1 Introduction ........................................................................................................ 1-3
2 Installation ........................................................................................................ 2-1
  2.1 Setup ............................................................................................................. 2-1
3 Creating a GeoBASIC Application .................................................................. 3-1
  3.1 GBStudio development environment ............................................................ 3-1
  3.2 Typical Development Cycle ......................................................................... 3-7
  3.3 Project Handling ............................................................................................ 3-3
  3.4 Common Problems ........................................................................................ 3-4
  3.5 Compiler Limitations .................................................................................... 3-5
4 Executing a GeoBASIC Program on the theodolite .................................... 4-6
  4.1 Loading a GeoBASIC program ..................................................................... 4-6
5 Executing a GeoBASIC Program on the Simulator .................................... 5-1
  5.1 General .......................................................................................................... 5-1
  5.2 User Interface ............................................................................................... 5-1
  5.3 Loading and executing GeoBASIC programs .............................................. 5-2
  5.4 Configuration of the Simulator .................................................................... 5-3
  5.5 GeoCom Mode ............................................................................................... 5-4
  5.6 SWTheo Mode ............................................................................................... 5-4
  5.7 Commonly asked questions and answers ..................................................... 5-7
6 Additional Debugging Functions .................................................................... 6-1
7 Multiple Language Support ........................................................................... 7-1
  7.1 Text Utility .................................................................................................. 7-2
8 Typical GeoBASIC Programming .................................................................... 8-1
8.1 The Text Dialog ................................................................. 8-1
8.2 The Graphics Dialog .......................................................... 8-5
8.3 Naming conventions ............................................................. 8-9

9 Refined GeoBASIC Concepts .................................................. 9-1
  9.1 Units ...................................................................................... 9-1
  9.2 The User Measurement Dialog .............................................. 9-2
  9.3 TPS1100 Configurability ....................................................... 9-5
  9.4 Interapplication-Call ............................................................. 9-8
  9.5 System Function Call .......................................................... 9-9
  9.6 System Event Generation ..................................................... 9-10

10 GeoBASIC Sample Programs ............................................... 10-1
  10.1 MeanHz — Mean Value of Horizontal Angle Measurements .... 10-1
  10.2 Sample Programs .............................................................. 10-8

11 Porting a TPS1000 Originated Program .............................. 11-1
  11.1 TPS1100 Hardware Related Changes ................................. 11-1
  11.2 Changes to the Simulator .................................................... 11-2
  11.3 New constructs in GB_1100 ................................................ 11-2
  11.4 GeoBASIC Source Changes ................................................ 11-3

12 GeoBASIC Releases ............................................................. 12-1
  12.1 Changes in GeoBASIC Release 1.30 ................................. 12-1
  12.2 Changes in GeoBASIC Release 2.10 ................................. 12-3
1 INTRODUCTION

GeoBASIC is a programming language for LEICA theodolites and their simulation on personal computers. The core language appears similar to today's common Windows BASIC dialects, thereby it is easy to learn and use. However, GeoBASIC's main power lies in its ability to use many of the existing theodolite subsystems and dialogs, just by calling an appropriate built-in function: for setting parameters, measuring, geodesy mathematics, and many things more. These tools at hand, the programmer can quickly and flexibly build sophisticated geodesy applications.

The user manual first describes the installation of GeoBASIC on a PC (Chapter 2). Then, after learning how to create a GeoBASIC application (Chapter 3), it will be shown how to actually load and execute a program on a LEICA theodolite (Chapter 4) and on the Windows simulation (Chapter 5).

As these technicalities are mastered, the main topic is programming in GeoBASIC. This manual will give you several hints on typical GeoBASIC programming (Chapter 8), and introduces you to the design and programming of the theodolite user interface and refined GeoBASIC concepts (Chapter 9).

Finally, GeoBASIC example programs are presented (Chapter 10). The reader will find a sample code for measuring and computing the mean value of several horizontal angles. Moreover, some introductory examples are given to tell how special problems can be treated.

| Note | All the details of the GeoBASIC language and system functions are composed in the "GeoBASIC Reference Manual". |
2 INSTALLATION

The requirements for using GeoBASIC are a Personal Computer based on an Intel 486 processor or higher and at least 8MB of main memory. The installation of the whole development environment occupies about 10 MB of disk space, excluding the PDF version of the manual. The delivered software needs Microsoft Win95, Win98 or WinNT to run successfully.

2.1 SETUP

The following directory structure is created during the installation per default. Notice that the location of this directory tree is user definable. Hence it is not a granted to be exactly that location. Notice also that the CodeConverter application is installed in a separate Setup installation procedure.

```
...+-SurveyOffice
    |   +--UserTools
    |       |   +--TPS1100Tools
    |       |       |   +--CodeConverter
    |       |       |   +--GBSamples
```

Content of the directories (only the main objects are listed):

- **TPS1100Tools**
  - TPS1100.exe
  - GBStudio.exe
  - GBI_1100_xxx.prg
  - and maybe several more tools, help files or DLL’s

- **CodeConverter**
  - CGB_Dlg.exe
  - Code_ex1.cod
  - GBC_xxx.exe
  - GBI_xxx.prg
  - GBI_1100_xxx.prg
  - CODE to GeoBASIC converter
  - CODE sample
  - GeoBASIC Compiler for TPS1000 series *
  - GeoBASIC Interpreter for TPS1000 series *
  - GeoBASIC Interpreter for TPS1100 series *
• …

Several TPS1100Sim specific directories which contain language files, code lists, configurations and things like that.

* xxx means: i.e. 210 for Release 2.10

**Loading the GeoBASIC Interpreter:**

The GeoBASIC Interpreter will be loaded automatically with the loading of the first application into the theodolite using the Software Upload for TPS1100. Hence you have to copy the GeoBASIC Interpreter (GBI_TPS1100_210.prg) into the same directory as the application before loading it. Otherwise you will get an error message. (For details, please see Chapter 4.1 Loading a GeoBASIC program or 5.3 Loading and executing GeoBASIC programs)
3 CREATING A GEOBASIC APPLICATION

Starting from the specification of a GeoBASIC application, several steps have to be performed until the program can be executed on the theodolite or by simulation:

1. Write the program,
2. compile the program,
3. load the program, either onto the simulation or the theodolite, and
4. start the execution of it.
5. if the execution fails, start a debugging session.

3.1 GBSTUDIO DEVELOPMENT ENVIRONMENT

GBStudio is an integrated development environment and includes a source editor, compiler, project handling and a source level debugger. It is able to debug GeoBASIC 2.10 applications for TPS1100 series total stations. Both, the TPS simulator and the TPS device as the execution platform are supported.
GBStudio contains several views for different purposes. The main source view is for showing/editing source files. The ‘Open Files’-tab can be used to switch quickly between different source windows. Toolbars help the user to start actions with one mouse click. The ‘Build/Output’-window is used to display informative messages of the compiler and during the debugging session for the user.

Use the integrated help system to get more descriptive explanations of what can be done with GBStudio. You can invoke the Help documentation by either using the context-help-cursor (Edit toolbar) or the shortcut F1, which opens the content page.

3.1.1 The Editor

It establishes a modern programming language editor, which supports syntax and keyword highlighting, multilevel undo/redo, Intellisense and Tooltip info, Bookmarks, indent and outdent of a block of source lines, and several other features.

The ‘Workspace Preferences’-dialog can be used to customize the features, which should be active during debugging.
To choose a different font use the ‘Font …’-buttons in the ‘Font’-tab, which will offer a dialog to choose one of the installed fonts on the system. Fonts can be chosen separately for the Editor window, Build/Debug output window and for the Watch Variable window.

3.1.2 The Compiler

The source-file has to be compiled before it can be loaded and executed. Compiling the source file with the GeoBASIC compiler results into 3 files, one for the executable object itself (file extension "*.gba"; i.e. `sample.gba`), one for the language data (file extension "*.lng"; i.e. `sample.lng`) and a debug-info file (file extension "*.gbd"; i.e. `sample.gbd`). The first two files are necessary to execute the program, either on a LEICA theodolite or with the simulator on a personal computer. The debug-info file is necessary for debugging a program using GBStudio. See the following diagram:

```
Diagram: Compiling a GeoBASIC program
```

The compiler is fully integrated in the development environment. The compilation of the source file is just one mouse click away. If an error occurs the editor will place the cursor automatically at the position of the error in the source window. Use Ctrl-F1 to get a more descriptive explanation of what caused the failure of the compilation process.

Depending on the compiler settings also the debug info file is generated which is necessary for debugging the application.
Depending on the selected project type, use either the ‘Default Project Preferences’-dialog or the ‘Project Preferences’-dialog to set the build options for the compiler.

The compiler understands the following options:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>The language on which the resulting application is based on. The default is ENGLISH, other languages are FRENCH, GERMAN, etc.</td>
</tr>
<tr>
<td>Character Set</td>
<td>The character set on which the application is based on. The default character set is 0.</td>
</tr>
<tr>
<td>Output File</td>
<td>The name of the resulting applications file name. If it is empty, the resulting files get the same file base name as the source code file.</td>
</tr>
</tbody>
</table>
Output Path

The path where the compiler places the generated application files. The default is the source directory, where the compiler gets the GeoBASIC source file. The path has to be absolute and has to end with a "\"-character.

Include Paths

Set one or more directory-paths for include files. The directory path must not have a "\" character at the end.

Generate Statistics

Enable this flag, if you want the compiler to generate some statistical information about the compiled application.

Generate Debug Info

Enable this flag, if you want the compiler to generate a debug info file, which is necessary to debug the application.

3.1.3 The Debugger

The debugger enables the programmer to debug GeoBASIC applications at source level. Operations like Step, Step Over, Run, Set breakpoints and watching the values of variables and some more operations are implemented.

To find errors in the source code an error catcher has been implemented which stops the execution of the application once the Err-variable changes its value. The error catching mechanism can be enabled and disabled during the debugging session at the needs of the developer.

The generated files include time stamp information. With this information GBSstudio is able to check if all involved objects are synchronous to each other. This feature also enables GBSstudio to debug an application, which may be in use for some time already. The only precondition, which has to be met is, that all files have to be saved for this purpose. Once the source code file changes debugging can only be started if the application is compiled anew. This means also that the application has to be loaded freshly onto hardware, which then initializes all its values. This feature is very valuable if a tested application shows error only after weeks or months of usage.

Depending on the selected project, use either the “Default Project Preferences” dialog or the “Project Preferences” dialog to set the build options for the debugging session.
For debugging the following values can be set:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>This setting determines the execution platform and if TPS over a serial line is served, which COM port should be used for communication.</td>
</tr>
<tr>
<td>Baud Rate</td>
<td>Is available only if one of the serial communication lines has been chosen. Choose an appropriate Baud rate.</td>
</tr>
<tr>
<td>System Idents</td>
<td>Determines the location of the system specific symbols file. Click on the “Browse…” button to get a file chooser dialog.</td>
</tr>
</tbody>
</table>
Since every loadable application on the TPS may have more than one entry point, one has to select a valid entry point of the application. This value can be entered before the debug info has been loaded, or after the debug info load operation. In the latter case choose the entry point by selecting an item from the drop down list.

<table>
<thead>
<tr>
<th>Entry Point</th>
<th>Since every loadable application on the TPS may have more than one entry point, one has to select a valid entry point of the application. This value can be entered before the debug info has been loaded, or after the debug info load operation. In the latter case choose the entry point by selecting an item from the drop down list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catch Runtime Errors</td>
<td>Enable this flag to catch runtime errors.</td>
</tr>
<tr>
<td>Number of Watch Variables</td>
<td>Select a value between 1 and 1000 watch variables. The number determines the table size on the server side. This value heavily influences the performance of certain debug operations. If you don’t a big number, then choose a smaller number for better performance.</td>
</tr>
<tr>
<td>Size of Shadow Memory</td>
<td>Select a value between 100 and 10000 Bytes. This will be the size of the shadow memory, where the server will keep a backup copy of the registered variables.</td>
</tr>
</tbody>
</table>

### 3.1.4 The Interpreter and the Firmware

Both have been adapted to provide all the additional functionality. Hence only firmware releases 2.10 and newer support GeoBASIC debugging with GBStudio. Please notice, that GBStudio cannot handle the TPS device state “Sleep Mode” correctly. Please disable the sleep feature of the TPS firmware if you want to avoid tedious timeout errors in GBStudio.

### 3.2 TYPICAL DEVELOPMENT CYCLE

#### 3.2.1 Open or Create a GeoBASIC main source file

Use the Open File command to open an existing GeoBASIC main source file or create a new file with the document type GBS.
If you choose to open an already existing project, then the defined main source file should be opened automatically.

### 3.2.2 Edit the application.

Type in or change an existing GeoBASIC application source code. Please, refer also to the GeoBASIC reference manual for a complete description of syntax and semantics of GeoBASIC and how to write applications in GeoBASIC.

The editor is capable of automatically correcting the case of keywords. If one types a blank after a keyword this features take place automatically. Switch this feature off in the Workspace Preferences dialog if you don’t want to use this feature.

CTRL-SPACE opens a drop down list of system-defined functions. This can be used to quickly select a system function. When the opening parenthesis is typed the parameter list will be showed as a tool tip and a reminder what the compiler expects. Use SHIFT-CTRL-SPACE anytime to open up this tool tip again. The displayed parameter list depends on the cursor position and moreover on the system function identifier just before the current cursor.

**Note:** Define also an entry point (GLOBAL SUB definition) of the application, which you can choose later to debug. This is the only identifier in a GeoBASIC application, which is case-sensitive. Make sure this entry point is linked to a menu item on the TPS user interface. Otherwise it will not be possible to debug the application (with the exception of the “BasicCodeProgram” type of application).

Save your changes by using CTRL-S or the Save command from the File menu.

### 3.2.3 Build the application

Press function key F7 or use the Build command from either the Build menu or Build toolbar.
If an error occurs, then the editor will place the cursor automatically near the location of the error. Correct the error and recompile it. Repeat these steps until your application compiles without any errors. Use CTRL-F1 if you want to get some more information on the last error occurred.

**Note**  
The usage of the compiler is protected by a hardware key. Without the right hardware key it is not possible to execute the compiler successfully. If the hardware key is not installed properly or it does not contain the license for the compiler then an error message will be displayed and execution will be terminated.

### 3.2.4 Start debugging

To start the debug session, choose the platform (TPS simulator or TPS instrument) and specific settings, you want to use, in the Project Preferences dialog. Make also sure the entry point of the application is set properly in the preferences dialog.

1. Switch on the debugging platform.
2. When using the TPS device:
   - Load the GeoBASIC interpreter.
3. Load the application you want to debug. (For details, please see Chapter 4.1 Loading a GeoBASIC program or 5.3 Loading and executing GeoBASIC programs)

**Note:**  
The application must have been build with ‘Generate Debug Info’ enabled.
Note: GBStudio uses the TPS device when the GSI settings are active. The GeoCOM online mode is not supported during the debugging process. Make sure the GSI communication settings are:
- 19200 Baud,
- No-Protocol,
- 8 Data Bits,
- No-Parity,
- CR/LF as terminator.
GBStudio cannot handle the sleep state of the TPS device correctly. Make sure the “Sleep after …”-mode is disabled.

The application source and the generated files must be synchronous, hence a source file, which has been changed, after the application has been built, cannot be debugged.

Start debugging by pressing the Start button on the Build toolbar or use the corresponding menu located command.

Start the application on the platform. The editor should now get a small mark (in the shape of an right sided arrow) on the left edge of the main source file window, which points to the very first executable statement of this entry point of the application.

3.2.5 Debugging

Use the commands of the Debug menu or toolbar to step through the application, set breakpoints, catch errors and watch variables as they change during the debugging process.

In the watch variable view you will be able to edit either the identifier of the watch variable entry or the value itself, if the debugging process is in a HALT state.
3.2.6 Stop debugging

Choose the Stop Debugging command to stop the debugging process. Just in case the application is executing a system function, then the debug server will not be able to terminate the application immediately. Instead the application will be terminated after the system call returns. Nevertheless, GBStudio can terminate the debugging session on the client side.

3.2.7 Watch Variables and Quick Watches

Watch variables can be added to the Watch Variable view by selecting a variable identifier and pressing the shortcut Ctrl-W.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR</td>
<td>0</td>
</tr>
<tr>
<td>Wi.iValue</td>
<td>0.000000</td>
</tr>
<tr>
<td>Wi.iDataType</td>
<td>3</td>
</tr>
<tr>
<td>Wi.iValid</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

Use the Quick Watch command if you don’t want to add the variable to the Watch Variable view. Instead the value will be printed into the Debug Output window.

Once added to the window it is possible to change either the identifier name or the value of it (if the point of execution is in the scope of the variable). Use a Double-Click on the identifier or the value to enter the edit mode.
Valid watch variable expressions may be of the following form only:

<table>
<thead>
<tr>
<th>Variable Expression</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VariableIdent</td>
<td>s, Err, line</td>
</tr>
<tr>
<td>StructIdent.Element</td>
<td>CurrPt.dHz, GMCircle.Center.dHeight, ArrayIdent(NumConstant)</td>
</tr>
<tr>
<td></td>
<td>arr(2), field(17,3)</td>
</tr>
</tbody>
</table>

All other possible text strings cannot be handled correctly in the current implementation and will be rejected for registration therefore.

Include exclusively expressions with numerical constants.

### 3.3 PROJECT HANDLING

GBStudio knows two different categories of projects, which are valid exclusively. First the default project, which is valid for any valid GBS-file. And second the so-called 'named' projects, which have the application specific information stored in a file. It should be emphasized that the default project only stores the settings of one project (similar to one main source file) at a time. Once the user chooses another main source file, he has to make sure that the default preferences are set appropriately. E.g., if the two source files have different application entry points, the user has to set it up accordingly.

The default project is active if the user doesn’t choose a project explicitly. Instead the user will just open a plain GeoBASIC source code file.
3.4 COMMON PROBLEMS

The most common problems, which may arise, are:

- GBStudio is not able to establish a connection to the GeoBASIC Debug Server.
  Solution: In the case of debugging with the simulator make sure the TPS simulator is running and “Switched On”. In the case of the TPS device make sure the right COM port has been chosen, the cables are connected and the communication settings are equal on both sides. Notice, that GBStudio only supports serial settings with 8 Bit, 1 Stop Bit, no Parity Bit and CR/LF as a packet terminator. Only the Baud Rate may vary.

