWITCH &SwCorr)
```

VB-Declaration

```
VB_TMC_GetAngSwitch(SwCorr As TMC_ANG_SWITCH)
```

ASCII-Request

```
%R1Q,2014:
```

ASCII-Response

```
%R1P,0,0:RC,InclineCorr[long],StandAxisCorr[long],
    CollimationCorr[long],TiltAxisCorr[long]
```

Remarks

This function returns the angular correction's state.

Parameters

SwCorr	out	Angular correction's state.

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetAngSwitch

Example

```
RC_TYPE         rc;
TMC_ANG_SWITCH  SwCorr;

// get the switch state for the angular
// correction

rc = TMC_GetAngSwitch(SwCorr);
if (SwCorr.eTiltAxisCorr == ON)
{
  // Tilting axis correction turned On
}
else
{
  // Tilting axis correction turned Off
}
```

16.7.2 TMC_GetInclineSwitch - Get the dual axis compensator's state

C-Declaration

```
TMC_GetInclineSwitch(ON_OFF_TYPE  &SwCorr)
```

VB-Declaration

```
VB_TMC_GetInclineSwitch(SwCorr As Long)
```

ASCII-Request

```
%R1Q,2007:
```

ASCII-Response

```
%R1P,0,0:RC,SwCorr[long]
```

Remarks

This function returns the current dual axis compensator's state.

Parameters

SwCorr	out	Dual axis compensator's state.

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetInclineSwitch

Example

```
RC_TYPE         rc;
ON_OFF_TYPE     SwCorr;

// clear distance first before you change the state
TMC_DoMeasure(TMC_CLEAR, TMC_AUTO,INC);

// deactivate the compensator
// if it is not already

rc = TMC_GetInclineSwitch(SwCorr);
if (SwCorr == ON)
{
  rc = TMC_SetInclineSwitch(OFF);
  if (rc == RC_OK)
  {
    // successfully deactvated
  }
  else
  {
    // set not successful (subsystem busy)
```

```
  }
}
```

16.7.3 TMC_SetInclineSwitch - Switch dual axis compensator on or off

C-Declaration

```
TMC_SetInclineSwitch(ON_OFF_TYPE  SwCorr)
```

VB-Declaration

```
VB_TMC_SetInclineSwitch(ByVal SwCorr As Long)
```

ASCII-Request

`%R1Q,2006:`*SwCorr[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function switches the dual axis compensator on or off.

Parameters

`SwCorr`	`in`	Dual axis compensator's state.

Return Codes

`RC_OK`	Execution successful.
`TMC_BUSY`	TMC resource is locked respectively TMC task is busy or a distance is existing. The incline state is not changed. Clear distance and repeat measurement.

See Also

`TMC_GetInclineSwitch`

Example

`see TMC_GetInclineSwitch`

16.7.4 TMC_GetEdmMode - Get the EDM measurement mode

C-Declaration

```
TMC_GetEdmMode(EDM_MODE &Mode)
```

VB-Declaration

```
VB_TMC_GetEdmMode(Mode As Long)
```

ASCII-Request

```
%R1Q,2021:
```

ASCII-Response

```
%R1P,0,0:RC,Mode[long]
```

Remarks

This function returns the EDM measurement mode.

Parameters

Mode	out	EDM measurement mode.

Return Codes

RC_OK	Execution always successful.

See Also

TMC_SetEdmMode

Example

```
RC_TYPE         rc;
EDM_MODE   Mode;

// set EDM mode to single standard
// if it is in any repeated mode

rc = TMC_GetEdmMode(Mode);
switch (Mode)
{
  case (EDM_CONT_STANDARD):
  case (EDM_CONT_DYNAMIC):
  case (EDM_CONT_FAST):
    rc = TMC_SetEdmMode(EDM_SINGLE_STANDARD);
    if (rc == RC_OK)
    {
      // set to single mode successful
    }
    else
    {
      // set not successful (subsystem busy)
    }
}
```

16.7.5 TMC_SetEdmMode - Set EDM measurement modes

C-Declaration

```
TMC_SetEdmMode(EDM_MODE Mode)
```

VB-Declaration

```
VB_TMC_SetEdmMode(ByVal Mode As Long)
```

ASCII-Request

`%R1Q,2020:`*Mode[long]*

ASCII-Response

`%R1P,0,0:`*RC*

Remarks

This function set the current measurement modes new. The measure function `TMC_DoMeasure(TMC_DEF_DIST)` will work with this configuration.

Parameters

`Mode`	`in`	EDM measurement mode.

Return Codes

`RC_OK`	Execution successful.
`TMC_BUSY`	TMC resource is locked respectively TMC task is busy. The EDM mode is not set. Repeat measurement.

See Also

`TMC_GetEdmMode`
`TMC_DoMeasure`

Example

see `TMC_GetEdmMode`

16.7.6 TMC_GetSimpleCoord - Get cartesian coordinates

C-Declaration

```
TMC_GetSimpleCoord(  SYSTIME WaitTime,
                     double &dCoordE,
                     double& dCoordN,
                     double& dCoordH,
                     TMC_INCLINE_PRG eProg)
```

VB-Declaration

```
VB_TMC_GetSimpleCoord(  ByVal WaitTime As Long,
                        dCoordE As Double,
                        dCoordN As Double,
                        dCoordH As Double,
                        ByVal eProg As Long)
```

ASCII-Request

`%R1Q,2116:`*WaitTime[long],eProg[long]*

ASCII-Response

`%R1P,0,0:`*RC,dCoordE[double], dCoordN[double], dCoordH[double]*

Remarks

This function get the cartesian co-ordinates if a valid distance existing. The parameter `WaitTime` defined the max wait time in order to get a valid distance. If after the wait time not a valid distance existing, the function initialise the parameter for the co-ordinates (`E,N,H`) with 0 and returns a error. For the co-ordinate calculate will require incline results. With the parameter `eProg` you have the possibility the incline results either to calculate or to measure it anew explicitly. We recommend to use the third variant, let the system determined (see parameters).

Parameters

`WaitTime`	in	Max. wait time to get a valid distance [ms]
`eProg`	in	Incline measuring program `TMC_MEA_INC`: Incline will explicit measured, the execution time for the function will so drastic increased respectively the measure rate will be slow down. **Note**: If you need a high accuracy then use this program: `TMC_PLANE_INC`: incline will be calculated, this variant is the fastest way to get the co-ordinates. **Note**: If you need a high measure rate and the station got a high stability on tilting use this program. `TMC_AUTO_INC`: the system decides whether the incline must be calculated or measured.

Note: The best performance regarding measure rate and accuracy you get with the automatically program, the instrument checks the conditions around the station. We recommend to take this mode any time.

dCoordE	out	eastern co-ordinate
dcoordN	out	northern co-ordinate
dcoordH	out	high co-ordinate

Return-Codes

RC_OK	Measurement ok
TMC_NO_FULL_CORRECTION	The results are not corrected by all active sensors. Co-ordinates are available. In order to secure witch correction is missing use the both functions TMC_IfDataAzeCorrError and TMC_IfDataIncCorrError
TMC_ACCURACY_GUARANTEE	Accuracy is not guaranteed, because the result are consist of measuring data which accuracy could not be verified by the system. Co-ordinates are available.
TMC_DIST_PPM	Wrong EDM settings, co-ordinates are not valid and set to 0. Set EDM ppm value to 0.
TMC_DIST_ERROR	No measuring, because of missing target point, co-ordinates are not valid and set to 0 Aim target point and try it again
TMC_BUSY	TMC resource is locked respectively TMC task is busy, co-ordinates are not valid and set to 0. Repeat measurement.
TMC_ANGLE_ERROR	Problems with angle res. incline sensor. At repeated occur call service.
RC_ABORT	Measurement through customer aborted.

RC_SHUT_DOWN	System power off through customer
RC_IVRESULT	No distance existing, co-ordinates are not valid and set to 0. Execute a distance measurement first.

See Also

TMC_GetCoordinate
TMC_IfDataAzeCorrError
TMC_IfDataIncCorrError

Example

```
RC_TYPE         Result;
TMC_ANG_SWITCH  SwCorr;
SYSTIME         WaitTime;
TMC_INCLINE_PRG ePrgm;
BOOLE           bExit;
Double          dCoordE,dCoordN,dCoordH;