- The application, which should be debugged, and/or the interpreter are not loaded.
  Solution: Load interpreter and/or application first, before you start debugging.

- The program source files are out of synchronicity with the compiled application.
  Solution: Recompile and reload the GeoBASIC application.

- The Debug Session cannot be started, because the system predefined symbol file could not be found.
  Solution: Use the “Project Preferences” dialog, Debug-Tab, to specify path and file name of the system predefined symbols.

- The Debug Session cannot be started, because no valid entry point has been chosen.
  Solution: Use the “Project Preferences” dialog, Debug-Tab, to specify a valid entry point. Valid entry points are defined in the source code as “GLOBAL SUB …” procedure names. Notice: the predefined entry points Install, Init and Stop are not valid entry points.

- During debugging a Step-Into an Include source file doesn’t open the source file and show the next statement. Or the compiler reports the error that he can’t open an Include file.
  Solution: Make sure that the “Project Preferences” dialog, Build-Tab, field “Include Directories”, contains the right path, where GBStudio can find the include source file.

- The second registering of a variable doesn’t show the associated value. Notice, a variable can be registered only once.

- During debugging the code source cannot be edited. We disabled this during the debug session to keep the source and the loaded application
synchronous. Stop the debug session to be able to edit the code source again.

- **The debug session hangs.** Conceptually it may happen that a notify message get lost from the server to the client. Then it might be possible that the “Stop Debug” and “Break” buttons are enabled only. Since the debug server has sent the notify message it waits for the next command. And because the client has missed the notification, it thinks the last command is still being under execution and waits for the never incoming notification.
  
  Solution. Use the “Break” button to check the current state. If the last command has been finished and above situation was the reason then this initiate a new notification of the current state.

### 3.5 COMPILER LIMITATIONS

The GeoBASIC programmer has to keep some limitations for his applications:

- One simple procedure or function may not contain more than 10 kB of code.
- The maximum size of an application (including memory space) is limited by the free memory size of the theodolite only. If no other applications are loaded there should be free memory up to several hundred kB on a theodolite.
- An application may not have more than 64kB of string literal in total.
- The number of global identifiers is limited to 3000.
- The overall maximum number of identifiers limits the number of local identifiers, which are about 60000.
4 EXECUTING A GEOBASIC PROGRAM ON THE THEODOLITE

As described in the Chapter 3.1.2 The Compiler, compiling a GeoBASIC program results in at least two files, the executable program itself and the language data. Before a program can be executed, these two files have to be loaded into the theodolite first. With the help of the Leica Survey Office Software Upload the two files can be loaded into TPS-memory and run automatically the install procedure of the GeoBASIC program. The install procedure has to take care of adding an item to a menu which links an external procedure of the GeoBASIC program (Global Sub) to an item in a menu list. Additional to this static link there is a more flexible concept to install an application via a user (definable) configuration. For further explanations how to install an GeoBASIC application read Chapter 9.3. If the menu item is added to a menu you can choose it to run a GeoBASIC program.

4.1 LOADING A GEOBASIC PROGRAM

GeoBASIC programs can be loaded into the theodolite using the Software Upload program from the Open Survey Suite. The procedure for loading a GeoBASIC application is as follows:
1. Verify that a serial link between PC and theodolite is established.
2. Switch theodolite into GeoCOM online mode.
3. Start Software Upload program.
4. Press <Transfer Files...> in <Utilities> menu of Software Upload.
5. Choose <Application Program> as Component Type.
6. Select directory which contains the loadable program (* .gba).
7. Choose language if the application supports multiple languages.
8. Select the application in the <Components> window.

Detailed explanations may be found in the documentation of Leica Survey Office - Software Upload.
GeoBASIC programs can also be loaded from the PC-Card into the theodolite using the build-in application loader. For details, please see description in the theodolite documentation.

**Note**  
Loading a program with identical names for module and external procedures as an already loaded program replaces this program and all its associated text modules in memory and the items in the menu list. Hence, transferring of more than one program with the *same* application name may cause unwanted effects.

**Note**  
For the build-in loader from the PC-Card, the files (*.GBA und *.lng) must be stored in the PC-Card folder “\TPS\APPL”. If necessarily, the GeoBASIC interpreter (gbi_xxx.prg) is loaded automatically from the same folder.
5 EXECUTING A GEOBASIC PROGRAM ON THE SIMULATOR

5.1 GENERAL
The TPS1100 simulation supports, among other features, the execution and debugging of GeoBASIC applications. The simulation may run in one of two modes:

- GeoCOM mode
- SWTheo mode

Running in GeoCOM mode the simulation operates the (hardware) theodolite connected to the PC via a serial port and uses it as a sensor device. In SWTheo mode, user triggered commands are redirected to the software simulation of the theodolite.

5.2 USER INTERFACE
The TPS1100 simulation main window contains two windows and a dialog box on start-up: the "TPS1100" window and the "Debug" window (see below). The TPS1100 window contains a replication of the (hardware) TPS1100 theodolite's user interface. In the "Debug" window, debug information are displayed. It is recommended to have always the debug window opened because some of the statements in the GeoBASIC source code (like the WRITE statement) might cause printing text into the "Debug" window.

The dialog box is called “Virtual Theodolite” and is used to type in raw measurement data for the simulation of measurements. See also section 5.6.2 for further explanations.
5.3 LOADING AND EXECUTING GEOBASIC PROGRAMS

The procedure for loading a GeoBASIC application is as follows:

1. Make sure the simulation is turned on.
2. Choose the „Load Basic Application“ entry from the „File“ menu.
3. Choose a desired GeoBASIC executable (extension .gba) and press the „Open“ button.

If the application could be loaded successfully, it can be executed by choosing the menu item (or in the special case of a code program the CODE button in MEAS-mode), which has been added by the Install routine of the application. There is also a more flexible possibility to install the application via a user (definable) configuration. Refer to Chapter 9.3.2 for more information.

If the menu item “Load Basic Application …“ is disabled (grey) then make sure no GeoBASIC application is running and maybe it’s necessary to press once or twice the ESC button of the TPS simulator.
5.4 CONFIGURATION OF THE SIMULATOR

The simulation is configurable via the „Configuration“ menu of the simulation main window. Here, the beep may be toggled using the „Beep On“ entry. A check mark left to the „Beep On“ indicates whether it is turned on or off. The „Instr. Connection …“ entry opens a dialog to configure the communication parameters for GeoCOM mode and to switch between GeoCOM and SWTheo mode as shown in the following figure.

Paths can be set for text management, GSI data, code list, GeoBASIC programs and configuration data in the dialog opened by the „Data Path“ menu entry.

It is highly recommended to set the paths, if they are not already set, to the following values:

<table>
<thead>
<tr>
<th>Path</th>
<th>Recommended value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Files</td>
<td>TPS1100Tools\TextDB</td>
</tr>
<tr>
<td>GSI and Log Files</td>
<td>TPS1100Tools\GSI</td>
</tr>
<tr>
<td>Internal Code List</td>
<td>TPS1100Tools\CodeList</td>
</tr>
<tr>
<td>External Code List</td>
<td>TPS1100Tools\CodeListPcCard</td>
</tr>
<tr>
<td>Basic Programs Path</td>
<td>TPS1100Tools\GBSamples</td>
</tr>
<tr>
<td>Configuration Data Path</td>
<td>TPS1100Tools\Config</td>
</tr>
</tbody>
</table>
5.5 GEOCOM MODE

5.5.1 Running the simulation in GeoCom mode

To switch to and run in GeoCOM mode follow this procedure:
1. Switch off simulation by single clicking under the down cursor of the TPS1100 window if not already off.
2. Verify that a serial link between PC and theodolite is established.
3. Switch off hardware theodolite if not already off or switch into GeoCOM online mode.
4. Select the appropriate communication parameters and „GeoCom“ in „Instr. Connection …“ dialog (see above) of the simulation. Confirm with the „OK“ button.
5. Start the simulation again using the „ON“ button of the TPS1100 window. The simulation now tries to communicate with the theodolite. If a connection can be established, and the port you have chosen was „COM1“, the title of the TPS1100 window will be „TPS 1100 <running, GeoCom on com1:)>“. Otherwise a dialog enables the user to choose whether other communication configurations should be tested or not. Notice that this may take up to one minute. If no connection could be established, the SWTheo is activated instead of GeoCOM after displaying a message box.

5.6 SWTHEO MODE

The software theodolite (Virtual Theodolite, SWTheo) is an emulation of a (hardware) theodolite. Its properties may be accessed via the „Meas Data Input…“ entry in the „Configuration“ menu while the simulation is running in SWTheo mode. Otherwise this menu entry is disabled.

5.6.1 Running the simulation in SWTheo mode

The procedure for switching to and running the simulation in SWTheo mode is as follows:
1. Switch off the simulation by single clicking under the down cursor of the TPS1100 window if it is not off already.
2. Open the GeoCOM dialog via the „Configuration“ menu.
3. Disable the GeoCOM enable box. Confirm with the „Ok“ button.
4. Start the simulation using the „ON“ button in the TPS1100 window.

5.6.2 User Interface

There are two dialogs to access the SWTheo from the simulation. The first one is called SWTheo dialog with the caption „Virtual Theodolite“ contains fields to change raw sensor data of the SWTheo as well as station data. This dialog is opened from the “Configuration” menu as stated above. The second dialog called SWTheo properties dialog (caption „Virtual Theodolite Properties“) may be triggered from the SWTheo dialog.

5.6.2.1 SWTheo Dialog

The dialog acts as the connection between the SWTheo and its virtual environment. Here, horizontal angle (Hz), vertical angle (V), and slope distance (Dist) to a virtual reflector as well as station data (N0, H0, E0), reflector (Hr) and instrument height (Hi) may be set. User input has to be confirmed using the “Set Data” button to take effect. Pressing the “Properties” button opens the Subsystems dialog.

Notice also that it is possible to define several sets of values. Choose a set by selecting the corresponding number off the measurement set. The values will be stored until they are changed.
5.6.2.2 SWTheo properties dialog

The SWTheo properties dialog is a tabbed dialog as shown below. Here you can set some basic values.

![Virtual Thendolite Properties dialog](image)

The „Units“ tab depicted in the last figure enables the user to choose between several display units for the SWTheo dialogs. Please notice these values do not change the settings of the simulation.

“Jittering” is supported for angles and distances. This functionality is applied by alternately adding and subtracting random values in a range depending on the angle and distance sliders, respectively. The jittering amplitude increases from left to right position of the slider. If the sliders are in their leftmost position, there is no jittering applied to the virtual sensor data.
5.7 COMMONLY ASKED QUESTIONS AND ANSWERS

Q:
After starting the simulation and turning on in SWTheo mode, the text „xxx“ will be displayed as the title of some or all of the function buttons. How can I avoid this problem?

A:
Some or all of the text data base files are not contained in the directory referenced by „Text Management Data Path“. Use the „Data Paths“ entry of the „Configuration“ menu to set it accordingly.

Q:
After loading a GeoBASIC program, the expected menu item does not appear in the dialog. What did I wrong?

A:
The menu manager needs an event to reread the menu definition. Press the ESC key to rebuild the menu.
6 ADDITIONAL DEBUGGING FUNCTIONS

There are a few additional features, which may be helpful while debugging the program.

For the simulator:
- The command `Write` writes the given argument to the debug window. This will have no effects on the TPS.
- The same is valid for `Send`, because it will be redirected to the debug window. But, of course, on TPS it will send data over the data link.
- If an error occurs then a message will be written to the debug window, showing the error code and the name of the system routine, which caused the error.

For the simulator and the TPS:
- `MMI_PrintStr` can be used to display and track results and errors.

See also the list of return codes in the appendix of the Reference Manual.
7 Multiple Language Support

The TPS 1100 series system software supports internationalisation in such a way that text fragments are handled extra to an application. Accessing these fragments will be done internally by tokens. GeoBASIC supports this technique in certain system calls. Anytime a system routine is called which needs a _Token instead of a string then this token will be added to the text token database. The compiler handles this automatically for the programmer and produces the already mentioned lng-file.

This text token database is the basis for supporting multiple languages. With the Text Utility you can produce new text token databases (mxx-files) in other languages. Loading the derived lxx-files on the TPS system for enabling the user to choose between the provided languages. (‘xx’ stands for the language abbreviation.)

Diagram: Generate language files.
Strings which are not passed to a Token parameter can not be handled with the Text Utility. They are hard coded into program object code. The only way to internationalise them is to use MMI_GetLangName to select an appropriate text string in GeoBASIC code separated by a conditional statement.

See sample file "language.gbs".

7.1 TEXT UTILITY

The TPS1100/1000 Text Utility (Text Translation Tool) supports GeoBASIC text files. This section describes the most important steps of generating multiple language files. The following picture shows the Text Utility after the import of a GeoBASIC text file:

7.1.1 Generating new language files

For creating a multiple language application, the following steps are necessary:

1. After starting the Text Utility press the button, select GeoBASIC Text Files (*.l??) in the choice list “File of type:” and open the generated *.lng file (i.e. sample.lng). Answer the question “Do you want to convert this file?” with YES. In the next dialog you can specify the path and the version of the text database which is generated from the *.lng file (i.e. samp1100.mng). The version is automatically included at the end of the file name. Press OK to start the conversion.

2. Press the button, select a language in the choice list “New language”, enter the path of the new language database and press OK to start the
7.1.2 Updating translated language files

After changing the GeoBASIC source file and re-compiling it, the following steps for updating the translated language files are necessary:

1. Press the -button again and open the generated *.lng file (i.e. sample.lng). The version of the text database which is generated must be increased (i.e. samp1101.mng).

2. Press the -button and open the target language you want to update (i.e. samp1100.mge). Edit the target language text column (indicated with T1). After updating the whole column press -button to generate the new loadable language file.
8  TYPICAL GEOBASIC PROGRAMMING

In this chapter some advice is given on how to program in GeoBASIC. The main attention is given to the user dialog — which is probably the most theodolite-specific part in GeoBASIC programming (besides using the system functions). Afterwards a proposal for naming conventions for GeoBASIC identifiers is given.

Note
To make programs easy and intuitive to use, the programmer should follow the given "standards" rather strictly. Moreover (s)he should have a basic understanding of the way how topographical surveying and mapping is actually performed.

8.1  THE TEXT DIALOG

8.1.1  The objects of the text dialog

The following text dialog is not a practical example, it shows only the most important text dialog objects:

Dialog line | Object name
---|---
<BASIC\ Text Dialog Objects> | Caption line: It is composed of the short caption "BASIC" and the caption "Text Dialog Objects".
<I am a text dialog object.> | String
<10587> | Integer value

CONT
8.1.2 Creating a text dialog

A new text dialog is created by `MMI_CreateTextDialog`.

```plaintext
MMI_CreateTextDialog(6, "BASIC", "Text Dialog Objects", "My help text.")
```

A text dialog with a short caption, here "BASIC", and a caption "Text Dialog Objects" is created. There is a total of 27 characters for the three parts, i.e. short caption, separation character (\' printed automatically) and caption. 6 lines (start counting from the first line below the caption – which is 0 – up to line 5) can be used. All lines are empty after the creation. The help text is set to "My help text." — it is shown when the user presses Shift-F1 and the help functionality of the theodolite is enabled.

8.1.3 Representation of the dialog objects

For every input and output the position on the display must be specified. The display is organized in lines and columns. The left upper position has line and column number 0. The line number is rising down and the column number is rising to the right. A display line is 29 characters wide. At most 6 lines are visible at any time, if the dialog contains more lines (up to 12 are possible) it is scrolled when necessary.

For floating point input/output a kind (for instance horizontal angle, distance, etc.) can be specified. Data is automatically transformed to the unit associated to the
kind according to the theodolite settings. Unit conversions are done by the system, all values with units defined in basic are considered to have SI units. (See Chapter 9.1)

All numeric output appears right aligned in their field (specified by coordinates and length). String output appears left aligned.

Each input/output routine needs a parameter lValid which defines if the value of the object is valid or not. If a value is not valid five dashes are displayed instead of the value.

Every numeric input/output needs a parameter iLen which determines the total character length of the field. If the length is too short for the representation of the numeric value, the field will be filled with the character ‘x’.

### 8.1.4 Output in text dialog

- **Strings:**
  
  ```basic
  MMI_PrintStr(0, 0, "I am a text dialog object.", TRUE)
  ```

  **Parameters:** column, line, string, lValid

- **Integer values:**
  
  ```basic
  MMI_PrintInt(10, 1, 10, 10578, TRUE)
  ```

  **Parameters:** column, line, iLen, integer value, lValid

- **Double (floating point) values without unit:**
  
  ```basic
  MMI_PrintVal(10, 2, 10, 3, 90478.568, TRUE, MMI_DEFAULT_MODE)
  ```

  **Parameters:** column, line, iLen, decimals, double value, lValid, Mode

- **Double (floating point) values with unit:**
  
  ```basic
  DIM hz AS Angle
  hz = PI/4
  MMI_PrintVal(10, 3, 8, 3, hz, TRUE, MMI_DIM_ON)
  ```

  **Parameters:** column, line, iLen, decimals, double value, lValid, Mode

### 8.1.5 Input in text dialog

Input is roughly dual to the output, except that the input functions return the button id of the button that terminated the edit process. For all numeric values there are the minimum and maximum values defined. The value is only valid, if it is between them.
• Strings:
  \texttt{MMI\_InputStr(17, 3, 10, sInput, lValid, iButtonId)}
  
  Parameters: column, line, string variable, lValid, button

• Integer values:
  \texttt{MMI\_InputInt(24, 4, 4, 100, 200, iValue, lValid, iButtonId)}
  
  Parameters: column, line, iLen, minimum value, maximum value, integer variable, lValid, button

• Double (floating point) values without unit:
  \texttt{MMI\_InputVal(19, 4, 8, 2, 0, 399.99, MMI\_DEFAULT\_MODE, dValue, lValid, iButtonId)}
  
  Parameters: column, line, iLen, decimals, minimum value, maximum value, mode, double variable, lValid, button

• Double (floating point) values with unit:
  \texttt{MMI\_InputVal(19, 4, 8, 2, 0, 399.99, MMI\_DIM\_ON, dValue, lValid, iButtonId)}
  
  Parameters: column, line, iLen, decimals, minimum value, maximum value, mode, double variable, lValid, button
• List: Lists take a variable of a predefined type as parameter.

   TYPE ListArray (25) AS String30 END

   This definition determines the maximum number of entries in a list to be 25, each one is a string of type String30. We create a list with 4 items and use the second entry as default (initial selection).

   DIM aList AS ListArray
   DIM iIndex AS Integer

   aList(1) = "List Item 1"
   aList(2) = "List Item 2"
   aList(3) = "List Item 3"
   aList(4) = "List Item 4"
   iIndex = 2
   MMI_InputList(8, 4, 12, 4, MMI_DEFAULT_MODE, aList, iIndex, lValid, iButtonId)

   Parameters: column, line, iLen, number of items, mode, list variable, index, lValid, button

8.2 THE GRAPHICS DIALOG

8.2.1 Positioning on the display

   Every graphics function needs the position on the display. The graphics display is organized in x- (horizontal) and y-pixels (vertical). The left upper position has x-pixel and y-pixel number 0. The x-pixel number is rising to the right and the y-pixel number is rising down. The size of the display is 232 times 48 pixels.

8.2.2 Creating a graphics dialog

   Calling MMI_CreateGraphDialog creates a new graphics dialog.

   MMI_CreateGraphDialog("BASIC", "Graphics Dialog", "My help text.")

   A graphics dialog with short caption "BASIC" and caption "Graphics Dialog" is created. The help text is set to "My help text." — it is shown when the user presses Shift-F1 and the help functionality of the theodolite is enabled.
8.2.3 Graphics functions

After having created the graphics dialog, the graphics functions may be used. (E.g. 
MMI_DrawLine, MMI_DrawCircle, MMI_DrawText, etc. See the 
“Reference Manual” for a detailed description.)

8.2.4 Deleting a dialog

When a dialog is not used any more it must be deleted. The name of the dialog 
deletion procedure is for text, measurement and graphics dialogs the same: 

```
MMI_DeleteDialog()
```

8.2.5 Mixing text and graphics dialogs

There can be only one text dialog at a time, i.e. an existing text dialog must be 
deleted with MMI_DeleteDialog before a new one can be created with 
MMI_CreateTextDialog.¹ The same holds for a graphics dialog (with the 
appropriate creation procedures). 

But a graphics dialog may be opened while a text dialog is active. (Note: The 
reverse is not the case: a text dialog may not be opened while a graphics dialog is 
open.) If a text dialog and a graphics dialog are open, the graphics dialog has 
priority, i.e. all future function calls are related to the graphics dialog (until it is 
closed). For example, MMI_AddButton (see below) will add the button to the 
graphics dialog, and all the display functions must be for graphic dialogs (such as 
MMI_DrawCircle, etc.).

8.2.6 Adding buttons

The user may add buttons to a dialog. (These buttons will be added to the defined 
buttons of the dialog.) When adding a button it must be specified what text should 
be displayed for that button. Such a text can be up to five characters long and is 
displayed centred above the button. 

Each button has an identification associated. This button id is needed 

¹ An existing text dialog is deleted automatically if a new text dialog is created.
Typical GeoBASIC Programming

- for specifying which button is to add in `MMI_AddButton`, and
- checking what button was pressed or that is returned from a system function.

**Example:**

We add the F1-button to the currently opened dialog, giving the meaning "CONT" to it.

```
MMI_AddButton( MMI_F1_KEY, "CONT" )
```

**Note**  The button id's are defined as constants in the compiler.

### 8.2.7 Responding to buttons

There are two procedures for coping with button presses:

- `MMI_CheckButton` queries whether there was a button pressed or not, and
- `MMI_GetButton` retrieves a pressed button. If there was no button pressed it waits until one is pressed. The second parameter to `MMI_GetButton` (the in-parameter `bAllKey`) determines what buttons are accepted:
  - If it is TRUE, any button is accepted.
  - If it is FALSE, only ESC, or a defined button (added with `MMI_AddButton`) are accepted.
Example:

The example does some work in a loop until Shift-F6 is pressed. As long as there is no button pressed, the display is constantly updated (e.g. the current angles from the theodolite are displayed). If there is a button pressed, this button is handled.

'bDone must be initialized
bDone = FALSE
DO WHILE NOT bDone
    ' as long as the job is not done
    ' check for defined buttons and get its id
    MMI_GetButton( buttonId, FALSE )
    SELECT CASE buttonId
    CASE MMI_F4_KEY
        ' handle MMI_F4_KEY
        CASE MMI_SHF6_KEY
            bDone = TRUE
            ' that’s it, terminate loop
            ELSE
                ' here go the other handled keys
                ELSE
                    ' here go the unhandled keys
                    END SELECT
                    ' update the display
                    LOOP

8.2.8 Standard key binding

It is clear that for the user it is important that the same name\(^2\) — and moreover the same key — always has the same meaning associated (at least conceptually). An exception is the F1-key, its meaning is not the same in a measurement dialog and in a configuration dialog. In the following table there are the standard key bindings with the caption, the text which is displayed above the keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Caption</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 in measurement dialog</td>
<td>ALL</td>
<td>Does first DIST, then REC. (See below)</td>
</tr>
<tr>
<td>F1 in configuration dialog</td>
<td>CONT</td>
<td>Continues to the logically following dialog.</td>
</tr>
</tbody>
</table>

\(^2\) For instance, the user of a LEICA theodolite assumes that DIST takes the distance (with the common dialogs), ALL does DIST and then REC, etc.
### 8.3 NAMING CONVENTIONS

We propose some naming conventions for GeoBASIC. More extensive conventions can be found in the naming conventions for Microsoft Access (which are tied closely to Visual Basic conventions).

#### 8.3.1 Variable names

Variable names of simple types (i.e. all the scalar types and strings) may be tagged to indicate their type. Prefixes are always lowercase so your eye goes past them to the first uppercase letter — where the base name begins. If the base name consists of more than one word, upper case letters within the name are used to distinguish its parts.

<table>
<thead>
<tr>
<th>Key</th>
<th>Caption</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>DIST</td>
<td>Start distance measurement.</td>
</tr>
<tr>
<td>F3</td>
<td>REC</td>
<td>Records the previously measured / computed data.</td>
</tr>
<tr>
<td>SHIFT-F1</td>
<td>HELP</td>
<td>Displays a help text if the theodolite help functionality is enabled. This key is provided and handled completely by the system, it is not accessible from GeoBASIC.</td>
</tr>
<tr>
<td>SHIFT-F6</td>
<td>QUIT</td>
<td>Terminates an application.</td>
</tr>
<tr>
<td>ESC</td>
<td></td>
<td>Cancels an input or goes a step back. GeoBASIC applications should handle it.</td>
</tr>
<tr>
<td>CODE</td>
<td></td>
<td>Shows the coding dialog.</td>
</tr>
</tbody>
</table>

Note: These naming conventions carry only a semantics for the programmer, not for the compiler.

---

The **base name** succinctly describes the object. For example, `PointNumber` or just `PointNo` for the number of a point. Object **tags** are short abbreviations and simplifications describing the type of the object. For example, the tag ‘`i`’ in `iPointNo` denotes that the type of the variable is `Integer`. The following table lists the tags for the GeoBASIC types.

<table>
<thead>
<tr>
<th>type</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>i</td>
</tr>
<tr>
<td>Logical</td>
<td>l</td>
</tr>
<tr>
<td>Double</td>
<td>d</td>
</tr>
<tr>
<td>Distance</td>
<td>d</td>
</tr>
<tr>
<td>Subdistance</td>
<td>d</td>
</tr>
<tr>
<td>Angle</td>
<td>d</td>
</tr>
<tr>
<td>Pressure</td>
<td>d</td>
</tr>
<tr>
<td>Temperature</td>
<td>d</td>
</tr>
<tr>
<td>String</td>
<td>s</td>
</tr>
</tbody>
</table>

Note that all types which represent floating point numbers are tagged by ‘`d`’. This is because operations valid for the type `Double` are also valid for the other `d`-tagged types.

If there are several similar object names, a **qualifier** may follow the name and further clarify it. For example if we kept two special point numbers, one for the first point and one for the last, the variable names would be the (qualified) variables `iPointNoFirst` and `iPointNoLast`.

**Structure types** do not have a default prefix, if needed the (abbreviated) type name could be used. For **arrays** the base name itself could contain the information that the variable names an array.

For **global variables** an additional prefix ‘`g`’ might be useful.

### 8.3.2 Constants and user-defined types

**Constants** begin with an upper case character. If constants contain only upper case characters (as most of the predefined constants do) the underscore ‘`_`’ is used to separate parts of the name. Often constants can be grouped together, then a prefix is used to denote their common criterion. For example the return codes use `RC`, as in `RC_OK`, `RC_ABORT`, etc.
Mostly constants are globally defined. For local constants an additional prefix 'loc' might be useful.

**User defined types** begin with an upper case character. Use the postfix '_TYPE', '_Type' or 'Type' (according to the naming convention used for the type name itself) appended to the type name to denote that it is a type structure. Alternatively, you can use a prefix 'T'. (For types these conventions are useful since GeoBASIC is not case sensitive. Hence, for example, if there is a type Date no variable can be named date. If the type has the name TDate or Date_Type or DateType, there can.) As for local constants, local types might be prefixed with 'loc'.

### 8.3.3 Procedures

A procedure name begins with an upper case letter and succinctly describes the action that is performed. Variables that denote parameters passed to a function or subroutine (in the parentheses after the function/subroutine name) should be well documented, also indicating whether they act as input, output, or input and output parameters.

### 8.3.4 Keywords

GeoBASIC keywords are all in upper case letters. For example, DIM, FOR, LOOP, FUNCTION, etc.

### 8.3.5 Labels

For error labels (ON ERROR GOTO) we use the function/subprocedure name with the qualifier '_Err' appended.
8.3.6  Remark on naming conventions

Naming conventions never replace the judicious use of comments in your GeoBASIC program code. Naming conventions are an extension of, not a replacement for, good program-commenting techniques.

Formulating, learning, and applying a consistent naming style require a significant initial investment of time and energy. However, you will be amply rewarded when you return to your application a year later to do maintenance or when you share your code with others. Once you implement standardised names, you will quickly grow to appreciate the initial effort you made.

To complete the discussion about naming conventions, we mention the use of program headers:

In every function/subprocedure there should be a header describing, at a minimum, purpose, and parameters passed and/or returned. (In addition there might be comments, the author's name, last revision date, notes, etc.)
9 Refined GeoBASIC Concepts

In GeoBASIC several concepts are implemented to utilise and standardise programming and applications.

9.1 Units

Working with units always gives rise to the problem that different users want to work with different units. In geodesy, take the vertical angle as an example: some surveyors measure in Gon, some in radians, others in percentages. And, in addition to the unit-problem, there is the question where to fix the zero point of some scale. Again for the vertical angle example: some surveyors want to have zenith angles, some nadirs, some something in between.

To cope with this situation there is a fine automatic unit handling system built in the theodolite system, and the GeoBASIC programmer can take full advantage of it. All that has to be done in a GeoBASIC program, is to keep all values in SI units and, when a value has to be displayed specify what kind of value it is: a horizontal angle, a vertical angle, a distance, a temperature, etc. All the formatting, together with choice of the right representation (the user may define this in his theodolite system configuration with which the GeoBASIC programmer is not concerned), and displaying the unit after the value are handled automatically. (Of course the programmer can also decide not to use this automation and handle everything on his own. But values obtained from the system will be in SI units anyway.)

9.1.1 What the GeoBASIC programmer has to do

• Use SI units throughout the program. All computations are done with values in SI units.
• When displaying, specify the correct data type i.e. Distance for the value is displayed. See description of the MMI_PrintVal function in the "Reference Manual".

We will give an example of measuring an horizontal angle, computing the difference to a given angle, and displaying the difference on the display. (Note that we use the GetAngleHz routine from the MeanHz program (see 10.1), and we assume that a text dialog has been opened properly. The angle difference is normalised to the range 0 to 2π.)
Example

```plaintext
DIM dHz1 AS Angle 'first horizontal angle
DIM dHz2 AS Angle 'second horizontal angle
DIM lValidHz2 AS Logical 'indicator if second angle is valid
DIM dDiffHz AS Angle 'the difference of the angles

'assume dHz1 is initialized here to an angle in radians
GetAngleHz( dHz2, lValidHz2 )
dDiffHz = dHz1 - dHz2
GM_AdjustAngleFromZeroToTwoPi( dDiffHz )
MMI_PrintVal( 20, 0, 8, 3, dDiffHz, lValidHz2, MMI_DIM_ON )

The output is as follows:
• If the GetAngleHz routine returned a valid angle, also the difference dDiffHz will be valid (this is why lValidHz2 is used in the MMI_PrintVal function). In this case the angle will be formatted in an 8 character wide field with 3 decimals, afterwards the unit according the theodolite system configuration will be displayed. Assume that gon is set and the angle difference was 1.5473452 radians, then at position 20 in line 0 the output will be « 98,507 g».
• If the angle returned from GetAngleHz was not valid, five dashes will be displayed « ----- g».
```

9.1.2 What the user/surveyor has to do

The user has to set up the units, in which he want to work, in the theodolite system configuration. All outputs that use the theodolite system will automatically be formatted according to this setting.

9.2 THE USER MEASUREMENT DIALOG

The User Measurement Dialog (sometimes referred as MDlg) standardises the visualisation of the measurement values in GeoBASIC. Each value (i.e. vertical angle, horizontal distance) has a predefined output format. Thus the GeoBASIC
programmer has only to define, on which line a value should be displayed. All lines begin with a brief description of the value.

*For example (Output of the horizontal distance):*

```
«Horiz.Dist: 158.287 m»
```

Additionally the measurement parameters and (self-definable) application parameters can be displayed in the measurement dialog. Thus a user is able to change measurement parameters immediately and without leaving the dialog. All measurement values and measurement parameters are saved in the theodolite’s data pool as system parameters.

We distinguish between measurement and application parameters. The former are defined by the system in its meaning and data type. The latter can be defined freely by the user. Please refer to Appendix H in the reference manual for a list of all system and application parameters, which can be used in a measurement dialog.

### 9.2.1 Configuration of the User Measurement Dialog

Before using the measurement dialog we have to define its contents. There are 3 types of possible entries:

- **System parameters:**
  The routine `GSI_SetLineMDlg` places a system parameter (measurement value or measurement settings) on a line.

- **Pure text line:**
  The routine `GSI_SetLineMDlgText` places any text on a line.

- **Application parameters:**
  The routine `GSI_SetLineMDlgPar` places a (self-definable) application parameter on a line.

*Note* The user measurement dialog configuration is automatically initialised with the entries of the first system measurement dialog.

Thus all lines which are not configured by the GeoBASIC programmer shows the same parameters as the first system measurement dialog. For further explanations how to configure the user measurement dialog read the description of the 3 system functions (`GSI_SetLineMDlg`, `GSI_SetLineMDlgText`, `GSI_SetLineMDlgPar`) in the reference manual.
9.2.2 Creating the User Measurement Dialog

After the definition of the content GSI_CreateMDlg analogous to the creation of a text dialog creates the user measurement dialog. For adding buttons to the dialog use MMI_AddButton.

9.2.3 Executing the User Measurement Dialog

In the following example a measurement dialog is created with the horizontal angle on line 2 and the buttons “DIST” on F2-key and “QUIT” on SHIFT-F6-key. All other lines are predefined by the system. After the creation of the dialog the measured values will be updated in a loop:

```plaintext
'Change line 2
GSI_SetLineMDlg(2, GSI_PAR_AngleHz)
GSI_CreateMDlg (2, "MEAS", "Measurement Test", "Measurement Help...")

'Addition of buttons
MMI_AddButton(MMI_F2_KEY, "DIST")
MMI_AddButton(MMI_SHF6_KEY, "QUIT")
lDone = FALSE
DO WHILE NOT lDone
  GSI_UpdateMeasurement(TMC_AUTO_INC, WAITTIME, lRecValid, iCode, FALSE)
  GSI_UpdateMDlg(iButton)
  SELECT CASE iButton
  CASE MMI_F2_KEY
    'DIST Button --> meas a distance and angles
    BAP_MeasDistAngle(iDistMode, dHz, dV, dDist, TRUE, MEAS)
  CASE MMI_SHF6_KEY, MMI_ESC_KEY
    'done --> exit this routinelDone = TRUE
  END SELECT
END WHILE
```

The routine GSI_UpdateMeasurement updates the measurement values in the theodolite data pool. GSI_UpdateMDlg updates the user measurement dialog with the new values and returns the pressed button. For further explanations read the description of these system routines in the reference manual.
If the user measurement dialog is not used any more it must be deleted with `MMI_DeleteDialog`.

See the example program `MEAS.GBS` for a typical usage of the user measurement dialog.

### 9.2.4 Mixing the User Measurement Dialog with Other Dialogs

There can be only one user measurement dialog at a time, i.e. an existing user measurement dialog must be deleted with `MMI_DeleteDialog` before a new one can be created with `GSI_CreateMDlg`. If a user measurement dialog is active, no text dialog can be opened and vice versa.

But a graphics dialog may be opened while a user measurement dialog is active.

**Note**
The reverse is not the case: a user measurement dialog may not be opened while a graphics dialog is open. If a user measurement dialog and a graphics dialog are open, the graphics dialog has priority, i.e. all future function calls are related to the graphics dialog until it is closed.

### 9.3 TPS1100 CONFIGURABILITY

In general, each part of an application, which should be accessible from outside, has to be of the form `GLOBAL SUB`. These points are known as entry points and can be used in two ways. First they can be linked to a menu item (of the a system), and second they can be described as configuration item.

#### 9.3.1 Adding the program in a System Menu

The easier way to access an entry point of an application is to link it to a menu item during the installation phase. Please refer to the Reference Manual `MMI_CreateMenuItem` for further explanations.
9.3.2 Import the program in a User Configuration

The TPS1100 series theodolites support the concept of individual configurations. In a configuration the user can define his own dialogs or menus and link them to certain events (i.e. pressing the PROG key or Power ON). If the event occurs then the linked dialog or the menu will be displayed. The user can create and change his configuration on the PC with the Customisation Tool.