TMC_GetAngSwitch(SwCorr);   // measure rate will
SwCorr.eInclineCorr=ON;     // be reduced with
SwCorr.eStandAxisCorr=ON;   // angle and incline
SwCorr.eCollimationCorr=ON; // corrections.
SwCorr.eTiltAxisCorr=ON;
TMC_DoMeasure(TMC_CLEAR); // clear distance first
TMC_SetAngSwitch(SwCorr); // before you can set the
                          // ANG switches, the
                          // distance must be
                          // cleared

TMC_DoMeasure(TMC_RTRK_DIST);// execute rapid
                             // tracking
                             // measurement

WaitTime=500;// set max. wait time 500 [ms]
eProg=TMC_AUTO_INC;// set automatically incline prgm
bExit=FALSE;
do
{
Result=TMC_GetSimpleCoord(WaitTime, dCoordE,
                          dCoordN, dCoordH,eProg);
switch(Result)
{
case RC_OK:
case TMC_NO_FULL_CORRECTION:
case TMC_ACCURACY_GUARANTEE:
```

```
        // in this cases are the coordinates
        // available
    Break;
    Default:
        bExit=TRUE;
        // in all other cases are the coordinates not
        // valid and set to 0
        // further errorhandling
    Break;
    }// end switch
    }// end do while
    while(!bExit);

    TMC_DoMeasure(TMC_CLEAR); // complete measurement
                              // and clear data
```

16.7.7 TMC_IfDataAzeCorrError - If ATR error occur

C-Declaration

```
TMC_IfDataAzeCorrError(BOOLE& bAtrCorrectionError)
```

VB-Declaration

```
VB_TMC_IfDataAzeCorrError
                (bAtrCorrectionError As Long)
```

ASCII-Request

```
%R1Q,2114:
```

ASCII-Response

```
%R1P,0,0:RC,bAtrCorrectionError[long]
```

Remarks

If you get back the return code TMC_ANGLE_NO_FULL_CORRECTION or TMC_ NO_FULL_CORRECTION from a measurement function, so you can find out with this function, whether the returned data record from the measurement function a missing deviation correction of the ATR included or not.

Parameters

`bAtrCorrectionError`	Out	Flag, if ATR correction error occurred or not `FALSE`: no error occurred `TRUE`: last data record not

corrected with the ATR-deviation

Return-Codes

RC_OK	Always

See Also

TMC_IfDataIncCorrError

Example

```
RC_TYPE         Result;
SYSTIME         WaitTime;
TMC_INCLINE_PRG ePrgm;
double          dCoordE,dCoordN,dCoordH;

TMC_DoMeasure(TMC_DEF_DIST);// execute single
                            // dist measurement

WaitTime=500;// set max. wait time 500 [ms]
eProg=TMC_AUTO_INC;// set automatically incline prgm

Result=TMC_GetSimpleCoord(WaitTime, dCoordE,
                          dCoordN, dCoordH,eProg);
switch(Result)
{
case TMC_NO_FULL_CORRECTION:
     TMC_IfDataAzeCorrError(bAzeCorrError);
     TMC_IfDataIncCorrError(bIncCorrError);
     if(bAzeCorrError)
     {
     // coordinates are not corrected with the Aze-
     // deviation correction
     }
     if(bIncCorrError)
     {
     // coordinates are not corrected with the
     // incline correction
     }
case RC_OK:
case TMC_ACCURACY_GUARANTEE:
     // in this cases are the coordinates
     // available
break;
default:
     // in all other cases are the coordinates not
     // valid and set to 0
     // further errorhandling
```

```
break;
}// end switch

TMC_DoMeasure(TMC_CLEAR); // complete measurement
                          // and clear data
```

16.7.8 TMC_IfDataIncCorrError - If incline error occur

C-Declaration

```
TMC_IfDataIncCorrError(BOOLE& bIncCorrectionError)
```

VB-Declaration

```
VB_TMC_IfDataIncCorrError
                (bIncCorrectionError As Long)
```

ASCII-Request

```
%R1Q,2115:
```

ASCII-Response

```
%R1P,0,0:RC,bIncCorrectionError[long]
```

Remarks

If you get back the return code TMC_ANGLE_NO_FULL_CORRECTION or TMC_ NO_FULL_CORRECTION from a measurement function, so you can find out with this function, whether the returned data record from the measurement function a missing inclination correction of the incline sensor included or not. A error information can only occur if the incline sensor is active.

Parameters

bIncCorrectionError	out	Flag, if incline correction error occurred or not FALSE: no error occurred TRUE: last data record not corrected with the incline-correction

Return-Codes

RC_OK	always

See Also

TMC_IfDataAzeCorrError

Example

```
see example TMC_IfDataAzeCorrError
```

16.7.9 TMC_SetAngSwitch - Enable/disable angle corrections

C-Declaration

```
TMC_SetAngSwitch(TMC_ANG_SWITCH Switch)
```

VB-Declaration

```
VB_TMC_SetAngSwitch(ByVal Switch As TMC_ANG_SWITCH)
```

ASCII-Request

```
%R1Q,2016:
```

ASCII-Response

```
%R1P,0,0:RC
```

Remarks

With this function you can enable/disable follow angle measurement correction.

`incline:`	The incline will be considered in the angle measurement if enabled.
`stand axis:`	The stand axis will be considered in the angle measurement if enabled.
`collimation:`	The collimation will be considered in the angle measurement if enabled
`tilt axis:`	The tilt axis will be considered in the angle measurement if enabled.

Note: You can set the various corrections only, if no distance is existing! (Use the function `TMC_DoMeasure(TMC_CLEAR,..)` in order to clear the distance)

Parameters

`Switch`	Angle measurement corrections

Return-Code

`RC_OK`	Corrections are set
`TMC_BUSY`	TMC resource is locked respectively TMC task is busy or a distance is existing. Clear distance and try it again.

See-Also

```
TMC_DoMeasure
TMC_GetAngSwitch
```

Example

```
See example TMC_QuickDist
```

16.7.10 TMC_GetSlopeDistCorr - Get slope distance correction factors

C-Declaration

```
TMC_GetSlopeDistCorr (double dPpmCorr,
                      double dPrismCorr)
```

VB-Declaration

```
VB_TMC_GetSlopeDistCorr(dPpmCorr   As Double,
                        dPrismCorr As Double)
```

ASCII-Request

```
%R1Q,2126:
```

ASCII-Response

```
%R1P,0,0: RC,dPpmCorr[double], dPrismCorr[double]
```

Remarks

This function retrieves the correction factors that are used for slope distance measurement corrections.

Parameters

dPpmCorr	out	General correction factor.
dPrismCorr	out	The correction factor of the prism.

Return Codes

RC_OK	Execution successful.

See Also

```
TMC_GetPrismCorr,
TMC_SetPrismCorr.
```

Example

-

17 WI - REGISTRATION - WIR

This chapter describes in which format measurements should be stored on PC-Card. We distinguish between the old GSI - 8 format and the newly established GSI - 16 format. In the former case the data part is 8 bytes and in the latter case 16 bytes.

17.1 CONSTANTS

Record Format Constants

```
typedef short WIR_RECFORMAT;
const WIR_RECFORMAT WIR_RECFORMAT_GSI = 0;
    // defines recording format is GSI (standard)
const WIR_RECFORMAT WIR_RECFORMAT_GSI16 = 1;
    // defines recording format is the new GSI-16
```

17.2 FUNCTIONS

17.2.1 WIR_GetRecFormat - Get Record Format

C-Declaration

```
WIR_GetRecFormat(WIR_RECFORMAT &RecFormat )
```

VB-Declaration

```
VB_WIR_GetRecFormat( RecFormat As Integer )
```

ASCII-Request

```
%R1Q,8011:
```

ASCII-Response

```
%R1P,0,0:RC, RecFormat[short]
```

Remarks

This function retrieves which recording format is in use.

Parameters

`RecFormat`	out	`WIR_RECFORMAT_GSI` or `WIR_RECFORMAT_GSI16`

Return Codes

`RC_OK`	On successful termination.

See Also

WIR_SetRecFormat

Example

see WIR_SetRecFormat

17.2.2 WIR_SetRecFormat - Set Record Format

C-Declaration

```
WIR_SetRecFormat( WIR_RECFORMAT RecFormat )
```

VB-Declaration

```
VB_WIR_SetRecFormat( RecFormat As Integer)
```

ASCII-Request

```
%R1Q,8012:RecFormat[short]
```

ASCII-Response

```
%R1P,0,0:RC
```

Remarks

This function sets which recording format should be used.

Parameters

RecFormat	in	WIR_RECFORMAT_GSI or WIR_RECFORMAT_GSI16

Return Codes

RC_OK	On successful termination.

See Also

WIR_GetRecFormat

Example