The import of a GeoBASIC program in a user configuration means, that an external GeoBASIC routine is linked with an item of a user defined menu, a button of a user defined dialog or directly with an event. If either the event occurs or the button is pressed or the menu item is selected, then the linked external routine is executed. For the import of a GeoBASIC program the Customisation Tool needs a special file named APPInfo-file with the necessary information about the program.

The usage of the APPInfo-file in the Customisation Tool:

- Start the Customisation Tool
- Open a configuration file, appropriate text- and definition files
- Choose Import Application from the file menu
- Check the box named with the program name (i.e. AppInfoExample)
- Press the OK button

Now the globally accessible subroutines may be added to menus, buttons, etc. simply by using drag and drop.

Generate the APPInfo-file

The APPInfo-file is automatically generated during compilation, if there is a application information (short AppInfo) section in the GeoBASIC source file.

**Note** The AppInfo-section has to occur at the end of the source code. The AppInfo-section is optional; if there is no AppInfo-section in the GeoBASIC source file, the APPInfo-file generation is omitted. The global routine “Install” is optional, since any global routine may be associated with a menu entry, using the APPInfo-file via the Customisation Tool.

The following GeoBASIC sample code illustrates the usage of the AppInfo-section in a GeoBASIC source file. See also the sample program AppInfoTest.gbs.
PROGRAM AppInfoExample

'-------------------------------------------------------
GLOBAL SUB GlobalSub1
    Dim dummy As Integer
    MMI_WriteMsgStr("AppInfoExample.", "GlobalSub1 in AppInfoExample called", MMI_MB_OK, dummy)
END GlobalSub1

'-------------------------------------------------------
GLOBAL SUB GlobalSub2
    Dim dummy As Integer
    MMI_WriteMsgStr("AppInfoExample.", "GlobalSub2 in AppInfoExample called", MMI_MB_OK, dummy)
END GlobalSub2

END AppInfoExample

'Application Information for Config Tool
'---------------------------------------
APPINFO

GENERAL
    SET Author "Leica AG, CH - Heerbrugg"
    SET Desc "AppInfo Example Application"
    SET TheoModel "TCA1100"
END GENERAL

ENTRYPOINT GlobalSub1
    SET CapLg "Global Sub 1"
    SET CapSh "GSUB1"
    SET Desc "test of appinfo subroutine 1"
END GlobalSub1

ENTRYPOINT GlobalSub2
    SET CapLg "Global Sub 2"
    SET CapSh "GSUB2"
    SET Help "displays a message and exits"
END GlobalSub2

END APPINFO

The global subroutines GlobalSub1 and GlobalSub2 are indicated as entry points for the import in a user configuration. Refer to Chapter 2.11 in the Reference Manual for a description of the syntax in BNF-form.
The following figure depicts the whole scenario, from the generation of the AppInfo file over the import in a user (definable) configuration to the loading of the configuration into the theodolite:

![Diagram of GeoBASIC workflow]

### 9.4 INTERAPPLICATION-CALL

The inter-application-call makes it possible to call a subroutine in another GeoBASIC program. With this concept the GeoBASIC programmer can use the same subroutine in several programs.
9.4.1 Definition of a subroutine for Interapplication-Call

If a subroutine should be called by another application, it must be defined as a global subroutine.

Example:

```basic
PROGRAM IAC2
GLOBAL SUB InterAppEntry
DIM iButton AS INTEGER
MMI_WriteMsgStr("Welcome in IAC2","IAC2", MMI_MB_OK, iButton)
END InterAppEntry
END IAC2
```

9.4.2 Call the global subroutine

Before calling the global subroutine, the GeoBASIC programmer has to check with `CSV_LibCallAvailable` if the subroutine is available. That usually means if it is loaded or not. If the subroutine available, he can invoke it with `CSV_LibCall`.

Example:

```basic
DIM lAvailable AS LOGICAL
'Check if global subroutine is available
CSV_LibCallAvailable("IAC2","InterAppEntry", lAvailable)
IF lAvailable
  'available, call global subroutine
  CSV_LibCall("IAC2", "InterAppEntry", "BASIC")
END IF
```

See the example program IAC.GBS and IAC2.GBS for a typical usage of interapplication-call. For further explanations read the description of `CSV_LibCall` and `CSV_LibCallAvailable` in the reference manual.

9.5 SYSTEM FUNCTION CALL

If a theodolite user creates his own configuration on the PC with the Customisation Tool, he has a wide selection of predefined system functions which he can add to menus, buttons, etc. After the loading of the configuration he calls the system functions by selecting the appropriate menu item or button.

The GeoBASIC programmer has the same possibilities. With the routine `CSV_SysCall` he can call the system functions in his programs. Because some system functions do not run on every theodolite type, there is a routine
CSV_SysCallAvailable, which returns if the system function can be executed.

Example:

```
DIM lAvailable AS Logical
CSV_SysCallAvailable(CSV_SFNC_PositCompassDlg, lAvailable)
IF lAvailable
   CSV_SysCall(CSV_SFNC_PositCompassDlg)
END IF
```

If the system function CSV_SFNC_PositCompassDlg can be executed (RCS mode is active), then the dialog RCS orientation with a compass is displayed. For further explanations read the function descriptions of CSV_SysCall and CSV_SysCallAvailable in the reference manual. In Appendix H of the reference manual there is a list of all system functions.

9.6 SYSTEM EVENT GENERATION

Every configuration for a TPS1100 series theodolite is event driven. The user or the system itself generates an event (e.g. the user has pressed the PROG key or the initialisation sequence is finished) and the configuration functionality executes then the linked action (menu, dialog, macro, application or system function).

A GeoBASIC program can generate all events, which can occur in the theodolite system software, also. To generate a system event the same functions can be used as for calling system functions. The routine CSV_SysCall is used for the generation of system events. The routine CSV_SysCallAvailable returns TRUE, if there is an action linked to the requested event and the action can be executed.

Example:

```
DIM lItemDefined AS Logical
CSV_SysCallAvailable(CSV_EFNC_CompensatorSetting, lItemDefined)
IF lItemDefined
   CSV_SysCall(CSV_EFNC_CompensatorSetting)
END IF
```

If a configuration item is defined for the system event CSV_EFNC_CompensatorSetting (compensator setting event; usually connected to a compensator setting dialog) CSV_EFNC_CompensatorSetting is generated and the appropriate system function, application, macro, dialog or menu is
executed. For further explanations read the function description of CSV_SysCall and CSV_SysCallAvailable in the reference manual. In Appendix H of the reference manual there is a list with all system events.
10 GEOBASIC SAMPLE PROGRAMS

10.1 MEANHZ — MEAN VALUE OF HORIZONTAL ANGLE MEASUREMENTS

10.1.1 Program description

The program "MeanHz" measures a number of horizontal angles and computes its arithmetic mean value. The measured angles and the mean angle can then be displayed graphically.

Program flow:
First, the user may enter the number of horizontal angles he wants to measure. (The number of angles must be within a certain range.) Then the angles are measured — each time the REC key is pressed the current horizontal angle is recorded.
As soon as the requested number of angles is measured, the mean angle is computed and displayed. Now the user has the choice either to display the angles graphically, to move the theodolite to the computed mean angle or to quit the program. (The program can be terminated with the ESC button or the QUIT button on shift-F6 at any time.)
Input the number of angles to measure

Measure the angles

Compute and display mean angle

Show Graphics?

Yes

Draw the angles

No

Move theo to mean angle?

Yes

Position theodolite on mean angle

No

End
10.1.2 Source code listing

See example file "meanhz.gbs"

PROGRAM Mean
  
  ' Sample application for building the mean value of angles
  '-----------------------------------------------------------
  ' Measures a user defined number of horizontal angles and calculate
  ' the mean angle. The measured and the mean angle can also be
  ' displayed graphically.
  
  ' GeoBASIC 1.0 for TPS1100 Series Instruments
  ' (c) Leica AG, CH - Heerbrugg 1998
  '-----------------------------------------------------------
  ' Global Declarations
  CONST MaxNoHz = 9 'Maximum number of angles that can be
  'measured
  CONST CaptionShort = "MEAN" 'Short caption (displayed lefthand, in
  'top line)

  'Type to store the angles (for graphics)
  TYPE DIM
    TAngles (MaxNoHz) AS Angle
  END

  DIM fId AS FileId 'File identification

  '-----------------------------------------------------------
  GLOBAL SUB Install
  ' -----
  ' Description
  ' Adds the program into the theodolite's PROG menu. The program's
  ' (application's) name is 'Mean', the global routine to start is
  ' 'Main' and the program menu item will be named 'MEAN HZ'.

    MMI_CreateMenuItem( "Mean", "Main", MMI_MENU_PROGMENU, "MEAN HZ")

  END Install

  SUB RecordValue (dHz As Angle, ByVal dMean As Angle)
    
    ' Description
    ' Writes the value to data link and file.
    
    DIM sVal1 As String30

    '...
DIM sVal2 As String30
DIM sOut As String255

ON Error Resume Next 'Ignore all errors

MMI_FormatVal(MMI_FFORMAT_HZANGLE, 10, 2, dHz, TRUE, MMI_DEFAULT_MODE, sVal1)
MMI_FormatVal(MMI_FFORMAT_HZANGLE, 10, 2, dMean, TRUE, MMI_DEFAULT_MODE, sVal2)
sOut = "hz: " + sVal1 + " mean: " + sVal2 'Compute output text

'Write to data link and file
Send(sOut)
Print(fId, sOut)

END RecordValue

'--------------------------------------------------------------------
SUB GetAngleHz ( dHz AS Angle, lValid AS Logical)
' -------
' Description
' Measures the horizontal angle 'valid' indicates if the dHz is 
' valid.
' Parameters
' IN: dHz.OUT, lValid
' DIM theoAngle AS TMC_Angle_Type 'The measured values
DIM iInfo AS Integer 'Return code

ON Error Resume Next 'Ignore all errors

TMC_GetAngle( theoAngle, iInfo )

IF (Err = RC_OK) THEN
  lValid = TRUE
dHz = theoAngle.dHz
ELSE
  lValid = FALSE
END IF

END GetAngleHz

'--------------------------------------------------------------------
SUB ShowGraphics( byVal iNoPoints AS Integer, angles AS TAngles,
  byVal dMean AS Angle )
  '-------
  ' Description
  ' Displays the measured and the mean horizontal angles 
  ' graphically.
  ' Parameters
  ' IN: iNoPoints, angles, dMean
  ' DIM iX AS Integer 'x coordinate
DIM iY AS Integer 'y coordinate
DIM iButton AS Integer 'button id

CONST CX = 90 'display center x coordinate
CONST CY = 24 'display center y coordinate
CONST DL = 20 'length of line
CONST HELPTEXT = "Visualizes the angles with lines from the station."
   "The computed mean angle is shown by the longer line."
   "The north angle is 0."

MMI_CreateGraphDialog( CaptionShort, "PICTURE", HELPTEXT )

'MMIDraw center and circle
MMI_DrawCircle( CX, CY, 3, 3, MMI_NO_BRUSH, MMI_PEN_BLACK )
MMI_DrawCircle( CX, CY, DL, DL, MMI_NO_BRUSH, MMI_PEN_BLACK )

'Draw lines for angles (there are iNoPoints angles)
DO WHILE iNoPoints > 0
   'compute the line
   iX = INT( DL * SIN(angles(INT(iNoPoints))) )
iY = INT( DL * COS(angles(INT(iNoPoints))) )
   MMI_DrawLine( CX, CY, CX+iX, CY-iY, MMI_PEN_BLACK )
iNoPoints = iNoPoints - 1
LOOP

'Draw line for dMean
iX = INT( (DL+4) * SIN(dMean) )
iY = INT( (DL+4) * COS(dMean) )
MMI_DrawLine( CX, CY, CX+iX, CY-iY, MMI_PEN_DASHED )

'Wait for key press and finish dialog
MMI_AddButton( MMI_F5_KEY, "END" )
MMI_GetButton( iButton, FALSE )
MMI_DeleteDialog()

END ShowGraphics

GLOBAL SUB Main

' Description
' Reads the number of points to be measured. Measures these points,
' calculates the mean value and shows the result or moves (if motorized) the TPS to calculated position.

DIM iNoPoints AS Integer 'number of points to measure
DIM iCurrNo AS Integer 'current point number
DIM lNoOk AS Logical 'TRUE if no of points are valid
DIM lHzOk AS Logical 'TRUE if measured hz is valid
DIM dHz AS Angle 'measured hz
DIM storeHz AS TAngles 'array of measured angles
DIM dMean AS Angle 'calculated mean angle
DIM lKeyPressed AS Logical 'TRUE if button pressed
DIM iButton AS Integer 'id of pressed button
DIM Family AS TPS_Fam_Type 'this data structure is used to store information about the system

ON Error Resume Next 'ignore errors

'check which type of instrument is active and open file
CSV_GetInstrumentFamily( Family )
IF ( Family.lSimulator ) THEN
    Open( "C:\results.txt", "Append", fId, 0 )
ELSE
    Open( "A:\results.txt", "Append", fId, 0 )
END IF

'set up dialog and input iNoPoints
MMI_CreateTextDialog ( 6, "MEAN", "HZ MEAN VALUE", "Compute mean HZ for a number of measurements." )

' *******************************
' * read in iNoPoints *
' *******************************
iNoPoints = 3
lNoOk = TRUE
MMI_PrintStr( 0, 0, "No of points: ", TRUE )
MMI_AddButton( MMI_F1_KEY, "CONT" )
MMI_AddButton( MMI_SHF6_KEY, "QUIT" )
MMI_InputInt( 26, 0, 2, 1, MaxNoHz, MMI_DEFAULT_MODE, iNoPoints, lNoOk, iButton )

'setup rest of dialog
iCurrNo = 1
MMI_PrintStr( 0, 1, "Curr. point: ", TRUE )
MMI_PrintVal( 26, 1, 2, 0, iCurrNo, TRUE, MMI_DEFAULT_MODE )
MMI_PrintStr( 0, 2, "HZ ", TRUE )
MMI_AddButton( MMI_F3_KEY, "REC" )

'init mean value
dMean = 0.0

'get iNoPoints points (abort if ESC or QUIT is pressed)
DO WHILE (iCurrNo <= iNoPoints) AND (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY)
    MMI_PrintVal( 26, 1, 2, 0, iCurrNo, lNoOk, MMI_DEFAULT_MODE )
    MMI_CheckButton( lKeyPressed )
    IF (lKey == MMI_ESC_KEY) OR (lKey == MMI_SHF6_KEY) THEN EXIT DO
    iCurrNo = iCurrNo + 1
    dMean = dMean + dHz
    lNoOk = TRUE
    MMM_PrintVal( 26, 1, 2, 0, iCurrNo, TRUE, MMI_DEFAULT_MODE )
    dMean = dMean / iNoPoints

    MMI_PrintStr( 0, 0, "MEAN ", TRUE )
    MMI_PrintVal( 26, 1, 2, 0, iNoPoints, TRUE, MMI_DEFAULT_MODE )
    MMI_PrintStr( 0, 2, "HZ ", TRUE )
    MMI_PrintVal( 26, 1, 2, 0, iNoPoints, TRUE, MMI_DEFAULT_MODE )
    MMI_PrintStr( 0, 4, "Mean: ", TRUE )
    MMI_PrintVal( 26, 1, 2, 0, dMean, TRUE, MMI_DEFAULT_MODE )
    MMI_PrintStr( 0, 6, "Hz ", TRUE )
    MMI_PrintVal( 26, 1, 2, 0, dMean, TRUE, MMI_DEFAULT_MODE )
    MMI_AddButton( MMI_F2_KEY, "REC" )
    MMI_AddButton( MMI_ESC_KEY, "QUIT" )
    MMI_AddButton( MMI_SHF6_KEY, "QUIT" )
    MMI_InputInt( 26, 0, 2, 1, MaxNoHz, MMI_DEFAULT_MODE, iNoPoints, lNoOk, iButton )
END DO
IF !KeyPressed THEN

  MMI_GetButton( iButton, FALSE )

  SELECT CASE iButton
        CASE MMI_F3_KEY, MMI_F1_KEY
          GetAngleHz( dHz, lHzOk )
          storeHz(iCurrNo) = dHz
          dMean = dMean + dHz
          'if REC pressed record values
          IF iButton = MMI_F3_KEY THEN
            RecordValue(dHz, dMean/iCurrNo)
          END IF
          iCurrNo = iCurrNo + 1
        END SELECT
        ELSE
          'update display
          GetAngleHz( dHz, lHzOk )
          MMI_PrintVal( 20, 2, 8, 3, dHz, lHzOk, MMI_DEFAULT_MODE )
          END IF
  END LOOP

'**************************
'  * show results  *
'**************************

'if execution should proceed
IF (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY) THEN

  'setup new buttons
  MMI_DeleteButton( MMI_F1_KEY )
  MMI_DeleteButton( MMI_F3_KEY )
  MMI_AddButton( MMI_F3_KEY, "SHOW" )
  MMI_AddButton( MMI_F4_KEY, "EXIT" )
  MMI_AddButton( MMI_F5_KEY, "GOTOM" )

  'compute mean value
  dMean = dMean / iNoPoints
  MMI_PrintStr( 0, 3, "Mean Hz : ", TRUE )
  MMI_PrintVal( 20, 3, 8, 3, dMean, TRUE, MMI_DEFAULT_MODE )

  DO WHILE (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY)
    AND (iButton <> MMI_F4_KEY)
    MMI_GetButton( iButton, FALSE )
SELECT CASE iButton
  CASE MMI_F3_KEY
    ShowGraphics( iNoPoints, storeHz, dMean )
    'move theo to the computed mean horizontal angle
  CASE MMI_F5_KEY
    BAP_PosTelescope(BAP_POSIT_HZ, BAP_POS_MSG, dMean, 0, 0.1, 0.1)
END SELECT

  LOOP
  END IF
  'clean up text dialog
  MMI_DeleteDialog()
  'close output file
  Close(fid)

END Main

END Mean

10.2 SAMPLE PROGRAMS

These code samples gives you some help for building your first applications. Each of them should give you some hints in a specific problem domain.

- **appinfotest.gbs** This example shows the use of the application information section in the GeoBASIC source file.
- **codefunc.gbs** An example of a program which will be called, when the Code-key has been pressed.
- **cursor.gbs** Cursor control in a dialog.
- **error_ha.gbs** This program shows how error handling changes execution of a program.
- **language.gbs** Take this program as an example to support multiple language applications. Two language files and its text databases are provided to see how multilingual support works.
- **meanz_gbs** This sample shows the calculation of the mean value of horizontal angle measurements, see Chapter 10.1.
- **meas.gbs** A simple example how to measure with BAP-functions, including Quick-Coding
- **meas_od.gbs** A simple example how to measure and how to record data in an own data-format, including Quick-Coding
- **stringer.gbs** This example shows in which situations typical errors may occur.
- **test.gbs** An empty frame for building up a GeoBASIC application.
- **tracking.gbs** This program shows possible techniques to take advantage of the measurement facilities.
- **menu.gbs** A simple menu handler.
- **dirlist.gbs** This example shows how to get PC card information and how to read a directories content.
- **inclmain.gbs** This example shows the usage of an include file.
- **iac.gbs** An example for an interapplication call.
11 PORTING A TPS1000 ORIGINATED PROGRAM

The implementation of the TPS1100 theodolite series includes several new concepts compared to the firmware of TPS1000 theodolites. To follow up these new concepts and to take care of functionality that has been changed or removed in the implementation of TPS1100 firmware, GeoBASIC programs, once developed for TPS1000 hardware, cannot be compiled without changing the source code. In this chapter we will cover this subject and we try to give some guidelines to help the developer to port the source code onto the new platform. During the design phase of GeoBASIC for TPS1100 systems we took certain care to make the migration as smooth as possible. Although all programs’ source code has to be changed, the effort to port it will be for the most applications not that high.

In the very end this means also that the developer has to maintain two source code bases.

11.1 TPS1100 HARDWARE RELATED CHANGES

11.1.1 Display Line Length

The TPS1100 series instruments use a different liquid crystal display. The difference means also that one can use only 29 characters per line. To be ‘independent’ of the display length we defined the string type DisplayLine. It does not contain the string length in the name, hence this should help in future to port applications. To be compatible with older, TPS1000 GeoBASIC programs we did not change all String30 declarations. Of course only 29 characters will be printed out to the display.

11.1.2 Keyboard

The number of keys has been reduced, there is no CONT-Key any longer. Remove all MMI_CONT_KEY appearances in the source code. We deleted the definition of this constant to make it more obvious to the programmer that he has to change the source code and think about any button assignments.
11.2 CHANGES TO THE SIMULATOR
Now TPSSim supports GeoBASIC programs larger than 64 KB. A restriction, which turned out in the past, bothered the most of the GeoBASIC program developers. We would like to point out that the SWTheo extension enables the programmer to influence the execution of a program. With specific dialogs the programmer gets the possibility to set or change certain (measurement) values. We hope this helps a lot to simulate a more realistic TPS environment and makes it almost obsolete to have an instrument at your hand to test your application. Of course, still the final test of an application has to be done on an instrument. See also the documentation of TPSSim for further explanations.

11.3 NEW CONSTRUCTS IN GB_1100
Due to some requests we added a few new constructs to GeoBASIC for TPS1100 instruments.

11.3.1 #include Statement
It is now possible to include a GeoBASIC source file in another one. Nevertheless only one level of inclusion is allowed.

11.3.2 MID$ statement
Mid$’s implementation has been extended. Now Mid$ can be used to assign a character or a substring to another string at a certain position. In this way single characters of a string can be set or replaced.
Examples:
T = “abcdef”
Mid$(T, 2, 1) = “+” results in “a+cdef”
Mid$(T, 4) = “--------” results in “a+c--------”

11.3.3 Application Info
A general concept of configurability has been introduced for the TPS1100 family of instruments. This gives totally new customisation possibilities into the hand of
the developer and more to the customer support. Up to a certain degree GeoBASIC supports this configurability. For example an assignment of a GeoBASIC program to a menu item can be changed by the new configuration utilities. Or it can be assigned to a function key.

To support these new features we extended the concept of the program by a section that describes the attributes of it.

This (informational) section can be appended optionally at the end of the source file. See the extra explanation of it to get further information about it.

11.4 GEOBASIC SOURCE CHANGES

Many GB programs have a similar structure. Therefore it does not surprise that many programs have to be rewritten in the same way to be compilable and executable for TPS1100 GeoBASIC.

11.4.1 General Dialog Changes

The CONT key does not exist any more on the TPS1100 instruments. Scan your source code for MMI_CONT_KEY and replace it by a function key. The TPS1100 guidelines use MMI_F1_KEY normally for the CONT key functionality. This might make it necessary to change your function key layout. Look at the existing dialogs to get an idea and to be more consistent to the built-in dialogs, to which function keys which functionality has been assigned.

In certain circumstances, where no function keys were left, the ESC key was the only way to leave a dialog. Normally ESC leaves a dialog with leaving values untouched.

MMI_SHIFT_ESC_KEY will not be supported any more. Instead one has to assign QUIT to (normally) Shift-F6. Quit leaves the whole application.

Note 'Old' versions of constants and functions are left aligned. Newer versions or replacements have been shifted to right. The listed changes are ordered in an assumed importance.
Please notice that GB-TPS1000 supports conceptually 2(3) dialogs at once; a text or a graphics dialog and in parallel a customisable measurement dialog - MDlg.

A typical application may create a text dialog and link a graphics dialog to a menu button. Notice, that both dialogs exist at the same time and distinguish this situation from another, where the text dialog will be deleted before the graphical dialog will be created. In the former case one can go back to the text dialog without recreating it. In the latter the text dialog has to be rebuilt. In GB_TPS1100 text and measurement dialog are mutually exclusive.

See the following scheme for a graphical explanation. "()" denotes a dialog.

<table>
<thead>
<tr>
<th>TPS1000</th>
<th>TPS1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_DeleteGraphDialog()</td>
<td>MMI_DeleteDialog()</td>
</tr>
<tr>
<td>MMI_DeleteTextDialog()</td>
<td></td>
</tr>
<tr>
<td>GSI_DeleteMeasDlg()</td>
<td></td>
</tr>
</tbody>
</table>

Graph overrides Text or MDlg. Text and MDlg are mutually exclusive. Only one can be defined at once. All three dialog types may have their own buttons.
### 11.4.2 Recording Format Settings

Deleted:
- GSI_GetRecFormat()
- GSI_SetRecFormat()

Replaced by (extended):
- GSI_GetRecMask()
- GSI_SetRecMask()

### 11.4.3 System Dialog Calls

Replacements for old dialog invocation calls:

<table>
<thead>
<tr>
<th>Old Function</th>
<th>New Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_CommDlg ()</td>
<td>CSV_SysCall (CSV_EFNC_GeoComSetup, Caption)</td>
</tr>
<tr>
<td>GSI_SelectTemplateFiles() and GSI_Setup ()</td>
<td>CSV_SysCall (CSV_EFNC_Setup, Caption)</td>
</tr>
<tr>
<td>GSI_StationData ()</td>
<td>CSV_SysCall (CSV_EFNC_SetStation, Caption)</td>
</tr>
<tr>
<td>GSI_TargetDlg ()</td>
<td>CSV_SysCall (CSV_EFNC_TargetData, Caption)</td>
</tr>
</tbody>
</table>
11.4.4 EDM Mode Changes

Replacement for `EDM_MODE` by the extended `BAP_SetMeasPrg ()`.

<table>
<thead>
<tr>
<th>TMC_GetEDMMode ()</th>
<th>BAP_SetMeasPrg ()</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_SetEDMMode ()</td>
<td>BAP_GetMeasPrg ()</td>
</tr>
</tbody>
</table>

Deleted EDM modes:
- EDM_SINGLE_STANDARD
- EDM_SINGLE_EXACT
- EDM_SINGLE_FAST
- EDM_CONT_STANDARD
- EDM_CONT_EXACT
- EDM_CONT_FAST
- EDM_UNDEFINED

New defined modes:
- BAP_RED_TRK_DIST
- BAP_SINGLE_REF_STANDARD
- BAP_SINGLE_REF_FAST
- BAP_SINGLE_REF_VISIBLE
- BAP_SINGLE_RLESS_VISIBLE
- BAP_CONT_REF_STANDARD
- BAP_CONT_REF_FAST
- BAP_CONT_RLESS_VISIBLE
- BAP_AVG_REF_STANDARD
- BAP_AVG_REF_VISIBLE
- BAP_AVG_RLESS_VISIBLE

11.4.5 Interface Changes

The following routines got a new interface.
- GSI_ImportCoordDlg ()
- GSI_ManCoordDlg ()

Refer to the reference manual to get the new interfaces.

11.4.6 Deleted and Added Identifiers and Types:

<table>
<thead>
<tr>
<th>TPS1000</th>
<th>TPS1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleted:</td>
<td>New:</td>
</tr>
<tr>
<td>CSV_MAX_USERS</td>
<td>CSV_WITH_REFLECTOR</td>
</tr>
<tr>
<td>CSV_ILLEGAL_USERNR</td>
<td>CSVWITHOUT_REFLECTOR</td>
</tr>
<tr>
<td>RC_CSV_ILLEGAL_USERNR</td>
<td></td>
</tr>
</tbody>
</table>

Deleted
- EDM_COMERR
- EDM_NOSIGNAL
<table>
<thead>
<tr>
<th>New:</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_SHIFT_CODE_KEY</td>
<td>MMI_SHIFT_CODE_KEY</td>
</tr>
<tr>
<td>MMI_SET_ANGLE_RELATION()</td>
<td>MMI_SET_ANGLE_RELATION()</td>
</tr>
<tr>
<td>MMI_HANGLE_CLOCKWISE_SOUTH</td>
<td>MMI_HANGLE_CLOCKWISE_SOUTH</td>
</tr>
<tr>
<td>Changed to return code:</td>
<td>Changed to return code:</td>
</tr>
<tr>
<td>MMI_UNDEF_LANG</td>
<td>MMI_UNDEF_LANG</td>
</tr>
<tr>
<td>For MDlg routines:</td>
<td>For MDlg routines:</td>
</tr>
<tr>
<td>MMI_FFORMAT_STRING</td>
<td>MMI_FFORMAT_STRING</td>
</tr>
<tr>
<td>New date format:</td>
<td>New date format:</td>
</tr>
<tr>
<td>MMI_DATE_JP</td>
<td>MMI_DATE_JP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deleted:</th>
<th>New:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_MENU_EXTRA</td>
<td>MMI_MENU_PROGRAMS</td>
</tr>
<tr>
<td>MMI_MENU_CONFIG</td>
<td>MMI_MENU_PROGMEMNU</td>
</tr>
<tr>
<td>*</td>
<td>MMI_MENU_AUTOEXEC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New GSI_ID values:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_SHZ</td>
<td>GSI_ID_SHZ</td>
</tr>
<tr>
<td>GSI_ID_CD_DSC</td>
<td>GSI_ID_CD_DSC</td>
</tr>
<tr>
<td>GSI_ID_PTCD_DSC</td>
<td>GSI_ID_PTCD_DSC</td>
</tr>
<tr>
<td>GSI_ID_PV_CD</td>
<td>GSI_ID_PV_CD</td>
</tr>
<tr>
<td>GSI_ID_PV_PTCD</td>
<td>GSI_ID_PV_PTCD</td>
</tr>
<tr>
<td>GSI_ID_ACT_PTID</td>
<td>GSI_ID_ACT_PTID</td>
</tr>
<tr>
<td>GSI_ID_BACKID</td>
<td>GSI_ID_BACKID</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA0</td>
<td>GSI_ID_APP_DATA0</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA1</td>
<td>GSI_ID_APP_DATA1</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA2</td>
<td>GSI_ID_APP_DATA2</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA3</td>
<td>GSI_ID_APP_DATA3</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA4</td>
<td>GSI_ID_APP_DATA4</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA5</td>
<td>GSI_ID_APP_DATA5</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA6</td>
<td>GSI_ID_APP_DATA6</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA7</td>
<td>GSI_ID_APP_DATA7</td>
</tr>
</tbody>
</table>
### New GSI_POINT_TYPE:
- GSI_BACKSIGHT
- GSI_POINT_CODE

### Deleted:
- TPS1100
- TPS1700
- TPS1800
- TPS5000
- TPS2003

### New:
- TPS1102
- TPS1103
- TPS1105

### Old TPS_FAM_Type:
- iClass
- lEDMBuiltIn
- lEDMTYPEII
- lMotorized
- lATR
- lEGL
- lDBVersion
- lDiodeLaser
- lLaserPlummet
- lSimulator

### New TPS_FAM_Type:
- iClass
- lEDMBuiltIn (always TRUE)
- lEDMTYPEII (always FALSE)
- lEDMTYPEIII (always TRUE)
- lEDMReflectorless
- lMotorized
- lATR
- lEGL
- lDBVersion
- lDiodeLaser
- lLaserPlummet
- lAutoCollimation
- lSimulator

### New:
- BAP_PRISM_MINI
11.4.7 Changes in System Functions

Deleted, because there is no equivalent function at the TPS1100 series instruments:

- BAP_GetFunctionality (), BAP_SetFunctionality ()
- BAP_SetFunctionalityDlg ()
- CSV_GetCurrentUser (), CSV_SetCurrentUser ()
- CSV_GetDL (), CSV_SetDL ()
- CSV_GetUserInstrumentName ()
- CSV_SetUserInstrumentName ()
- CSV_GetUserName (), CSV_SetUserName ()
- GSI_GetStdRecMask ()
- GSI_GetStdRecMaskAll ()
- GSI_GetStdRecMaskCartesian ()

Replaced by equivalent functions:

- GSI_WiDlg ()
- GSI_StartDisplay ()
- GSI_GetStdDialogMask ()

Enhanced in certain ways. See the extended identifiers and constants above or refer to the reference manual:

- WI-values
- CSV_GetPrismType (), CSV_SetPrismType ()
- CSV_GetInstrumentFamily ()
- GetMemoryCardInfo ()
- MMI_GetAngleRelation (), MMI_SetAngleRelation ()
- MMI_SetDateFormat (), MMI_GetDateFormat ()

New functions see reference manual for further details:
Interapplication and system calls
CSV_SysCallAvailable ()
CSV_SysCall ()
CSV_LibCall ()
CSV_LibCallAvailable ()

11.4.8 Returncodes

Their definitions have been coupled totally to the definitions of the TPS1100 firmware. Please refer to the Appendix F in the reference manual for a detailed listing.
12 GEOBASIC RELEASES

12.1 CHANGES IN GEOBASIC RELEASE 1.30

The Release 1.30 of GeoBASIC contains several new subroutines. It reflects user requests and improvements in the TPS1100 Series firmware Release 2.0.

Note: This GeoBASIC Release 1.30 needs at least the TPS1100 Series firmware Release 2.0.

The following paragraph shows the changed items. For a detailed explanation, please see the “GeoBASIC Reference Manual”

12.1.1 New functions in Release 1.30

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SearchPrism</td>
<td>search prism</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>returns if an alternative user task was running (i.e. FNC or PROG was pressed)</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>returns the internal instrument temperature</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>resets the &quot;WasRunning&quot;-flag</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>returns if distance tracking is running</td>
</tr>
<tr>
<td>GSI_ExecQCoding</td>
<td>executes Quick-Coding with/without recording</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>starts a distance measurement after changing the distance mode (new buttons in FNC menu)</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>returns the current measurement display number</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>' returns if a valid code-list for Quick-Coding is selected</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>returns the current recording mask</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>returns the recording order measurement-code or code-measurement block</td>
</tr>
<tr>
<td>GSI_GetWiEntryText</td>
<td>Get coding text-data from the Theodolite data pool</td>
</tr>
</tbody>
</table>
GSI_SelectCode
select a code-list-code, but without recording it
(allows the recording in another format)

GSI_SetMDlgNr
changes the measurement dialog (used i.e. for
>DISP buttons)

GSI_SetQCodeMode
enables Quick-Coding

GSI_SetRecMaskNr
changes the recording mask

GSI_SetRecOrder
defines the recording order

MMI_GetVAngleMode
returns if the V-angle is running (even if a valid
distance is available)

MMI_SetVangleMode
defines the V-angle mode

TMC_GetAtmCorr
gets the atmosphere part of distance measurement
corrections

TMC_GetGeomProjection
gets the projection part of distance measurement
corrections

TMC_GetGeomReduction
gets the reduction to the reference part of
distance measurement corrections

TMC_GetInclineStatus
returns the inclination status (i.e. ready
for recording)

TMC_SetAtmCorr
sets the atmosphere part of distance measurement
corrections

TMC_SetGeomProjection
sets the projection part of distance measurement
corrections

TMC_SetGeomReduction
sets the reduction to the reference part of distance
measurement corrections
12.1.2 New constants in Release 1.30

GSI_GET_NEXT
GSI_MAXDlg_LINES
GSI_MAX_MDLG_MASKS
GSI_MAX_REC_MASKS
GSI_MAX_REC_WI
GSI_MULTI_REC
GSI_NO_FILE_CHANGE
GSI_SEARCH_FROM_END
TPS1101

12.1.3 New datatypes in Release 1.30

HzAngle
VAngle
TMC_GEOM_PROJECTION_Type
TMC_GEOM_REDUCTION_Type
TMC_ATM_TEMPERATURE_Type

12.1.4 New CSV_SysCall constants in Release 1.30

CSV_SFNC_CheckOrientation
CSV_SFNC_CurrentSetPpmDlg
CSV_SFNC_DefSearchAreaDlg
CSV_SFNC_LoadApplDlg
CSV_SFNC_LoadSysLangDlg
CSV_SFNC_SetDefaultSearchRange
CSV_SFNC_ToggleMeasPrgFastRapidTrk
CSV_SFNC_ToggleMeasPrgRefRL
CSV_SFNC_ToggleMeasPrgStdTracking
CSV_SFNC_ToggleSearchArea
CSV_SFNC_ToggleVAngleMode

12.2 CHANGES IN GEOBASIC RELEASE 2.10

The Release 2.10 of GeoBASIC contains the first edition of the integrated development environment GBStudio.

It contains also a few minor bug fixes.
Note: This GeoBASIC Release 2.10 needs at least the TPS1100 Series firmware Release 2.10 or the TPS1100 Series Simulator 2.10.