```
// if in old GSI format then switch to new format
RetCode = WIR_GetRecFormat(Format);
if (Format == WIR_RECFORMAT_GSI)
{
  // switch to GSI16 format
  RetCode = WIR_SetRecFormat(WIR_RECFORMAT_GSI16);
}
```

18 PORTING A TPS1000 APPLICATION

The implementation of the TPS1100 theodolite series includes several new concepts compared to the firmware of TPS1000 theodolites. To take care of the new functionality, which has been changed or removed in the implementation of TPS1100 firmware, a few changes in GeoCOM for TPS1100 theodolites were necessary.

This chapter contains all RPCs and data types, which has changed in GeoCOM. It should help the developer to port a GeoCOM client application for TPS1000 theodolite series onto the new platform.

18.1 RPC CHANGES

The following section contains a list of all replaced, deleted and new RPCs. Refer to the RPC description in the corresponding subsystem to get further information on how to use the new RPCs.

18.1.1 Beep On/Off

TPS1000 GeoCOM RPC:	Changes in TPS1100 GeoCOM:
`BMM_BeepOff`	Replaced by: `IOS_BeepOff`
`BMM_BeepOn`	Replaced by: `IOS_BeepOn`

18.1.2 Central Services - CSV

TPS1000 GeoCOM RPC:	Changes in TPS1100 GeoCOM:
`CSV_GetUserInstrumentName`	Deleted
`CSV_SetUserInstrumentName`	Deleted

18.1.3 Electronic Distance Measurement - EDM

TPS1000 GeoCOM RPC:	Changes in TPS1100 GeoCOM:
`EDM_GetBumerang`	Deleted

TPS1000 GeoCOM RPC:	Changes in TPS1100 GeoCOM:
EDM_GetTrkLightBrightness EDM_GetTrkLightSwitch	Replaced by: EDM_GetEGLIntensity
EDM_On	Deleted
EDM_SetBumerang	Deleted
EDM_SetTrkLightBrightness EDM_SetTrkLightSwitch	Replaced by: EDM_SetEGLIntensity

18.1.4 Theodolite Measurement and Calculation - TMC

TPS1000 GeoCOM RPC:	Changes in TPS1100 GeoCOM:
-	New: TMC_GetSlopeDistCorr

18.2 DATA TYPES AND CONSTANTS CHANGES

The following data types and constants has changed or are new. Refer to the chapter constants and types in the corresponding subsystem to get the full description.

TPS1000 GeoCOM type definition:	**Changes in TPS1100 GeoCOM:**
-	New: EDM_EGLINTENSITY_TYPE
-	New: IOS_BEEP_STDINTENS
-	New: TMC_FACE
-	New: WIR_RECFORMAT
BAP_MEASURE_PRG	Extended: BAP_MEASURE_PRG
EDM_MODE	Changed: EDM_MODE
TMC_MEASURE_PRG	Extended: TMC_MEASURE_PRG
TPS_DEVICE_CLASS	Extended. TPS_DEVICE_CLASS
TPS_DEVICE_TYPE	Extended: TPS_DEVICE_TYPE

18.3 NEW RETURNCODES

The definitions of the returncodes have been coupled totally to the definitions of the TPS1100 firmware. Please refer to Appendix A for a detailed listing.

19 GEOCOM RELEASES

This chapter shows the changes between the different Releases of GeoCOM

19.1 RELEASE 1.04

This GeoCOM Release 1.04 was introduced with TPS Firmware Release 2.0.

The major change in this Release is the implementation of the binary protocol. Details see COM_GetBinaryAvailable.

19.1.1 RPC Changes

There are a few RPC's replaces with newer functions:

Old function	New function
AUT_GetAtrStatus	AUS_GetUserAtrState
AUT_GetAtrStatus	AUS_SetUserAtrState
AUT_GetLockStatus	AUS_GetUserLockState
AUT_SetLockStatus	AUS_SetUserLockState

The older RPC's are still working, but should not be used for new implementations.

19.1.2 New Functions

AUT_GetSearchArea
AUT_GetUserSpiral
AUT_SetSearchArea
AUT_SetUserSpiral
BAP_SearchTarget

19.2 RELEASE 1.05

This GeoCOM Release 1.05 was introduced with TPS Firmware Release 2.10.
There are a few new functions added.

19.2.1 RPC Changes

There are a few RPC's replaces with newer functions:

Old function	New function
CSV_GetVBat	CSV_CheckPower

The older RPC is still working, but should not be used for new implementations.

19.2.2 New Constants

TPS_DEVICE_RL_EXT

19.2.3 New Functions

AUS_GetRcsSearchSwitch, Aus_SwitchRcsSearch
BAP_GetTargetType, BAP_SetTargetType
BAP_GetPrismType, BAP_SetPrismType
BAP_GetPrismDef, BAP_SetPrismDef

APPENDIX

A RETURN CODES

The return codes described here are codes, which may be returned from RPC's and GeoCOM general functions (COMF). A successful completion will be denoted by RC_OK. Almost all of the return codes are error codes. Nevertheless, some of them have a more informational character. Therefore, refer also to the description of a specific function. In a special context the meaning of a return code might vary a little bit.

The list described here is organised in subsystem related categories. The `RetCodeName` describes the constant as it is defined for the TPS1100 series instruments. Additionally to find an error code by number they are given too.

A-1 GENERAL RETURN CODES

TPS 0 0x0

RetCodeName	Value	Hex	Description
RC_OK	0	0x0	Function successfully completed.
RC_UNDEFINED	1	0x1	Unknown error, result unspecified.
RC_IVPARAM	2	0x2	Invalid parameter detected. Result unspecified.
RC_IVRESULT	3	0x3	Invalid result.
RC_FATAL	4	0x4	Fatal error.
RC_NOT_IMPL	5	0x5	Not implemented yet.
RC_TIME_OUT	6	0x6	Function execution timed out. Result unspecified.
RC_SET_INCOMPL	7	0x7	Parameter setup for subsystem is incomplete.
RC_ABORT	8	0x8	Function execution has been aborted.
RC_NOMEMORY	9	0x9	Fatal error - not enough memory.
RC_NOTINIT	10	0xA	Fatal error - subsystem not initialized.
RC_SHUT_DOWN	12	0xC	Subsystem is down.
RC_SYSBUSY	13	0xD	System busy/already in use of another process. Cannot execute

RetCodeName	Value	Hex	Description
			function.
RC_HWFAILURE	14	0xE	Fatal error - hardware failure.
RC_ABORT_APPL	15	0xF	Execution of application has been aborted (SHIFT-ESC).
RC_LOW_POWER	16	0x10	Operation aborted - insufficient power supply level.
RC_IVVERSION	17	0x11	Invalid version of file, ...
RC_BATT_EMPTY	18	0x12	Battery empty
RC_NO_EVENT	20	0x14	no event pending.
RC_OUT_OF_TEMP	21	0x15	out of temperature range
RC_INSTRUMENT_TILT	22	0x16	instrument tilting out of range
RC_COM_SETTING	23	0x17	communication error
RC_NO_ACTION	24	0x18	RC_TYPE Input 'do no action'
RC_SLEEP_MODE	25	0x19	Instr. run into the sleep mode

ANG 256 0x100

RetCodeName	Value	Hex	Description
ANG_ERROR	257	0x101	Angles and Inclinations not valid
ANG_INCL_ERROR	258	0x102	inclinations not valid
ANG_BAD_ACC	259	0x103	value accuracy not reached
ANG_BAD_ANGLE_ACC	260	0x104	angle-accuracy not reached
ANG_BAD_INCLIN_ACC	261	0x105	inclination accuracy not reached
ANG_WRITE_PROTECTED	266	0x10A	no write access allowed
ANG_OUT_OF_RANGE	267	0x10B	value out of range
ANG_IR_OCCURED	268	0x10C	function aborted due to interrupt
ANG_HZ_MOVED	269	0x10D	hz moved during incline measurement
ANG_OS_ERROR	270	0x10E	troubles with operation system
ANG_DATA_ERROR	271	0x10F	overflow at parameter values
ANG_PEAK_CNT_UFL	272	0x110	too less peaks
ANG_TIME_OUT	273	0x111	reading timeout
ANG_TOO_MANY_EXPOS	274	0x112	too many exposures wanted
ANG_PIX_CTRL_ERR	275	0x113	picture height out of range
ANG_MAX_POS_SKIP	276	0x114	positive exposure dynamic overflow
ANG_MAX_NEG_SKIP	277	0x115	negative exposure dynamic overflow
ANG_EXP_LIMIT	278	0x116	exposure time overflow
ANG_UNDER_EXPOSURE	279	0x117	picture under-exposured
ANG_OVER_EXPOSURE	280	0x118	picture over-exposured

RetCodeName	Value	Hex	Description
ANG_TMANY_PEAKS	300	0x12C	too many peaks detected
ANG_TLESS_PEAKS	301	0x12D	too less peaks detected
ANG_PEAK_TOO_SLIM	302	0x12E	peak too slim
ANG_PEAK_TOO_WIDE	303	0x12F	peak to wide
ANG_BAD_PEAKDIFF	304	0x130	bad peak difference
ANG_UNDER_EXP_PICT	305	0x131	too less peak amplitude
ANG_PEAKS_INHOMOGEN	306	0x132	in-homogenous peak amplitudes
ANG_NO_DECOD_POSS	307	0x133	no peak decoding possible
ANG_UNSTABLE_DECOD	308	0x134	peak decoding not stable
ANG_TLESS_FPEAKS	309	0x135	too less valid fine-peaks

ATA 512 0x200

RetCodeName	Value	Hex	Description
ATA_RC_NOT_READY	512	0x200	ATR-System is not ready.
ATA_RC_NO_RESULT	513	0x201	Result isn't available yet.
ATA_RC_SEVERAL_TARGETS	514	0x202	Several Targets detected.
ATA_RC_BIG_SPOT	515	0x203	Spot is too big for analyze.
ATA_RC_BACKGROUND	516	0x204	Background is too bright.
ATA_RC_NO_TARGETS	517	0x205	No targets detected.
ATA_RC_NOT_ACCURAT	518	0x206	Accuracy worse than asked for.
ATA_RC_SPOT_ON_EDGE	519	0x207	Spot is on the edge of the sensing area.
ATA_RC_BLOOMING	522	0x20A	Blooming or spot on edge detected.
ATA_RC_NOT_BUSY	523	0x20B	ATR isn't in a continuous mode.
ATA_RC_STRANGE_LIGHT	524	0x20C	Not the spot of the own target illuminator.
ATA_RC_V24_FAIL	525	0x20D	Communication error to sensor (ATR).
ATA_RC_HZ_FAIL	527	0x20F	No Spot detected in Hz-direction.
ATA_RC_V_FAIL	528	0x210	No Spot detected in V-direction.
ATA_RC_HZ_STRANGE_L	529	0x211	Strange light in Hz-direction.
ATA_RC_V_STRANGE_L	530	0x212	Strange light in V-direction.
ATA_SLDR_TRANSFER_PENDING	531	0x213	On multiple ATA_SLDR_OpenTransfer.
ATA_SLDR_TRANSFER_ILLEGAL	532	0x214	No ATA_SLDR_OpenTransfer happened.

RetCodeName	Value	Hex	Description
ATA_SLDR_DATA_ ERROR	533	0x215	Unexpected data format received.
ATA_SLDR_CHK_SUM_ ERROR	534	0x216	Checksum error in transmitted data.
ATA_SLDR_ADDRESS_ ERROR	535	0x217	Address out of valid range.
ATA_SLDR_INV_ LOADFILE	536	0x218	Firmware file has invalid format.
ATA_SLDR_ UNSUPPORTED	537	0x219	Current (loaded) Firmware doesn't support upload.

EDM 768 0x300

RetCodeName	Value	Hex	Description
EDM_SYSTEM_ERR	769	0x301	Fatal EDM sensor error. See for the exact reason the original EDM sensor error number. In the most cases a service problem
EDM_INVALID_COMMAND	770	0x302	Invalid command or unknown command, see command syntax.
EDM_BOOM_ERR	771	0x303	Boomerang error.
EDM_SIGN_LOW_ERR	772	0x304	Received signal to low, prism to far away, or natural barrier, bad environment, etc.
EDM_DIL_ERR	773	0x305	DIL distance measurement out of limit.
EDM_SIGN_HIGH_ERR	774	0x306	Received signal to strong, prism to near, stranger light effect.
EDM_DEV_NOT_ INSTALLED	778	0x30A	Device like EGL, DL is not installed.
EDM_NOT_FOUND	779	0x30B	Search result invalid. For the exact explanation see in the description of the called function.
EDM_ERROR_RECEIVED	780	0x30C	Communication ok, but an error reported from the EDM sensor.
EDM_MISSING_SRVPWD	781	0x30D	No service password is set.
EDM_INVALID_ANSWER	782	0x30E	Communication ok, but an unexpected answer received.
EDM_SEND_ERR	783	0x30F	Data send error, sending buffer is full.

RetCodeName	Value	Hex	Description
EDM_RECEIVE_ERR	784	0x310	Data receive error, like parity buffer overflow.
EDM_INTERNAL_ERR	785	0x311	Internal EDM subsystem error.
EDM_BUSY	786	0x312	Sensor is working already, abort current measuring first.
EDM_NO_ MEASACTIVITY	787	0x313	No measurement activity started.
EDM_CHKSUM_ERR	788	0x314	Calculated checksum, resp. received data wrong (only in binary communication mode possible).
EDM_INIT_OR_STOP_ ERR	789	0x315	During start up or shut down phase an error occured. It is saved in the DEL buffer.
EDM_SRL_NOT_ AVAILABLE	790	0x316	Red laser not available on this sensor HW.
EDM_MEAS_ABORTED	791	0x317	Measurement will be aborted (will be used for the lasersecurity)
EDM_SLDR_TRANSFER_ PENDING	798	0x31E	Multiple OpenTransfer calls.
EDM_SLDR_TRANSFER_ ILLEGAL	799	0x31F	No opentransfer happened.
EDM_SLDR_DATA_ ERROR	800	0x320	Unexpected data format received.
EDM_SLDR_CHK_SUM_ ERROR	801	0x321	Checksum error in transmitted data.
EDM_SLDR_ADDR_ ERROR	802	0x322	Address out of valid range.
EDM_SLDR_INV_ LOADFILE	803	0x323	Firmware file has invalid format.
EDM_SLDR_ UNSUPPORTED	804	0x324	Current (loaded) firmware doesn't support upload.
EDM_UNKNOW_ERR	808	0x328	Undocumented error from the EDM sensor, should not occur.

GMF 1024 0x400

RetCodeName	Value	Hex	Description
GM_WRONG_AREA_DEF	1025	0x401	Wrong Area Definition.
GM_IDENTICAL_PTS	1026	0x402	Identical Points.
GM_PTS_IN_LINE	1027	0x403	Points on one line.

RetCodeName	Value	Hex	Description
GM_OUT_OF_RANGE	1028	0x404	Out of range.
GM_PLAUSIBILITY_ERR	1029	0x405	Plausibility error.
GM_TOO_FEW_ OBSERVATIONS	1030	0x406	To few Observations to calculate the average.
GM_NO_SOLUTION	1031	0x407	No Solution.
GM_ONE_SOLUTION	1032	0x408	Only one solution.
GM_TWO_SOLUTIONS	1033	0x409	Second solution.
GM_ANGLE_SMALLER_ 15GON	1034	0x40A	Warning: Intersection angle < 15gon.