Note: GeoBASIC applications, compiled with GeoBASIC 1.30, are also executable on the TPS1100 Series firmware Releases 2.10. For running these applications, the GeoBASIC interpreter 1.30 must be loaded. There is no debugging-support for GBStudio! Different Releases of GeoBASIC applications on the same instrument are not supported!
1 Introduction ................................................................................................................. 1-3
2 Installation .................................................................................................................. 2-1
  2.1 Setup ....................................................................................................................... 2-1
3 Creating a GeoBASIC Application ...................................................................... 3-1
  3.1 GeoBASIC IDE ....................................................................................................... 3-1
  3.2 The GeoBASIC Interpreter .................................................................................. 3-6
4 Executing a GeoBASIC Program on the theodolite ........................................ 4-1
  4.1 Loading a GeoBASIC program .............................................................................. 4-1
5 Executing a GeoBASIC Program on the Simulation ...................................... 5-1
  5.1 General .................................................................................................................... 5-1
  5.2 User Interface ........................................................................................................ 5-1
  5.3 GeoCom Mode ...................................................................................................... 5-3
  5.4 SWTheo Mode ...................................................................................................... 5-4
  5.5 Commonly asked questions and answers ............................................................. 5-7
6 Debugging GeoBASIC Programs ........................................................................ 6-1
7 Multiple Language Support ................................................................................. 7-1
  7.1 Text Utility ............................................................................................................. 7-2
8 Typical GeoBASIC Programming ........................................................................ 8-1
  8.1 The Text Dialog ................................................................................................... 8-1
  8.2 The Graphics Dialog ........................................................................................... 8-5
  8.3 Naming conventions ............................................................................................. 8-9
GeoBASIC User Manual 1 — Introduction

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Refined GeoBASIC Concepts</td>
<td>9-1</td>
</tr>
<tr>
<td>9.1</td>
<td>Units</td>
<td>9-1</td>
</tr>
<tr>
<td>9.2</td>
<td>The User Measurement Dialog</td>
<td>9-2</td>
</tr>
<tr>
<td>9.3</td>
<td>TPS1100 Configurability</td>
<td>9-5</td>
</tr>
<tr>
<td>9.4</td>
<td>Interapplication-Call</td>
<td>9-8</td>
</tr>
<tr>
<td>9.5</td>
<td>System Function Call</td>
<td>9-9</td>
</tr>
<tr>
<td>9.6</td>
<td>System Event Generation</td>
<td>9-10</td>
</tr>
<tr>
<td>10</td>
<td>GeoBASIC Sample Programs</td>
<td>10-1</td>
</tr>
<tr>
<td>10.1</td>
<td>MeanHz — Mean Value of Horizontal Angle Measurements</td>
<td>10-1</td>
</tr>
<tr>
<td>10.2</td>
<td>Sample Programs</td>
<td>10-8</td>
</tr>
<tr>
<td>11</td>
<td>Porting a TPS1000 Originated Program</td>
<td>11-1</td>
</tr>
<tr>
<td>11.1</td>
<td>TPS1100 Hardware Related Changes</td>
<td>11-1</td>
</tr>
<tr>
<td>11.2</td>
<td>Changes to the Simulator</td>
<td>11-2</td>
</tr>
<tr>
<td>11.3</td>
<td>New constructs in GB_1100</td>
<td>11-2</td>
</tr>
<tr>
<td>11.4</td>
<td>GeoBASIC Source Changes</td>
<td>11-3</td>
</tr>
<tr>
<td>12</td>
<td>Changes in GeoBASIC Release 1.30</td>
<td>12-1</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

GeoBASIC is a programming language for LEICA theodolites and their simulation on personal computers. The core language appears similar to today's common Windows BASIC dialects, thereby it is easy to learn and use. However, GeoBASIC's main power lies in its ability to use many of the existing theodolite subsystems and dialogs, just by calling an appropriate built-in function: for setting parameters, measuring, geodesy mathematics, and many things more. These tools at hand, the programmer can quickly and flexibly build sophisticated geodesy applications.

The user manual first describes the installation of GeoBASIC on a PC (Chapter 2). Then, after learning how to create an GeoBASIC application (Chapter 3), it will be shown how to actually load and execute a program on a LEICA theodolite (Chapter 4) and on the Windows simulation (Chapter 5).

As these technicalities are mastered, the main topic is programming in GeoBASIC. This manual will give you several hints on typical GeoBASIC programming (Chapter 8), and introduces you to the design and programming of the theodolite user interface and refined GeoBASIC concepts (Chapter 9).

Finally, GeoBASIC example programs are presented (Chapter 10). The reader will find a sample code for measuring and computing the mean value of several horizontal angles. Moreover some introductory examples are given to tell how special problems can be treated.

Note All the details of the GeoBASIC language and system functions are composed in the "GeoBASIC Reference Manual".
2 INSTALLATION

The requirements for using GeoBASIC are a Personal Computer based on an Intel 486 processor or higher and at least 8MB of main memory. The installation of the whole development environment occupies about 10 MB of disk space, excluding the PDF version of the manual. The delivered software needs Microsoft Win95, Win98 or WinNT to run successfully.

2.1 SETUP

The following directory structure is created during the installation per default. Notice that the location of this directory tree is user definable. Hence it is not a granted to be exactly that location. Notice also that the CodeConverter application is installed in a separate Setup installation procedure.

```
...+-SurveyOffice
    |    
    +-UserTools
        |    
        +-TPS1100Tools
            |    
            + - CodeConverter
            |    
            + - GBSamples
```

Content of the directories (only the main objects are listed):

- **TPS1100Tools**
  - `TPS1100.exe` - TPS Simulator for TPS1100 Series
  - `GB_IDE.exe` - GeoBASIC IDE application
  - `GBI_1100_101.prg` - GeoBASIC Interpreter for TPS1100 series
  - ... and maybe several more tools

- **CodeConverter**
  - `CGB_Dlg.exe` - CODE to GeoBASIC converter
  - `Code_ex1.cod` - CODE sample
  - `GBC_229.exe` - GeoBASIC Compiler for TPS1000 series
  - `GBI_229.prg` - GeoBASIC Interpreter for TPS1000 series
  - `GBI_1100_101.prg` - GeoBASIC Interpreter for TPS1100 series
Several TPS1100Sim specific directories which contain language files, code lists, configurations and things like that.

**Loading the GeoBASIC Interpreter:**

The GeoBASIC Interpreter will be loaded automatically with the loading of the first application into the theodolite using the Software Upload for TPS1100. Hence you have to copy the GeoBASIC Interpreter (GBI_TPS1100_101.prg) into the same directory as the application before loading it. Otherwise you will get an error message.
3 CREATING A GEOBASIC APPLICATION

Starting from the specification of a GeoBASIC application, several steps have to be performed until the program can be executed on the theodolite or by simulation:
1. Write the program,
2. compile the program,
3. load the program, either onto the simulation or the theodolite, and
4. start the execution of it.

3.1 GEOBASIC IDE

While processing step 1 (write the program) and step2 (compile the program) the programmer is supported by the windows tool GeoBASIC IDE (Integrated Development Environment).

3.1.1 Writing a GeoBASIC source-file

The GeoBASIC IDE offers a simple text editor, with it the programmer can work on the source-files directly without using an external editor. After starting the GeoBASIC IDE application select the NEW-button (        ) to create a new source-file (i.e. sample.gbs) or the OPEN-button (          ) to change an existing one. The usage of the IDE editor is identical to the most Windows text editors. See the next picture of the IDE of how it looks like.
3.1.2 Compiling a GeoBASIC program

The source-file has to be compiled before it can be loaded and executed. Compiling the source file with the GeoBASIC compiler results into two files, one for the executable object itself (file extension ".gba"; i.e. sample.gba) and one for the language data (file extension ".lng"; i.e. sample.lng). These two files are necessary to execute the program, either on a LEICA theodolite or with the simulator on a personal computer. See the following diagram:

```
Diagram: Compiling a GeoBASIC program
```

```
Sample gbs

<table>
<thead>
<tr>
<th>GB-compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample.gba</td>
</tr>
<tr>
<td>sample.lng</td>
</tr>
</tbody>
</table>
```
The GeoBASIC compiler is integrated in GeoBASIC IDE. The following picture shows the IDE after a successful compilation of Meanhz.gbs:

The Compilation is started either by selecting <Compile Program> in the <Compile>-menu, pressing Ctrl+F7 or clicking COMPILE-button ( ). In any case the selected window determines which source-file has to be compiled.

During the compilation process the compiler checks for a correct program. If the compiler recognises an error it produces an error message in the output window and the compilation is stopped. The following window shows a stop during compilation of Test.gbs because of the undefined identifier “Rectangle”:

In the output window the line (i.e. 33) and column (i.e. 18) of the program, where the error occurred, is displayed. Additional the cursor is moved on this position in the program. The error identification number (i.e. 61) references to further explanations. Set cursor on the line with the error number and use the shortcut <Shift-F1> to get a more detailed explanation of the error. Select <How To Use> in the <Help>-menu for a list of all error codes and a detailed information about the whole IDE functionality.

In the case that a semantic condition could not be met the line and column position might be not correct. E.g. the source of lines 18 and 19:
generates the following error message in the output window:

C:\GeoBASIC\Samples\Meanhz.gbs(19)(3): error 25, type mismatch

This seems to be not correct but it’s a follow-up of the fact that the semantic information is available only if the last statement is processed to the end of it. Hence the next symbol has been already got from the input symbol stream. Therefore, the symbol pointer points to the next symbol. In our example it is the call of a system subroutine. Be aware of this fact if you track back an error.

The GeoBASIC programmer has to keep some limitations for his applications:

• One simple procedure or function may not contain more than 10 kB of code.
• The maximum size of an application (including memory space) is limited by the free memory size of the theodolite only. If no other applications are loaded there should be free memory up to several hundred kB on a theodolite.
• An application may not have more than 64kB of string literal in total.
• The number of global identifiers is limited to 3000.
• The overall maximum number of identifiers limits the number of local identifiers, which is about 60000.

Note

The usage of the compiler is protected by a hardware key. Without the right hardware key it is not possible to execute the compiler successfully. If the hardware key is not installed properly or it does not contain the license for the compiler then an error message will be displayed and execution will be terminated.

Compile Options

The Selection of <Compile Options> in the <Compile>-menu displays the following dialog box:
An GeoBASIC programmer has to make the following settings before the first compilation:

- **Language:**
  Set the application’s language. Default is ENGLISH.

- **Character Set:**
  Set the application’s character set. Default is 0.

- **Output File:**
  Set the name of the resulting applications file name. If it is empty, the resulting files get the same name as the source-file but with different file-extensions (normally).

- **Output Path:**
  Set the path where the compiler places the generated application files. The default is the source directory, where the compiler gets the GeoBASIC source program. The path has to be absolute and has to end with a "\" character.

- **Include Path:**
  Set one or more directory-paths of include files. The directory path must not have a "\" character at the end.

The IDE is capable to remember the last settings and the opened files. They will be restored/reopened at the next start.

**Statistics**

If the **Statistics**-item in the **Compile**-menu is checked the compiler will generate statistical information about the application which will be printed into the Output window.
The following information will be given:

- **Tokens**: Number of Tokens of the text database. They will be written into the *.lng-file.
- **Globals**: Number of global objects, for example data types, subroutines, and so on.
- **GlobalMem**: Maximum global memory needed during runtime.
- **LocalMem**: Maximum local memory needed during runtime per application invocation.
- **CodeLen**: Length of produced code, excluding the string table.

The total of all memory sizes will give the size of the necessary memory to run the application.

**Note**
Your GeoBASIC source files must have been compiled without errors in order to be loadable.

### 3.2 THE GEOBASIC INTERPRETER

The GeoBASIC interpreter is a program that "understands" the compiler-generated object file and executes it. In the windows simulation, the interpreter is already included. In the theodolite however, the interpreter will be loaded automatically with the loading of the first application into the theodolite using the Software Upload for TPS1100. (Hence the interpreter must be in the same directory as the application.)
4 EXECUTING A GEOBASIC PROGRAM ON THE THEODOLITE

As described in the Chapter 3.1.2, compiling a GeoBASIC program results in two files, the executable program itself and the language data. Before a program can be executed, these two files have to be loaded into the theodolite first. With the help of the Leica Survey Office Software Upload the two files can be loaded into TPS-memory and run automatically the install procedure of the GeoBASIC program. The install procedure has to take care of adding an item to a menu which links an external procedure of the GeoBASIC program (Global Sub) to an item in a menu list. Additional to this static link there is a more flexible concept to install an application via a user (definable) configuration. For further explanations how to install an GeoBASIC application read Chapter 9.3.

If the menu item is added to a menu you can choose it to run a GeoBASIC program.

4.1 LOADING A GEOBASIC PROGRAM

GeoBASIC programs can be loaded into the theodolite using the Software Upload program from the Open Survey Suite. The procedure for loading a GeoBASIC application is as follows:

1. Verify that a serial link between PC and theodolite is established.
2. Switch theodolite into GeoCOM online mode.
3. Start Software Upload program.
4. Press <Transfer Files...> in <Utilities> menu of Software Upload.
5. Choose <Application Program> as Component Type.
6. Select directory which contains the loadable program (*.gba).
7. Choose language if the application supports multiple languages.
8. Select the application in the <Components> window.

Detailed explanations may be found in the documentation of Leica Survey Office - Software Upload.
Note

Loading a program with identical names for module and external procedures as an already loaded program replaces this program and all its associated text modules in memory and the items in the menu list. Hence, transferring of more than one program with the *same* application name may cause unwanted effects.
5 EXECUTING A GEOBASIC PROGRAM ON THE SIMULATION

5.1 GENERAL
The TPS1100 simulation supports, among other features, the execution and debugging of GeoBASIC applications. The simulation may run in one of two modes:

- GeoCOM mode
- SWTheo mode

Running in GeoCOM mode the simulation operates the (hardware) theodolite connected to the PC via a serial port and uses it as a sensor device. In SWTheo mode, user triggered commands are redirected to the software simulation of the theodolite.

5.2 USER INTERFACE
The TPS1100 simulation main window contains two windows and a dialog box on start-up: the "TPS1100" window and the "Debug" window (see below). The TPS1100 window contains a replication of the (hardware) TPS1100 theodolite’s user interface. In the "Debug" window, debug information are displayed. It is recommended to have always the debug window opened because some of the statements in the GeoBASIC source code (like the WRITE statement) might cause printing text into the "Debug" window.

The dialog box is called “Virtual Theodolite” and is used to type in raw measurement data for the simulation of measurements. See also section 5.4.3 for further explanations.
The simulation is configurable via the “Configuration” menu of the simulation main window. Here, the beep may be toggled using the “Beep On” entry. A check mark left to the “Beep On” indicates whether it is turned on or off. The “Instr. Connection …” entry opens a dialog to configure the communication parameters for GeoCOM mode and to switch between GeoCOM and SWT Theo mode as shown in the following figure.
Paths can be set for text management, GSI data, code list, GeoBASIC programs and configuration data in the dialog opened by the „Data Path“ menu entry. It is highly recommended to set the paths, if they are not already set, to the following values:

<table>
<thead>
<tr>
<th>Path</th>
<th>Recommended value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Files</td>
<td><code>TPS1100Tools\TextDB</code></td>
</tr>
<tr>
<td>GSI and Log Files</td>
<td><code>TPS1100Tools\GSI</code></td>
</tr>
<tr>
<td>Internal Code List</td>
<td><code>TPS1100Tools\CodeList</code></td>
</tr>
<tr>
<td>External Code List</td>
<td><code>TPS1100Tools\CodeListPcCard</code></td>
</tr>
<tr>
<td>Basic Programs Path</td>
<td><code>TPS1100Tools\GBSamples</code></td>
</tr>
<tr>
<td>Configuration Data Path</td>
<td><code>TPS1100Tools\Config</code></td>
</tr>
</tbody>
</table>

5.3 **GEOCOM MODE**

5.3.1 **Running the simulation in GeoCom mode**

To switch to and run in GeoCOM mode follow this procedure:

1. Switch off simulation by single clicking under the down cursor of the TPS1100 window if not already off.
2. Verify that a serial link between PC and theodolite is established.
3. Switch off hardware theodolite if not already off or switch into GeoCOM online mode.
4. Select the appropriate communication parameters and „GeoCom“ in „Instr. Connection …“ dialog (see above) of the simulation. Confirm with the „OK“ button.
5. Start the simulation again using the „ON“ button of the TPS1100 window. The simulation now tries to communicate with the theodolite. If a connection can be established, and the port you have chosen was „COM1“; the title of the TPS1100 window will be „TPS 1100 <running, GeoCom on com1:>“. Otherwise a dialog enables the user to choose whether other communication configurations should be tested or not. Notice that this may take up to one minute.
If no connection could be established, the SWTheo is activated instead of GeoCOM after displaying a message box.

5.4 SWTHEO MODE

The software theodolite (Virtual Theodolite, SWTheo) is an emulation of a (hardware) theodolite. Its properties may be accessed via the „Meas Data Input …“ entry in the „Configuration“ menu while the simulation is running in SWTheo mode. Otherwise this menu entry is disabled.

5.4.1 Running the simulation in SWTheo mode

The procedure for switching to and running the simulation in SWTheo mode is as follows:

1. Switch off the simulation by single clicking under the down cursor of the TPS1100 window if it is not off already.
2. Open the GeoCOM dialog via the „Configuration“ menu.
3. Disable the GeoCOM enable box. Confirm with the „Ok“ button.
4. Start the simulation using the „ON“ button in the TPS1100 window.

5.4.2 Loading and executing GeoBASIC programs

The procedure for loading a GeoBASIC application is as follows:

1. Make sure the simulation is turned on.
2. Choose the „Load Basic Application“ entry from the „File“ menu.
3. Choose a desired GeoBASIC executable (extension .gba) and press the „Open“ button.

If the application could be loaded successfully, it can be executed by choosing the menu item (or in the special case of a code program the CODE button in MEAS-mode), which has been added by the Install routine of the application. There is also a more flexible possibility to install the application via a user (definable) configuration. Refer to Chapter 9.3.2 for more information.