GM_INVALID_ TRIANGLE_TYPE	1035	0x40B	Invalid triangle.
GM_INVALID_ ANGLE_SYSTEM	1036	0x40C	Invalid angle unit.
GM_INVALID_ DIST_SYSTEM	1037	0x40D	Invalid distance unit.
GM_INVALID_V_SYSTEM	1038	0x40E	Invalid vertical angle.
GM_INVALID_ TEMP_SYSTEM	1039	0x40F	Invalid temperature system.
GM_INVALID_ PRES_SYSTEM	1040	0x410	Invalid pressure unit.
GM_RADIUS_ NOT_POSSIBLE	1041	0x411	Invalid radius.
GM_NO_ PROVISIONAL_VALUES	1042	0x412	GM2: insufficient data.
GM_SINGULAR_MATRIX	1043	0x413	GM2: bad data
GM_TOO_MANY_ ITERATIONS	1044	0x414	GM2: bad data distr.
GM_IDENTICAL_ TIE_POINTS	1045	0x415	GM2: same tie points.
GM_SETUP_ EQUALS_TIE_POINT	1046	0x416	GM2: sta/tie point same.

TMC 1280 0x500

RetCodeName	Value	Hex	Description
TMC_NO_FULL_ CORRECTION	1283	0x503	Warning: measurement without full correction
TMC_ACCURACY_ GUARANTEE	1284	0x504	Info : accuracy can not be guarantee

RetCodeName	Value	Hex	Description
TMC_ANGLE_OK	1285	0x505	Warning: only angle measurement valid
TMC_ANGLE_NO_FULL_CORRECTION	1288	0x508	Warning: only angle measurement valid but without full correction
TMC_ANGLE_ACCURACY_GUARANTEE	1289	0x509	Info : only angle measurement valid but accuracy can not be guarantee
TMC_ANGLE_ERROR	1290	0x50A	Error : no angle measurement
TMC_DIST_PPM	1291	0x50B	Error : wrong setting of PPM or MM on EDM
TMC_DIST_ERROR	1292	0x50C	Error : distance measurement not done (no aim, etc.)
TMC_BUSY	1293	0x50D	Error : system is busy (no measurement done)
TMC_SIGNAL_ERROR	1294	0x50E	Error : no signal on EDM (only in signal mode)

MEM 1536 0x600

RetCodeName	Value	Hex	Description
MEM_OUT_OF_MEMORY	1536	0x600	out of memory
MEM_OUT_OF_HANDLES	1537	0x601	out of memory handles
MEM_TAB_OVERFLOW	1538	0x602	memory table overflow
MEM_HANDLE_INVALID	1539	0x603	used handle is invalid
MEM_DATA_NOT_FOUND	1540	0x604	memory data not found
MEM_DELETE_ERROR	1541	0x605	memory delete error
MEM_ZERO_ALLOC_ERR	1542	0x606	tried to allocate 0 bytes
MEM_REORG_ERR	1543	0x607	can't reorganize memory

MOT 1792 0x700

RetCodeName	Value	Hex	Description
MOT_RC_UNREADY	1792	0x700	Motorization not ready
MOT_RC_BUSY	1793	0x701	Motorization is handling another task
MOT_RC_NOT_OCONST	1794	0x702	Not in velocity mode
MOT_RC_NOT_CONFIG	1795	0x703	Motorization is in the wrong mode or busy
MOT_RC_NOT_POSIT	1796	0x704	Not in posit mode

RetCodeName	Value	Hex	Description
MOT_RC_NOT_SERVICE	1797	0x705	Not in service mode
MOT_RC_NOT_BUSY	1798	0x706	Motorization is handling no task
MOT_RC_NOT_LOCK	1799	0x707	Not in tracking mode
MOT_RC_NOT_SPIRAL	1800	0x708	Not in spiral mode

LDR 2048 0x800

RetCodeName	Value	Hex	Description
LDR_PENDING	2048	0x800	Transfer is already open
LDR_PRGM_OCC	2049	0x801	Maximal number of applications reached
LDR_TRANSFER_ ILLEGAL	2050	0x802	No Transfer is open
LDR_NOT_FOUND	2051	0x803	Function or program not found
LDR_ALREADY_EXIST	2052	0x804	Loadable object already exists
LDR_NOT_EXIST	2053	0x805	Can't delete. Object does not exist
LDR_SIZE_ERROR	2054	0x806	Error in loading object
LDR_MEM_ERROR	2055	0x807	Error at memory allocation/release
LDR_PRGM_NOT_EXIST	2056	0x808	Can't load text-object because application does not exist
LDR_FUNC_LEVEL_ERR	2057	0x809	Call-stack limit reached
LDR_RECURSIV_ERR	2058	0x80A	Recursive calling of an loaded function
LDR_INST_ERR	2059	0x80B	Error in installation function
LDR_FUNC_OCC	2060	0x80C	Maximal number of functions reached
LDR_RUN_ERROR	2061	0x80D	Error during a loaded application program
LDR_DEL_MENU_ERR	2062	0x80E	Error during deleting of menu entries of an application
LDR_OBJ_TYPE_ERROR	2063	0x80F	Loadable object is unknown
LDR_WRONG_SECKEY	2064	0x810	Wrong security key
LDR_ILLEGAL_LOADADR	2065	0x811	Illegal application memory address
LDR_IEEE_ERROR	2066	0x812	Loadable object file is not IEEE format
LDR_WRONG_APPL_ VERSION	2067	0x813	Bad application version number

BMM 2304 0x900

RetCodeName	Value	Hex	Description
BMM_XFER_PENDING	2305	0x901	Loading process already opened
BMM_NO_XFER_OPEN	2306	0x902	Transfer not opened
BMM_UNKNOWN_CHARSET	2307	0x903	Unknown character set
BMM_NOT_INSTALLED	2308	0x904	Display module not present
BMM_ALREADY_EXIST	2309	0x905	Character set already exists
BMM_CANT_DELETE	2310	0x906	Character set cannot be deleted
BMM_MEM_ERROR	2311	0x907	Memory cannot be allocated
BMM_CHARSET_USED	2312	0x908	Character set still used
BMM_CHARSET_SAVED	2313	0x909	Char-set cannot be deleted or is protected
BMM_INVALID_ADR	2314	0x90A	Attempt to copy a character block outside the allocated memory
BMM_CANCELANDADR_ERROR	2315	0x90B	Error during release of allocated memory
BMM_INVALID_SIZE	2316	0x90C	Number of bytes specified in header does not match the bytes read
BMM_CANCELAND INVSIZE_ERROR	2317	0x90D	Allocated memory could not be released
BMM_ALL_GROUP_OCC	2318	0x90E	Max. number of character sets already loaded
BMM_CANT_DEL_LAYERS	2319	0x90F	Layer cannot be deleted
BMM_UNKNOWN_LAYER	2320	0x910	Required layer does not exist
BMM_INVALID_LAYERLEN	2321	0x911	Layer length exceeds maximum

TXT 2560 0xA00

RetCodeName	Value	Hex	Description
TXT_OTHER_LANG	2560	0xA00	text found, but in an other language
TXT_UNDEF_TOKEN	2561	0xA01	text not found, token is undefined
TXT_UNDEF_LANG	2562	0xA02	language is not defined
TXT_TOOMANY_LANG	2563	0xA03	maximal number of languages reached
TXT_GROUP_OCC	2564	0xA04	desired text group is already in use

RetCodeName	Value	Hex	Description
TXT_INVALID_GROUP	2565	0xA05	text group is invalid
TXT_OUT_OF_MEM	2566	0xA06	out of text memory
TXT_MEM_ERROR	2567	0xA07	memory write / allocate error
TXT_TRANSFER_PENDING	2568	0xA08	text transfer is already open
TXT_TRANSFER_ILLEGAL	2569	0xA09	text transfer is not opened
TXT_INVALID_SIZE	2570	0xA0A	illegal text data size
TXT_ALREADY_EXIST	2571	0xA0B	language already exists

MMI 2816 0xB00

RetCodeName	Value	Hex	Description
MMI_BUTTON_ID_EXISTS	2817	0xB01	Button ID already exists
MMI_DLG_NOT_OPEN	2818	0xB02	Dialog not open
MMI_DLG_OPEN	2819	0xB03	Dialog already open