If the menu item “Load Basic Application …” is disabled (grey) then make sure no GeoBASIC application is running and maybe it’s necessary to press once or twice the ESC button of the TPS simulator.
5.4.3 User Interface

There are two dialogs to access the SWTheo from the simulation. The first one is called SWTheo dialog with the caption „Virtual Theodolite“ contains fields to change raw sensor data of the SWTheo as well as station data. This dialog is opened from the “Configuration” menu as stated above. The second dialog called SWTheo properties dialog (caption „Virtual Theodolite Properties“) may be triggered from the SWTheo dialog.

5.4.3.1 SWTheo Dialog

The dialog acts as the connection between the SWTheo and its virtual environment. Here, horizontal angle (Hz), vertical angle (V), and slope distance (Dist) to a virtual reflector as well as station data (N0, H0, E0), reflector (Hr) and instrument height (Hi) may be set. User input has to be confirmed using the “Set Data” button to take effect. Pressing the “Properties” button opens the Subsystems dialog.

![SWTheo Dialog](image)

Notice also that it is possible to define several sets of values. Choose a set by selecting the corresponding number off the measurement set. The values will be stored until they are changed.

5.4.3.2 SWTheo properties dialog

The SWTheo properties dialog is a tabbed dialog as shown below. Here you can set some basic values.
The „Units“ tab depicted in the last figure enables the user to choose between several display units for the SWTheo dialogs. Please notice these values do not change the settings of the simulation.

“Jittering” is supported for angles and distances. This functionality is applied by alternately adding and subtracting random values in a range depending on the angle and distance sliders, respectively. The jittering amplitude increases from left to right position of the slider. If the sliders are in their leftmost position, there is no jittering applied to the virtual sensor data.
5.5 COMMONLY ASKED QUESTIONS AND ANSWERS

Q: After starting the simulation and turning on in SWTheo mode, the text “xxx” will be displayed as the title of some or all of the function buttons. How can I avoid this problem?

A: Some or all of the text data base files are not contained in the directory referenced by “Text Management Data Path”. Use the “Data Paths” entry of the “Configuration” menu to set it accordingly.

Q: After loading a GeoBASIC program, the expected menu item does not appear in the dialog. What did I wrong?

A: The menu manager needs an event to reread the menu definition. Press the ESC key to rebuild the menu.
6 DEBUGGING GEOBASIC Programs

The debugging facilities of the GeoBASIC development environment are somewhat limited. Although, there are a few features, which may be helpful while debugging the program.

For the simulator:

- The command `Write` writes the given argument to the debug window. This will have no effects on the TPS.
- The same is valid for `Send`, because it will be redirected to the debug window. But, of course, on TPS it will send data over the data link.
- If an error occurs then a message will be written to the debug window, showing the error code and the name of the system routine, which caused the error.

For the simulator and the TPS:

- `MMI_PrintStr` can be used to display and track results and errors.

See also the list of return codes in the appendix of the Reference Manual.
7  **MULTIPLE LANGUAGE SUPPORT**

The TPS 1100 series system software supports internationalisation in such a way that text fragments are handled extra to an application. Accessing these fragments will be done internally by tokens. GeoBASIC supports this technique in certain system calls. Anytime a system routine is called which needs a `Token` instead of a string then this token will be added to the text token database. The compiler handles this automatically for the programmer and produces the already mentioned `lng`-file.

This text token database is the basis for supporting multiple languages. With the Text Utility you can produce new text token databases (`mxx`-files) in other languages. Loading the derived `lxx`-files on the TPS system for enabling the user to choose between the provided languages. ('xx' stands for the language abbreviation.)

![Diagram: Generate language files.](image)
Strings which are not passed to a \_Token parameter can not be handled with the Text Utility. They are hard coded into program object code. The only way to internationalise them is to use MMI\_GetLangName to select an appropriate text string in GeoBASIC code separated by a conditional statement.

See sample file "language.gbs".

### 7.1 TEXT UTILITY

The TPS1100/1000 Text Utility (Text Translation Tool) supports GeoBASIC text files. This section describes the most important steps of generating multiple language files. The following picture shows the Text Utility after the import of a GeoBASIC text file:

#### 7.1.1 Generating new language files

For creating a multiple language application, the following steps are necessary:

1. After starting the Text Utility press the \_button, select GeoBASIC Text Files (*.l??) in the choice list “File of type:” and open the generated *.lng file (i.e. sample.lng). Answer the question “Do you want to convert this file?” with YES. In the next dialog you can specify the path and the version of the text database which is generated from the *.lng file (i.e. samp1100.mng). The version is automatically included at the end of the file name. Press OK to start the conversion.

2. Press the \_button, select a language in the choice list “New language”, enter the path of the new language database and press OK to start the
generation of the new language database (i.e. sampl100.mge). Now translate the text in column “Text”.

**Note**  Do not edit the first token with the text “\"X"\". This string is needed by the GeoBASIC Interpreter. Also the special strings for MMI_INVERSE_ON (“\"R+\"”) and MMI_INVERSE_OFF (“\"R-\"”) must be left unchanged.

After the translation press the -button, select the path and enter the name of the loadable language file and press OK to start the generation of the file (i.e. sample.lge).

### 7.1.2 Updating translated language files

After changing the GeoBASIC source file and re-compiling it, the following steps for updating the translated language files are necessary:

1. Press the -button again and open the generated *.lng file (i.e. sample.lng). The version of the text database which is generated must be increased (i.e. sampl101.mng).

2. Press the -button and open the target language you want to update (i.e. sampl100.mge). Edit the target language text column (indicated with T1). After updating the whole column press -button to generate the new loadable language file.
8 TYPICAL GEOBASIC PROGRAMMING

In this chapter some advice is given on how to program in GeoBASIC. The main attention is given to the user dialog — which is probably the most theodolite-specific part in GeoBASIC programming (besides using the system functions). Afterwards a proposal for naming conventions for GeoBASIC identifiers is given.

Note

To make programs easy and intuitive to use, the programmer should follow the given "standards" rather strictly. Moreover (s)he should have a basic understanding of the way how topographical surveying and mapping is actually performed.

8.1 THE TEXT DIALOG

8.1.1 The objects of the text dialog

The following text dialog is not a practical example, it shows only the most important text dialog objects:

```
Dialog line Object name
<BASIC Text Dialog Objects> Caption line: It is composed of the short caption "BASIC" and the caption "Text Dialog Objects".
<I am a text dialog object.> String
10587 Integer value
90479.568
50.000 g
List Item 1
CONT |
```

Double (floating point) value without unit
Double (floating point) value with unit: If the type of the double value is Angle, Distance, Subdistance, etc., the according unit is printed automatically
List: It is for selecting an item among several with the cursor keys
Button: The buttons inform the user about the functionality of the function key (F1..F6).

8.1.2 Creating a text dialog

A new text dialog is created by `MMI_CreateTextDialog`.

```plaintext
MMI_CreateTextDialog(6, "BASIC", "Text Dialog Objects", "My help text.")
```

A text dialog with a short caption, here "BASIC", and a caption "Text Dialog Objects" is created. There is a total of 27 characters for the three parts, i.e. short caption, separation character (\' printed automatically) and caption. 6 lines (start counting from the first line below the caption – which is 0 – up to line 5) can be used. All lines are empty after the creation. The help text is set to "My help text." — it is shown when the user presses Shift-F1 and the help functionality of the theodolite is enabled.

8.1.3 Representation of the dialog objects

For every input and output the position on the display must be specified. The display is organized in lines and columns. The left upper position has line and column number 0. The line number is rising down and the column number is rising to the right. A display line is 29 characters wide. At most 6 lines are visible at any time, if the dialog contains more lines (up to 12 are possible) it is scrolled when necessary.

For floating point input/output a kind (for instance horizontal angle, distance, etc.) can be specified. Data is automatically transformed to the unit associated to the
kind according to the theodolite settings. Unit conversions are done by the system, all values with units defined in basic are considered to have to SI units. (See Chapter 9.1)

All numeric output appears right aligned in their field (specified by coordinates and length). String output appears left aligned.

Each input/output routine needs a parameter `lValid` which defines if the value of the object is valid or not. If a value is not valid five dashes are displayed instead of the value.

Every numeric input/output needs a parameter `iLen` which determines the total character length of the field. If the length is too short for the representation of the numeric value, the field will be filled with the character ‘x’.

### 8.1.4 Output in text dialog

- **Strings:**
  ```bc
  MMI_PrintStr(0, 0, "I am a text dialog object.", TRUE)
  ```
  Parameters: column, line, string, `lValid`

- **Integer values:**
  ```bc
  MMI_PrintInt(10, 1, 10, 10578, TRUE)
  ```
  Parameters: column, line, `iLen`, integer value, `lValid`

- **Double (floating point) values without unit:**
  ```bc
  MMI_PrintVal(10, 2, 10, 3, 90478.568, TRUE, MMI_DEFAULT_MODE)
  ```
  Parameters: column, line, `iLen`, decimals, double value, `lValid`, `Mode`

- **Double (floating point) values with unit:**
  ```bc
  DIM hz AS Angle
  hz = PI/4
  MMI_PrintVal(10, 3, 8, 3, hz, TRUE, MMI_DIM_ON)
  ```
  Parameters: column, line, `iLen`, decimals, double value, `lValid`, `Mode`

### 8.1.5 Input in text dialog

Input is roughly dual to the output, except that the input functions return the button id of the button that terminated the edit process. For all numeric values there are the minimum and maximum values defined. The value is only valid, if it is between them.
• Strings:
  `MMI_InputStr(17, 3, 10, sInput, lValid, iButtonId)`
  Parameters: column, line, string variable, lValid, button

• Integer values:
  `MMI_InputInt(24, 4, 100, 200, iValue, lValid, iButtonId)`
  Parameters: column, line, iLen, minimum value, maximum value, integer variable, lValid, button

• Double (floating point) values without unit:
  `MMI_InputVal(19, 4, 8, 2, 0, 399.99, MMI_DEFAULT_MODE, dValue, lValid, iButtonId)`
  Parameters: column, line, iLen, decimals, minimum value, maximum value, mode, double variable, lValid, button

• Double (floating point) values with unit:
  `MMI_InputVal(19, 4, 8, 2, 0, 399.99, MMI_DIM_ON, dValue, lValid, iButtonId)`
  Parameters: column, line, iLen, decimals, minimum value, maximum value, mode, double variable, lValid, button
• List: Lists take a variable of a predefined type as parameter.
  
  ```basic
  TYPE ListArray (25) AS String30 END
  ```
  
  This definition determines the maximum number of entries in a list to be 25, each one is a string of type String30. We create a list with 4 items and use the second entry as default (initial selection).

  ```basic
  DIM aList AS ListArray
  DIM iIndex AS Integer

  aList(1) = "List Item 1"
  aList(2) = "List Item 2"
  aList(3) = "List Item 3"
  aList(4) = "List Item 4"
  iIndex = 2
  MMI_InputList(8, 4, 12, 4, MMI_DEFAULT_MODE, aList, iIndex, lValid, iButtonId)
  ```

  Parameters: column, line, iLen, number of items, mode, list variable, index, lValid, button

8.2 THE GRAPHICS DIALOG

8.2.1 Positioning on the display

Every graphics function needs the position on the display. The graphics display is organized in x- (horizontal) and y-pixels (vertical). The left upper position has x-pixel and y-pixel number 0. The x-pixel number is rising to the right and the y-pixel number is rising down. The size of the display is 232 times 48 pixels.

8.2.2 Creating a graphics dialog

Calling MMI_CreateGraphDialog creates a new graphics dialog.

```basic
  MMI_CreateGraphDialog("BASIC", "Graphics Dialog", "My help text.")
```

A graphics dialog with short caption "BASIC" and caption "Graphics Dialog" is created. The help text is set to "My help text." — it is shown when the user presses Shift-F1 and the help functionality of the theodolite is enabled.
8.2.3 Graphics functions

After having created the graphics dialog, the graphics functions may be used. (E.g. `MMI_DrawLine`, `MMI_DrawCircle`, `MMI_DrawText`, etc. See the "Reference Manual" for a detailed description.)

8.2.4 Deleting a dialog

When a dialog is not used any more it must be deleted. The name of the dialog deletion procedure is for text, measurement and graphics dialogs the same:
```
MMI_DeleteDialog()
```

8.2.5 Mixing text and graphics dialogs

There can be only one text dialog at a time, i.e. an existing text dialog must be deleted with `MMI_DeleteDialog` before a new one can be created with `MMI_CreateTextDialog`. The same holds for a graphics dialog (with the appropriate creation procedures).

But a graphics dialog may be opened while a text dialog is active. (Note: The reverse is not the case: a text dialog may not be opened while a graphics dialog is open.) If a text dialog and a graphics dialog are open, the graphics dialog has priority, i.e. all future function calls are related to the graphics dialog (until it is closed). For example, `MMI_AddButton` (see below) will add the button to the graphics dialog, and all the display functions must be for graphic dialogs (such as `MMI_DrawCircle`, etc.).

8.2.6 Adding buttons

The user may add buttons to a dialog. (These buttons will be added to the defined buttons of the dialog.) When adding a button it must be specified what text should be displayed for that button. Such a text can be up to five characters long and is displayed centred above the button.

Each button has an identification associated. This button id is needed.

---

1 An existing text dialog is deleted automatically if a new text dialog is created.
• for specifying which button is to add in MMI_AddButton, and
• checking what button was pressed or that is returned from a system function.

Example:

We add the F1-button to the currently opened dialog, giving the meaning "CONT" to it.

    MMI>AddButton( MMI_F1_KEY, "CONT" )

\textbf{Note} The button id's are defined as constants in the compiler.

\subsection*{8.2.7 Responding to buttons}

There are two procedures for coping with button presses:

• MMI_CheckButton queries whether there was a button pressed or not, and
• MMI_GetButton retrieves a pressed button. If there was no button pressed it waits until one is pressed. The second parameter to MMI_GetButton (the in-parameter bAllKey) determines what buttons are accepted:
  - If it is \texttt{TRUE}, any button is accepted.
  - If it is \texttt{FALSE}, only ESC, or a defined button (added with MMI_AddButton) are accepted.
Example:

The example does some work in a loop until Shift-F6 is pressed. As long as there is no button pressed, the display is constantly updated (e.g. the current angles from the theodolite are displayed). If there is a button pressed, this button is handled.

'bDone must be initialized
bDone = FALSE
DO WHILE NOT bDone 'as long as the job is not done
  'check for defined buttons and get its id
  MMI_GetButton( buttonId, FALSE )
  SELECT CASE buttonId 'handle it
  CASE MMI_F4_KEY
    'handle MMI_F4_KEY
    CASE MMI_SHF6_KEY
      bDone = TRUE 'that’s it, 'terminate loop
      CASE ...
    'here go the other handled keys
  ELSE
    'here go the unhandled keys
    END SELECT
    'update the display
  LOOP

8.2.8 Standard key binding

It is clear that for the user it is important that the same name\(^2\) — and moreover the same key — always has the same meaning associated (at least conceptually). An exception is the F1-key, its meaning is not the same in a measurement dialog and in a configuration dialog. In the following table there are the standard key bindings with the caption, the text which is displayed above the keys:

<table>
<thead>
<tr>
<th>Key in measurement dialog</th>
<th>Caption</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 in measurement dialog</td>
<td>ALL</td>
<td>Does first DIST, then REC. (See below)</td>
</tr>
<tr>
<td>F1 in configuration dialog</td>
<td>CONT</td>
<td>Continues to the logically following dialog.</td>
</tr>
</tbody>
</table>

\(^2\) For instance, the user of a LEICA theodolite assumes that DIST takes the distance (with the common dialogs), ALL does DIST and then REC, etc.
8.3 NAMING CONVENTIONS

We propose some naming conventions for GeoBASIC. More extensive conventions can be found in the naming conventions for Microsoft Access (which are tied closely to Visual Basic conventions).³

8.3.1 Variable names

Variable names of simple types (i.e. all the scalar types and strings) may be tagged to indicate their type. Prefixes are always lowercase so your eye goes past them to the first uppercase letter — where the base name begins. If the base name consists of more than one word, upper case letters within the name are used to distinguish its parts.

<table>
<thead>
<tr>
<th>Key</th>
<th>Caption</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F2</td>
<td>DIST</td>
<td>Start distance measurement.</td>
</tr>
<tr>
<td>F3</td>
<td>REC</td>
<td>Records the previously measured / computed data.</td>
</tr>
<tr>
<td>SHIFT-F1</td>
<td>HELP</td>
<td>Displays a help text if the theodolite help functionality is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This key is provided and handled completely by the system, it is not accessible from GeoBASIC.</td>
</tr>
<tr>
<td>SHIFT-F6</td>
<td>QUIT</td>
<td>Terminates an application.</td>
</tr>
<tr>
<td>ESC</td>
<td></td>
<td>Cancels an input or goes a step back. GeoBASIC applications should handle it.</td>
</tr>
<tr>
<td>CODE</td>
<td></td>
<td>Shows the coding dialog.</td>
</tr>
</tbody>
</table>


Note These naming conventions carry only a semantics for the programmer, not for the compiler.
The base name succinctly describes the object. For example, PointNumber or justPointNo for the number of a point. Object tags are short abbreviations and simplifications describing the type of the object. For example, the tag 'i' in iPointNo denotes that the type of the variable is Integer. The following table lists the tags for the GeoBASIC types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>i</td>
</tr>
<tr>
<td>Logical</td>
<td>l</td>
</tr>
<tr>
<td>Double</td>
<td>d</td>
</tr>
<tr>
<td>Distance</td>
<td>d</td>
</tr>
<tr>
<td>Subdistance</td>
<td>d</td>
</tr>
<tr>
<td>Angle</td>
<td>d</td>
</tr>
<tr>
<td>Pressure</td>
<td>d</td>
</tr>
<tr>
<td>Temperature</td>
<td>d</td>
</tr>
<tr>
<td>String</td>
<td>s</td>
</tr>
</tbody>
</table>

Note that all types which represent floating point numbers are tagged by 'd'. This is because operations valid for the type Double are also valid for the other d–tagged types.

If there are several similar object names, a qualifier may follow the name and further clarify it. For example if we kept two special point numbers, one for the first point and one for the last, the variable names would be the (qualified) variables iPointNoFirst and iPointNoLast.