MMI_DLG_SPEC_MISMATCH	2820	0xB04	Number of fields specified with OpenDialogDef does not match
MMI_DLGDEF_EMPTY	2821	0xB05	Empty dialog definition
MMI_DLGDEF_NOT_OPEN	2822	0xB06	Dialog definition not open
MMI_DLGDEF_OPEN	2823	0xB07	Dialog definition still open
MMI_FIELD_ID_EXISTS	2824	0xB08	Field ID already exists
MMI_ILLEGAL_APP_ID	2825	0xB09	Illegal application ID
MMI_ILLEGAL_BUTTON_ID	2826	0xB0A	Illegal button ID
MMI_ILLEGAL_DLG_ID	2827	0xB0B	Illegal dialog ID
MMI_ILLEGAL_FIELD_COORDS	2828	0xB0C	Illegal field coordinates or length/height
MMI_ILLEGAL_FIELD_ID	2829	0xB0D	Illegal field ID
MMI_ILLEGAL_FIELD_TYPE	2830	0xB0E	Illegal field type
MMI_ILLEGAL_FIELD_FORMAT	2831	0xB0F	Illegal field format
MMI_ILLEGAL_FIXLINES	2832	0xB10	Illegal number of fix dialog lines
MMI_ILLEGAL_MB_TYPE	2833	0xB11	Illegal message box type
MMI_ILLEGAL_MENU_ID	2834	0xB12	Illegal menu ID
MMI_ILLEGAL_MENUITEM_ID	2835	0xB13	Illegal menu item ID
MMI_ILLEGAL_NEXT_ID	2836	0xB14	Illegal next field ID
MMI_ILLEGAL_TOPLINE	2837	0xB15	Illegal topline number

RetCodeName	Value	Hex	Description
MMI_NOMORE_BUTTONS	2838	0xB16	No more buttons per dialog/menu available
MMI_NOMORE_DLGS	2839	0xB17	No more dialogs available
MMI_NOMORE_FIELDS	2840	0xB18	No more fields per dialog available
MMI_NOMORE_MENUS	2841	0xB19	No more menus available
MMI_NOMORE_MENUITEMS	2842	0xB1A	No more menu items available
MMI_NOMORE_WINDOWS	2843	0xB1B	No more windows available
MMI_SYS_BUTTON	2844	0xB1C	The button belongs to the MMI
MMI_VREF_UNDEF	2845	0xB1D	The parameter list for OpenDialog is uninitialized
MMI_EXIT_DLG	2846	0xB1E	The MMI should exit the dialog
MMI_KEEP_FOCUS	2847	0xB1F	The MMI should keep focus within field being edited
MMI_NOMORE_ITEMS	2848	0xB20	Notification to the MMI that no more items available

COM 3072 0xC00

RetCodeName	Value	Hex	Description
RC_COM_ERO	3072	0xC00	Initiate Extended Runtime Operation (ERO).
RC_COM_CANT_ENCODE	3073	0xC01	Cannot encode arguments in client.
RC_COM_CANT_DECODE	3074	0xC02	Cannot decode results in client.
RC_COM_CANT_SEND	3075	0xC03	Hardware error while sending.
RC_COM_CANT_RECV	3076	0xC04	Hardware error while receiving.
RC_COM_TIMEDOUT	3077	0xC05	Request timed out.
RC_COM_WRONG_FORMAT	3078	0xC06	Packet format error.
RC_COM_VER_MISMATCH	3079	0xC07	Version mismatch between client and server.
RC_COM_CANT_DECODE_REQ	3080	0xC08	Cannot decode arguments in server.
RC_COM_PROC_UNAVAIL	3081	0xC09	Unknown RPC, procedure ID invalid.
RC_COM_CANT_ENCODE_REP	3082	0xC0A	Cannot encode results in server.
RC_COM_SYSTEM_ERR	3083	0xC0B	Unspecified generic system error.
RC_COM_FAILED	3085	0xC0D	Unspecified error.

RetCodeName	Value	Hex	Description
RC_COM_NO_BINARY	3086	0xC0E	Binary protocol not available.
RC_COM_INTR	3087	0xC0F	Call interrupted.
RC_COM_ REQUIRES_8DBITS	3090	0xC12	Protocol needs 8bit encoded characters.
RC_COM_TR_ID_ MISMATCH	3093	0xC15	Transaction ID mismatch error.
RC_COM_NOT_GEOCOM	3094	0xC16	Protocol not recognizable.
RC_COM_ UNKNOWN_PORT	3095	0xC17	(WIN) Invalid port address.
RC_COM_ERO_END	3099	0xC1B	ERO is terminating.
RC_COM_OVERRUN	3100	0xC1C	Internal error: data buffer overflow.
RC_COM_SRVR_ RX_CHECKSUM_ERROR	3101	0xC1D	Invalid checksum on server side received.
RC_COM_CLNT_ RX_CHECKSUM_ERROR	3102	0xC1E	Invalid checksum on client side received.
RC_COM_PORT_ NOT_AVAILABLE	3103	0xC1F	(WIN) Port not available.
RC_COM_PORT_ NOT_OPEN	3104	0xC20	(WIN) Port not opened.
RC_COM_NO_PARTNER	3105	0xC21	(WIN) Unable to find TPS.
RC_COM_ERO_ NOT_STARTED	3106	0xC22	Extended Runtime Operation could not be started.
RC_COM_CONS_REQ	3107	0xC23	Att to send cons reqs
RC_COM_SRVR_ IS_SLEEPING	3108	0xC24	TPS has gone to sleep. Wait and try again.
RC_COM_SRVR_IS_OFF	3109	0xC25	TPS has shut down. Wait and try again.

DPL 3328 0xD00

RetCodeName	Valu	Hex	Description
DPL_RC_NOCREATE	3328	0xD00	no file creation, fatal
DPL_RC_NOTOPEN	3329	0xD01	bank not open
DPL_RC_ALRDYOPEN	3330	0xD02	a databank is already open
DPL_RC_NOTFOUND	3331	0xD03	databank file does not exist
DPL_RC_EXISTS	3332	0xD04	databank already exists
DPL_RC_EMPTY	3333	0xD05	databank is empty
DPL_RC_BADATA	3334	0xD06	bad data detected
DPL_RC_BADFIELD	3335	0xD07	bad field type
DPL_RC_BADINDEX	3336	0xD08	bad index information
DPL_RC_BADKEY	3337	0xD09	bad key type
DPL_RC_BADMODE	3338	0xD0A	bad mode

RetCodeName	Valu	Hex	Description
DPL_RC_BADRANGE	3339	0xD0B	bad range
DPL_RC_DUPLICATE	3340	0xD0C	duplicate keys not allowed
DPL_RC_INCOMPLETE	3341	0xD0D	record is incomplete
DPL_RC_IVDBID	3342	0xD0E	invalid db project id
DPL_RC_IVNAME	3343	0xD0F	invalid name
DPL_RC_LOCKED	3344	0xD10	data locked
DPL_RC_NOTLOCKED	3345	0xD11	data not locked
DPL_RC_NODATA	3346	0xD12	no data found
DPL_RC_NOMATCH	3347	0xD13	no matching key found
DPL_RC_NOSPACE	3348	0xD14	no more (disk) space left
DPL_RC_NOCLOSE	3349	0xD15	could not close db (sys. error)
DPL_RC_RELATIONS	3350	0xD16	record still has relations
DPL_RC_NULLPTR	3351	0xD17	null pointer
DPL_RC_BADFORMAT	3352	0xD18	bad databank format, wrong version
DPL_RC_BADRECTYPE	3353	0xD19	bad record type
DPL_RC_OUTOFMEM	3354	0xD1A	no more (memory) space left
DPL_RC_CODE_ MISMATCH	3355	0xD1B	code mismatch
DPL_RC_NOTINIT	3356	0xD1C	db has not been initialized
DPL_RC_NOTEXIST	3357	0xD1D	trf. for old db's does not exist
DPL_RC_NOTOK	4864	0x1300	not ok
DPL_RC_IVAPPL	4865	0x1301	invalid database system appl.
DPL_RC_NOT_ AVAILABLE	4866	0x1302	database not available
DPL_RC_NO_CODELIST	4867	0x1303	no codelist found
DPL_RC_TO_MANY_ CODELISTS	4868	0x1304	more then DPL_MAX_CODELISTS found

FIL 3840 0xF00

RetCodeName	Value	Hex	Description
RC_FIL_NO_ERROR	3840	0xF00	Operation completed successfully.
RC_FIL_FILNAME_ NOT_FOUND	3845	0xF05	File name not found.
RC_FIL_NO_MAKE_ DIRECTORY	3880	0xF28	Cannot create directory.
RC_FIL_RENAME_ FILE_FAILED	3886	0xF2E	Rename of file failed.
RC_FIL_INVALID_PATH	3888	0xF30	Invalid path specified.
RC_FIL_FILE_	3898	0xF3A	Cannot delete file.

RetCodeName	Value	Hex	Description
NOT_DELETED			
RC_FIL_ILLEGAL_ORIGIN	3906	0xF42	Illegal origin.
RC_FIL_END_OF_FILE	3924	0xF54	End of file reached.
RC_FIL_NO_MORE_ROOM_ON_MEDIUM	3931	0xF5B	Medium full.
RC_FIL_PATTERN_DOES_NOT_MATCH	3932	0xF5C	Pattern does not match file names.
RC_FIL_FILE_ALREADY_OPEND_FOR_WR	3948	0xF6C	File is already open with write permission.
RC_FIL_WRITE_TO_MEDIUM_FAILED	3957	0xF75	Write operation to medium failed.
RC_FIL_START_SEARCH_NOT_CALLED	3963	0xF7B	FIL_StartList not called.
RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE	3964	0xF7C	No medium existent in device.
RC_FIL_ILLEGAL_FILE_OPEN_TYPE	3965	0xF7D	Illegal file open type.
RC_FIL_MEDIUM_NEWLY_INSERTED	3966	0xF7E	Medium freshly inserted into device.
RC_FIL_MEMORY_FAILED	3967	0xF7F	Memory failure. No more memory available.
RC_FIL_FATAL_ERROR	3968	0xF80	Fatal error during file operation.
RC_FIL_FAT_ERROR	3969	0xF81	Fatal error in file allocation table.
RC_FIL_ILLEGAL_DRIVE	3970	0xF82	Illegal drive chosen.
RC_FIL_INVALID_FILE_DESCR	3971	0xF83	Illegal file descriptor.
RC_FIL_SEEK_FAILED	3972	0xF84	Seek failed.
RC_FIL_CANNOT_DELETE	3973	0xF85	Cannot delete file.
RC_FIL_MEDIUM_WRITE_PROTECTED	3974	0xF86	Medium is write protected.
RC_FIL_BATTERY_LOW	3975	0xF87	Medium backup battery is low.
RC_FIL_BAD_FORMAT	3976	0xF88	Bad medium format.
RC_FIL_UNSUPPORTED_MEDIUM	3977	0xF89	Unsupported PC-Card detected.
RC_FIL_RENAME_DIR_FAILED	3978	0xF8A	Directory exists already

WIR 5120 0x1400

RetCodeName	Value	Hex	Description
WIR_PTNR_OVERFLOW	5121	0x1401	point number overflow
WIR_NUM_ASCII_CARRY	5122	0x1402	carry from number to ascii conversion
WIR_PTNR_NO_INC	5123	0x1403	can't increment point number
WIR_STEP_SIZE	5124	0x1404	wrong step size
WIR_BUSY	5125	0x1405	resource occupied
WIR_CONFIG_FNC	5127	0x1407	user function selected
WIR_CANT_OPEN_FILE	5128	0x1408	can't open file
WIR_FILE_WRITE_ ERROR	5129	0x1409	can't write into file
WIR_MEDIUM_NOMEM	5130	0x140A	no anymore memory on PC-Card
WIR_NO_MEDIUM	5131	0x140B	no PC-Card
WIR_EMPTY_FILE	5132	0x140C	empty GSI file
WIR_INVALID_DATA	5133	0x140D	invalid data in GSI file
WIR_F2_BUTTON	5134	0x140E	F2 button pressed
WIR_F3_BUTTON	5135	0x140F	F3 button pressed
WIR_F4_BUTTON	5136	0x1410	F4 button pressed
WIR_F5_BUTTON	5137	0x1411	F5 button pressed
WIR_F6_BUTTON	5138	0x1412	F6 button pressed
WIR_SHF2_BUTTON	5139	0x1413	SHIFT F2 button pressed

AUT 8704 0x2200

RetCodeName	Value	Hex	Description
AUT_RC_TIMEOUT	8704	0x2200	Position not reached
AUT_RC_DETENT_ ERROR	8705	0x2201	Positioning not possible due to mounted EDM
AUT_RC_ANGLE_ERROR	8706	0x2202	Angle measurement error
AUT_RC_MOTOR_ ERROR	8707	0x2203	Motorization error
AUT_RC_INCACC	8708	0x2204	Position not exactly reached
AUT_RC_DEV_ERROR	8709	0x2205	Deviation measurement error
AUT_RC_NO_TARGET	8710	0x2206	No target detected
AUT_RC_MULTIPLE_ TARGETS	8711	0x2207	Multiple target detected
AUT_RC_BAD_	8712	0x2208	Bad environment conditions

RetCodeName	Value	Hex	Description
ENVIRONMENT			
AUT_RC_DETECTOR_ERROR	8713	0x2209	Error in target acquisition
AUT_RC_NOT_ENABLED	8714	0x220A	Target acquisition not enabled
AUT_RC_CALACC	8715	0x220B	ATR-Calibration failed
AUT_RC_ACCURACY	8716	0x220C	Target position not exactly reached
AUT_RC_DIST_STARTED	8717	0x220D	Info: dist. Measurement has been started

BAP 9216 0x2400

RetCodeName	Value	Hex	Description
BAP_CHANGE_ALL_TO_DIST	9217	0x2401	Command changed from ALL to DIST

SAP 9472 0x2500

RetCodeName	Valu	Hex	Description
SAP_ILLEGAL_SYSMENU_NUM	9473	0x2501	Illegal system menu number

COD 9728 0x2600

RetCodeName	Valu	Hex	Description
COD_RC_LIST_NOT_VALID	9728	0x2600	List not initialized.
COD_RC_SHORTCUT_UNKNOWN	9729	0x2601	Shortcut or code unknown.
COD_RC_NOT_SELECTED	9730	0x2602	Codelist selection wasn't possible.
COD_RC_MANDATORY_FAIL	9731	0x2603	Mandatory field has no valid value.
COD_RC_NO_MORE_ATTRIB	9732	0x2604	maximal number of attr. are defined.

BAS	9984	0x2700	
RetCodeName	**Value**	**Hex**	**Description**
BAS_ILL_OPCODE	9984	0x2700	Illegal opcode.
BAS_DIV_BY_ZERO	9985	0x2701	Division by Zero occured.
BAS_STACK_UNDERFLOW	9986	0x2702	Interpreter stack underflow.
BAS_STACK_OVERFLOW	9987	0x2703	Interpreter stack overflow.
BAS_NO_DLG_EXIST	9988	0x2704	No dialog is defined.
BAS_DLG_ALREADY_EXIST	9989	0x2705	Only one dialog may be defined at once.
BAS_INSTALL_ERR	9990	0x2706	General error during installation.
BAS_FIL_INV_MODE	9995	0x270B	Invalid file access mode.
BAS_FIL_TABLE_FULL	9996	0x270C	Maximum number of open files overflow.
BAS_FIL_ILL_NAME	9997	0x270D	Illegal file name.
BAS_FIL_ILL_POS	9998	0x270E	Illegal file position, hence < 1.
BAS_FIL_ILL_OPER	9999	0x270F	Illegal operation on this kind of file.
BAS_MENU_ID_INVALID	10000	0x2710	Invalid menu id detected.
BAS_MENU_TABLE_FULL	10001	0x2711	Internal menu id table overflow.

IOS	10240	0x2800	
RetCodeName	**Value**	**Hex**	**Description**
IOS_CHNL_DISABLED	10240	0x2800	channel is disabled
IOS_NO_MORE_CHAR	10241	0x2801	no more data available
IOS_MAX_BLOCK_LEN	10242	0x2802	reached max. block length
IOS_HW_BUF_OVERRUN	10243	0x2803	hardware buffer overrun (highest priority)
IOS_PARITY_ERROR	10244	0x2804	parity error
IOS_FRAMING_ERROR	10245	0x2805	framing error
IOS_DECODE_ERROR	10246	0x2806	decode error
IOS_CHKSUM_ERROR	10247	0x2807	checksum error (lowest priority)
IOS_COM_ERROR	10248	0x2808	general communication error
IOS_FL_RD_ERROR	10280	0x2828	flash read error
IOS_FL_WR_ERROR	10281	0x2829	flash write error
IOS_FL_CL_ERROR	10282	0x282A	flash erase error

CNF 10496 0x2900			
RetCodeName	**Valu**	**Hex**	**Description**
CNF_INI_NOTOPEN	10497	0x2901	INI-file not opened
CNF_INI_NOTFOUND	10498	0x2902	Warning: Could not find section or key
CNF_CONT	10499	0x2903	Return code of system function
CNF_ESC	10500	0x2904	Return code of system function
CNF_QUIT	10501	0x2905	Return code of system function
CNF_DATA_INVALID	10502	0x2906	Config. file data not valid
CNF_DATA_OVERFLOW	10503	0x2907	Config. file data exceed valid amount
CNF_NOT_COMPLETE	10504	0x2908	Config. file data not complete
CNF_DLG_CNT_ OVERFLOW	10505	0x2909	Too many executed dialogs
CNF_NOT_EXECUTABLE	10506	0x290A	Item not executable
CNF_AEXE_OVERFLOW	10507	0x290B	Autoexec table full
CNF_PAR_LOAD_ERR	10508	0x290C	Error in loading parameter
CNF_PAR_SAVE_ERR	10509	0x290D	Error in saving parameter
CNF_FILE_MISSING	10510	0x290E	Parameter filename/path not valid
CNF_SECTION_MISSING	10511	0x290F	Section in parameter file missing
CNF_HEADER_FAIL	10512	0x2910	Default file wrong or an entry is missing
CNF_PARMETER_FAIL	10513	0x2911	Parameter-line not complete or missing
CNF_PARMETER_SET	10514	0x2912	Parameter-set caused an error
CNF_RECMASK_FAIL	10515	0x2913	RecMask-line not complete or missing
CNF_RECMASK_SET	10516	0x2914	RecMask-set caused an error
CNF_MEASDLGLIST_ FAIL	10517	0x2915	MeasDlgList-line not complete or missing
CNF_MEASDLGLIST_SET	10518	0x2916	MeasDlgList-set caused an error
CNF_APPL_OVERFLOW	10519	0x2917	Application table full

B HARDWARE INTERFACE

B-1 SERIAL INTERFACE

A RS-232 interface is used as a hardware link between the TPS1100 and an external computer.

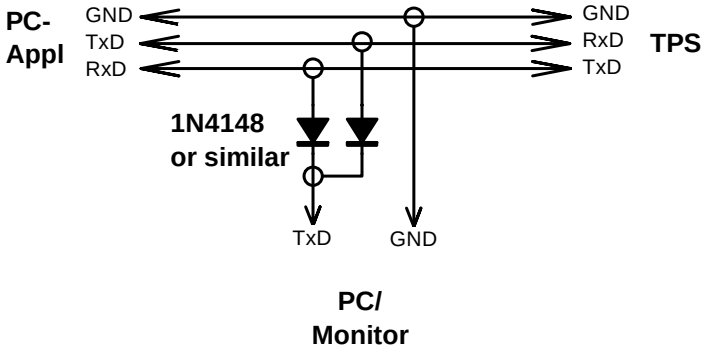
Signal paths	RxD	
	TxD	
	Signal Ground	
Voltage levels	Logical 0 +3V to +25V	
	Logical 1 -3V to -25V	
Baud rate	2400	
	4800	
	9600	
	19200	Default
Parity	None	Fixed
Data bits	8	Fixed
Stop bits	1	Fixed
Terminator	CR/LF	Default

The default settings for the interface are 19200 Baud, 8 data bits, 1 stop bit, no parity. The communication terminator is set to CR/LF. The parameters marked as 'Fixed' may not be changed. The other parameters are variable may be changed by the user.

B-2 DEBUGGING UTILITY

When debugging communicating systems it may be hard to locate the source of an error. Especially in combination with radios to communicate wireless, the number of error sources increases. The following should be checked carefully therefore:

- Are all communication parameters set up properly? Do both participants share the same parameters?
- Have the serial buffer been flushed after opening the serial port? If not and you are using the ASCII protocol then use a leading <LF> to clear the receiver buffer. In the function call protocol you do not need to take care of that.
- When using the ASCII protocol: Is your implementation of the protocol flow indeed synchronous? Or are you sending requests before having received the last reply?
- Are handshake lines for the radios set correctly?
- In case of character errors check shielding of the radio wiring and potential buffer overflow. In case of Windows on 386 and 486 computers, check the UART type. If you do not have a UART with built in buffers (16550 type), you may loose characters too.



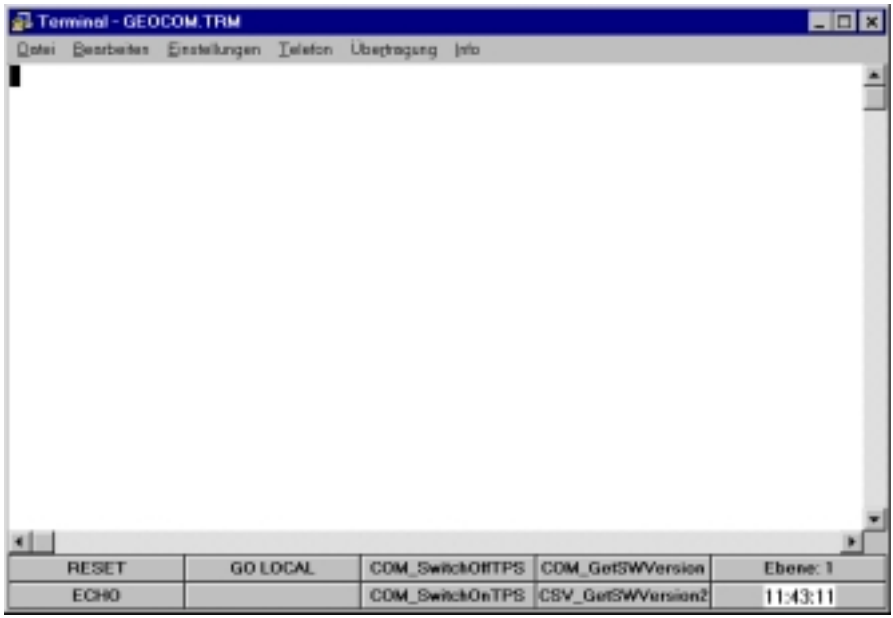
It may be helpful for debugging purposes to build up a special cable to monitor the data transfers.

C PROVIDED SAMPLES

C-1 SETTINGS FOR TERMINAL EMULATOR

To see how ASCII protocol works take a closer look at the provided settings file for the application Terminal.exe.

Use the terminal emulator Terminal.exe which has been provided by Microsoft with Windows 3.1/3.11 and use `geocom.trm` as the set up file for it. If you start Terminal.exe and open `geocom.trm` the following window is displayed:



Connect the TPS1100 instrument to the PC by using a standard GSI connection cable. Start the emulator. `Geocom.trm` has been set up to the default parameters of GeoCOM (19200 Baud, 8 bit, no parity, 1 stop bit). Be sure that the instrument and the terminal emulator use the same settings and switch the instrument to online mode. If not already enabled then enable the function key bar of the application.

Then you can use the emulator to send requests by pressing the assigned buttons. The replies will appear in the emulator's window.

C-2 PROGRAM FRAMES

C-2.1 VBA Sample Program

The sample program shows how simple it is to build an effective application with Visual Basic. The sample program represents a simple measurement task that measures and displays the Hz angle and the V angle continuously. In addition you have the possibility to perform a distance measurement with the following distance measurement programs: single distance standard, single distance fast and tracking.
In order to execute this example program, install MSVB6.0 (or later) on your hard disk and copy the following files in a directory of your choice:

`\SAMPLES\VB\VBSAMPLE.VBP`	Visual Basic Project of the sample.
`\SAMPLES\VB\VBSAMPLE.FRM`	Main form of the sample.
`\SAMPLES\VB\VBSAMPLE_SETUP.FRM`	Communication parameter setup form.
`\SAMPLES\VB\STUBS32P.BAS`	Contains the declarations of the TPS1100 system functions.
`\SAMPLES\VB\GCOM104.DLL`	Contains the implementation of GeoCOM.

Finally connect the TPS1100 Theodolite with the preferred serial port on your personal computer and invoke the executable file. Press the `Setup` button to select the communication parameters (Serial Port, Baudrate, Protocol) and start the application with the button `Go online`. The button `Quit` terminates the application.

C-2.2 C/C++ Sample Programs

The provided sample programs show simple Visual C++ MFC (Microsoft foundation classes) applications. The functionality is exactly the same as in the Visual Basic program above.

The following files have to be copied into a Visual C++ Version 6.0 (or later) working directory in order to build a 32bit application:

`\SAMPLES\VC\GEOCOM_SAMPLE.DSW`	Work space file of the project
`\SAMPLES\VC\*.CPP`	C++ source files

`\SAMPLES\VC\*.H`	C++ header files
`\SAMPLES\VC\GEOCOM_SAMPLE.RC`	Resource file 1
`\SAMPLES\VC\RES\GEOCOM_SAMPLE.RC`	Resource file 2
`\SAMPLES\VC\RES\GEOCOM_SAMPLE.ICO`	Icon file
`\SAMPLES\VC\GCOM104.DLL`	Contains the implementation of GeoCOM
`\SAMPLES\VC\Externals\GCOM104.LIB`	GeoCOM Library
`\SAMPLES\VC\Externals\COM_PUB.HPP`	Header file for GeoCOM

Note: 16 bit GeoCOM version and sample programs are not supported any more.

D LIST OF RPC'S

D-1 ALPHA ORDER

D-2 NUMERIC ORDER

Number	Name	Page