Structure types do not have a default prefix, if needed the (abbreviated) type name could be used. For arrays the base name itself could contain the information that the variable names an array.

For global variables an additional prefix 'g' might be useful.

8.3.2 Constants and user-defined types

Constants begin with an upper case character. If constants contain only upper case characters (as most of the predefined constants do) the underscore '_' is used to separate parts of the name. Often constants can be grouped together, then a prefix is used to denote their common criterion. For example the return codes use RC, as in RC_OK, RC_ABORT, etc.
Mostly constants are globally defined. For local constants an additional prefix 'loc' might be useful.

User defined types begin with an upper case character. Use the postfix '_TYPE', '_Type' or 'Type' (according to the naming convention used for the type name itself) appended to the type name to denote that it is a type structure. Alternatively, you can use a prefix 'T'. (For types these conventions are useful since GeoBASIC is not case sensitive. Hence, for example, if there is a type Date no variable can be named date. If the type has the name TDate or Date_TYPE or DateType, there can.) As for local constants, local types might be prefixed with 'loc'.

8.3.3 Procedures

A procedure name begins with an upper case letter and succinctly describes the action that is performed. Variables that denote parameters passed to a function or subroutine (in the parentheses after the function/subroutine name) should be well documented, also indicating whether they act as input, output, or input and output parameters.

8.3.4 Keywords

GeoBASIC keywords are all in upper case letters. For example, DIM, FOR, LOOP, FUNCTION, etc.

8.3.5 Labels

For error labels (ON ERROR GOTO) we use the function/subprocedure name with the qualifier '_Err' appended.
SUB LabelExample ()
  'code of the procedure

LabelExample_Err:
  SELECT CASE ERR
    'handle specific errors here
  CASE ELSE
    'generic error handler here
  END SELECT
END LabelExample

8.3.6  Remark on naming conventions

Naming conventions never replace the judicious use of comments in your GeoBASIC program code. Naming conventions are an extension of, not a replacement for, good program-commenting techniques.

Formulating, learning, and applying a consistent naming style require a significant initial investment of time and energy. However, you will be amply rewarded when you return to your application a year later to do maintenance or when you share your code with others. Once you implement standardised names, you will quickly grow to appreciate the initial effort you made.

To complete the discussion about naming conventions, we mention the use of program headers:

In every function/subprocedure there should be a header describing, at a minimum, purpose, and parameters passed and/or returned. (In addition there might be comments, the author's name, last revision date, notes, etc.)
9 REFINED GEOBASIC CONCEPTS

In GeoBASIC several concepts are implemented to utilise and standardise programming and applications.

9.1 UNITS

Working with units always gives rise to the problem that different users want to work with different units. In geodesy, take the vertical angle as an example: some surveyors measure in Gon, some in radians, others in percentages. And, in addition to the unit-problem, there is the question where to fix the zero point of some scale. Again for the vertical angle example: some surveyors want to have zenith angles, some nadirs, some something in between.

To cope with this situation there is a fine automatic unit handling system built in the theodolite system, and the GeoBASIC programmer can take full advantage of it. All that has to be done in a GeoBASIC program, is to keep all values in SI units and, when a value has to be displayed specify what kind of value it is: a horizontal angle, a vertical angle, a distance, a temperature, etc. All the formatting, together with choice of the right representation (the user may define this in his theodolite system configuration with which the GeoBASIC programmer is not concerned), and displaying the unit after the value are handled automatically. (Of course the programmer can also decide not to use this automation and handle everything on his own. But values obtained from the system will be in SI units anyway.)

9.1.1 What the GeoBASIC programmer has to do

- Use SI units throughout the program. All computations are done with values in SI units.
- When displaying, specify the correct data type i.e. Distance for the value is displayed. See description of the MMI_PrintVal function in the "Reference Manual".

We will give an example of measuring an horizontal angle, computing the difference to a given angle, and displaying the difference on the display. (Note that we use the GetAngleHz routine from the MeanHz program (see 10.1), and we assume that a text dialog has been opened properly. The angle difference is normalised to the range 0 to 2×π.)
Example

```basic
DIM dHz1 AS Angle 'first horizontal angle
DIM dHz2 AS Angle 'second horizontal angle
DIM lValidHz2 AS Logical 'indicator if second 'angle is valid
DIM dDiffHz AS Angle 'the difference of the 'angles

'assume dHz1 is initialized here to an angle 'in radians
GetAngleHz( dHz2, lValidHz2 )
dDiffHz = dHz1 - dHz2
GM_AdjustAngleFromZeroToTwoPi( dDiffHz )
MMI_PrintVal( 20, 0, 8, 3, dDiffHz, lValidHz2, MMI_DIM_ON )
```

The output is as follows:
- If the `GetAngleHz` routine returned a valid angle, also the difference `dDiffHz` will be valid (this is why `lValidHz2` is used in the `MMI_PrintVal` function). In this case the angle will be formatted in an 8 character wide field with 3 decimals, afterwards the unit according the theodolite system configuration will be displayed. Assume that `gon` is set and the angle difference was 1.5473452 radians, then at position 20 in line 0 the output will be «98,507 g».
- If the angle returned from `GetAngleHz` was not valid, five dashes will be displayed «----- g».

### 9.1.2 What the user/surveyor has to do

The user has to set up the units, in which he want to work, in the theodolite system configuration. All outputs that use the theodolite system will automatically be formatted according to this setting.

### 9.2 THE USER MEASUREMENT DIALOG

The User Measurement Dialog (sometimes referred as MDlg) standardises the visualisation of the measurement values in GeoBASIC. Each value (i.e. vertical angle, horizontal distance) has a predefined output format. Thus the GeoBASIC
programmer has only to define, on which line a value should be displayed. All lines begin with a brief description of the value.

*For example (Output of the horizontal distance):*

```
  «Horiz.Dist: 158.287 m»
```

Additionally the measurement parameters and (self-definable) application parameters can be displayed in the measurement dialog. Thus a user is able to change measurement parameters immediately and without leaving the dialog. All measurement values and measurement parameters are saved in the theodolite’s data pool as system parameters.

We distinguish between measurement and application parameters. The former are defined by the system in its meaning and data type. The latter can be defined freely by the user. Please refer to Appendix H in the reference manual for a list of all system and application parameters, which can be used in a measurement dialog.

### 9.2.1 Configuration of the User Measurement Dialog

Before using the measurement dialog we have to define its contents. There are 3 types of possible entries:

- **System parameters:**
  The routine `GSI_SetLineMDlg` places a system parameter (measurement value or measurement settings) on a line.

- **Pure text line:**
  The routine `GSI_SetLineMDlgText` places any text on a line.

- **Application parameters:**
  The routine `GSI_SetLineMDlgPar` places a (self-definable) application parameter on a line.

**Note**

The user measurement dialog configuration is automatically initialised with the entries of the first system measurement dialog. Thus all lines which are not configured by the GeoBASIC programmer shows the same parameters as the first system measurement dialog. For further explanations how to configure the user measurement dialog read the description of the 3 system functions (`GSI_SetLineMDlg`, `GSI_SetLineMDlgText`, `GSI_SetLineMDlgPar`) in the reference manual.
9.2.2 Creating the User Measurement Dialog

After the definition of the content GSI_CreateMDlg analogous to the creation of a text dialog creates the user measurement dialog. For adding buttons to the dialog use MMI_AddButton.

9.2.3 Executing the User Measurement Dialog

In the following example a measurement dialog is created with the horizontal angle on line 2 and the buttons “DIST” on F2-key and “QUIT” on SHIFT-F6-key. All other lines are predefined by the system. After the creation of the dialog the measured values will be updated in a loop:

```plaintext
'Change line 2
GSI_SetLineMDlg(2, GSI_PAR_AngleHz)
GSI_CreateMDlg(2, "MEAS", "Measurement Test",
               "Measurement Help...")

'Addition of buttons
MMI_AddButton(MMI_F2_KEY, "DIST")
MMI_AddButton(MMI_SHF6_KEY, "QUIT")
lDone = FALSE
DO WHILE NOT lDone
   GSI_UpdateMeasurement(TMC_AUTO_INC, WAITTIME,
                         lRecValid, iCode, FALSE)
   GSI_UpdateMDlg(iButton)
   SELECT CASE iButton
      CASE MMI_F2_KEY
         'DIST Button --> meas a distance and angles
         BAP_MeasDistAngle(iDistMode, dHz, dV, dDist, TRUE,
                            MEAS)
      CASE MMI_ESC_KEY, MMI_SHF6_KEY
         'done --> exit this routine
         lDone = TRUE
   END SELECT
   LOOP 'end measurement loop
   'delete measurement dialog
   MMI_DeleteDialog()
```

The routine GSI_UpdateMeasurement updates the measurement values in the theodolite data pool. GSI_UpdateMDlg updates the user measurement dialog with the new values and returns the pressed button. For further explanations read the description of these system routines in the reference manual.
If the user measurement dialog is not used any more it must be deleted with \texttt{MMI\_DeleteDialog}.

See the example program \texttt{MEAS.GBS} for a typical usage of the user measurement dialog.

\subsection{Mixing the User Measurement Dialog with Other Dialogs}

There can be only one user measurement dialog at a time, i.e. an existing user measurement dialog must be deleted with \texttt{MMI\_DeleteDialog} before a new one can be created with \texttt{GSI\_CreateMDlg}. If a user measurement dialog is active, no text dialog can be opened and vice versa. But a graphics dialog may be opened while a user measurement dialog is active.

\begin{center}
\begin{tabular}{|l|l|}
\hline
\textbf{Note} & The reverse is not the case: a user measurement dialog may not be opened while a graphics dialog is open. If a user measurement dialog and a graphics dialog are open, the graphics dialog has priority, i.e. all future function calls are related to the graphics dialog until it is closed. \\
\hline
\end{tabular}
\end{center}

\section{TPS1100 CONFIGURABILITY}

In general, each part of an application, which should be accessible from outside, has to be of the form ‘GLOBAL SUB’. These points are known as entry points and can be used in two ways. First they can be linked to a menu item (of the a system), and second they can be described as configuration item.

\subsection{Adding the program in a System Menu}

The easier way to access an entry point of an application is to link it to a menu item during the installation phase. Please refer to the Reference Manual \texttt{MMI\_CreateMenuItem} for further explanations.
9.3.2 Import the program in a User Configuration

The TPS1100 series theodolites support the concept of individual configurations. In a configuration the user can define his own dialogs or menus and link them to certain events (i.e. pressing the PROG key or Power ON). If the event occurs then the linked dialog or the menu will be displayed. The user can create and change his configuration on the PC with the Customisation Tool.

The import of a GeoBASIC program in a user configuration means, that an external GeoBASIC routine is linked with an item of a user defined menu, a button of a user defined dialog or directly with an event. If either the event occurs or the button is pressed or the menu item is selected, then the linked external routine is executed. For the import of a GeoBASIC program the Customisation Tool needs a special file named APPInfo-file with the necessary information about the program.

The usage of the APPInfo-file in the Customisation Tool:

- Start the Customisation Tool
- Open a configuration file, appropriate text- and definition files
- Choose Import Application from the file menu
- Check the box named with the program name (i.e. AppInfoExample)
- Press the OK button

Now the globally accessible subroutines may be added to menus, buttons, etc. simply by using drag and drop.

**Generate the AppInfo-file**

The AppInfo-file is automatically generated during compilation, if there is a application information (short AppInfo) section in the GeoBASIC source file.

| Note | The AppInfo-section has to occur at the end of the source code. The AppInfo-section is optional; if there is no AppInfo-section in the GeoBASIC source file, the AppInfo-file generation is omitted. The global routine "Install" is optional, since any global routine may be associated with a menu entry, using the AppInfo-file via the Customisation Tool. |

The following GeoBASIC sample code illustrates the usage of the AppInfo-section in a GeoBASIC source file. See also the sample program AppInfoTest.gbs.
PROGRAM AppInfoExample

'-------------------------------------------------------
GLOBAL SUB GlobalSub1
Dim dummy As Integer
MMI_WriteMsgStr("AppInfoExample.", "GlobalSub1 in AppInfoExample called", MMI_MB_OK, dummy)
END GlobalSub1

'-------------------------------------------------------
GLOBAL SUB GlobalSub2
Dim dummy As Integer
MMI_WriteMsgStr("AppInfoExample.", "GlobalSub2 in AppInfoExample called", MMI_MB_OK, dummy)
END GlobalSub2

END AppInfoExample

'Application Information for Config Tool
'---------------------------------------APPINFO

APPINFO

GENERAL
SET Author "Leica AG, CH - Heerbrugg"
SET Desc "AppInfo Example Application"
SET TheoModel "TCA1100"
END GENERAL

ENTRYPOINT GlobalSub1
SET CapLg "Global Sub 1"
SET CapSh "GSUB1"
SET Desc "test of appinfo subroutine 1"
END GlobalSub1

ENTRYPOINT GlobalSub2
SET CapLg "Global Sub 2"
SET CapSh "GSUB2"
SET Help "displays a message and exits"
END GlobalSub2

END APPINFO

The global subroutines GlobalSub1 and GlobalSub2 are indicated as entry points for the import in a user configuration. Refer to Chapter 2.11 in the Reference Manual for a description of the syntax in BNF-form.
9.4 INTERAPPLICATION-CALL

The inter-application-call makes it possible to call a subroutine in another GeoBASIC program. With this concept the GeoBASIC programmer can use the same subroutine in several programs.
9.4.1 Definition of a subroutine for Interapplication-Call

If a subroutine should be called by another application, it must be defined as a global subroutine.

Example:

```plaintext
PROGRAM IAC2
  GLOBAL SUB InterAppEntry
  DIM iButton AS INTEGER
  MMI_WriteMsgStr("Welcome in IAC2","IAC2", MMI_MB_OK, iButton)
END InterAppEntry
END IAC2
```

9.4.2 Call the global subroutine

Before calling the global subroutine, the GeoBASIC programmer has to check with `CSV_LibCallAvailable` if the subroutine is available. That usually means if it is loaded or not. If the subroutine available, he can invoke it with `CSV_LibCall`.

Example:

```plaintext
DIM lAvailable AS LOGICAL
'Check if global subroutine is available
CSV_LibCallAvailable("IAC2","InterAppEntry", lAvailable)
IF lAvailable
  'available, call global subroutine
  CSV_LibCall("IAC2", "InterAppEntry", "BASIC")
END IF
```

See the example program IAC.GBS and IAC2.GBS for a typical usage of inter-application-call. For further explanations read the description of `CSV_LibCall` and `CSV_LibCallAvailable` in the reference manual.

9.5 SYSTEM FUNCTION CALL

If a theodolite user creates his own configuration on the PC with the Customisation Tool, he has a wide selection of predefined system functions which he can add to menus, buttons, etc. After the loading of the configuration he calls the system functions by selecting the appropriate menu item or button.

The GeoBASIC programmer has the same possibilities. With the routine `CSV_SysCall` he can call the system functions in his programs. Because some system functions do not run on every theodolite type, there is a routine
CSV_SysCallAvailable, which returns if the system function can be executed.

**Example:**

```basic
DIM lAvailable AS Logical
CSV_SysCallAvailable(CSV_SFNC_PositCompassDlg, lAvailable)
IF lAvailable
    CSV_SysCall(CSV_SFNC_PositCompassDlg)
END IF
```

If the system function CSV_SFNC_PositCompassDlg can be executed (RCS mode is active), then the dialog RCS orientation with a compass is displayed. For further explanations read the function descriptions of CSV_SysCall and CSV_SysCallAvailable in the reference manual. In Appendix H of the reference manual there is a list of all system functions.

### 9.6 SYSTEM EVENT GENERATION

Every configuration for a TPS1100 series theodolite is event driven. The user or the system itself generates an event (e.g. the user has pressed the PROG key or the initialisation sequence is finished) and the configuration functionality executes then the linked action (menu, dialog, macro, application or system function).

A GeoBASIC program can generate all events, which can occur in the theodolite system software, also. To generate a system event the same functions can be used as for calling system functions. The routine CSV_SysCall is used for the generation of system events. The routine CSV_SysCallAvailable returns TRUE, if there is an action linked to the requested event and the action can be executed.

**Example:**

```basic
DIM lItemDefined AS Logical
CSV_SysCallAvailable(CSV_EFNC_CompensatorSetting, lItemDefined)
IF lItemDefined
    CSV_SysCall(CSV_EFNC_CompensatorSetting)
END IF
```

If a configuration item is defined for the system event CSV_EFNC_CompensatorSetting (compensator setting event; usually connected to a compensator setting dialog) CSV_EFNC_CompensatorSetting is generated and the appropriate system function, application, macro, dialog or menu is
executed. For further explanations read the function description of CSV_SysCall and CSV_SysCallAvailable in the reference manual. In Appendix H of the reference manual there is a list with all system events.
10 GEOBASIC SAMPLE PROGRAMS

10.1 MEANHZ — MEAN VALUE OF HORIZONTAL ANGLE MEASUREMENTS

10.1.1 Program description

The program "MeanHz" measures a number of horizontal angles and computes its arithmetic mean value. The measured angles and the mean angle can then be displayed graphically.

Program flow:
First, the user may enter the number of horizontal angles he wants to measure. (The number of angles must be within a certain range.) Then the angles are measured — each time the REC key is pressed the current horizontal angle is recorded.
As soon as the requested number of angles is measured, the mean angle is computed and displayed. Now the user has the choice either to display the angles graphically, to move the theodolite to the computed mean angle or to quit the program. (The program can be terminated with the ESC button or the QUIT button on shift-F6 at any time.)
Input the number of angles to measure

Measure the angles

Compute and display mean angle

Show Graphics?

Move theodolite to mean angle?

End

Draw the angles

Position theodolite on mean angle
10.1.2 Source code listing

See example file "meanhz.gbs"

PROGRAM Mean
' Sample application for building the mean value of angles
' Measures a user defined number of horizontal angles and calculate
' the mean angle. The measured and the mean angle can also be
' displayed graphically.
' GeoBASIC 1.0 for TPS1100 Series Instruments
' (c) Leica AG, CH - Heerbrugg 1998
' Global Declarations
CONST MaxNoHz = 9 'Maximum number of angles that can be
' measured
CONST CaptionShort = "MEAN" 'Short caption (displayed lefthand, in
' top line)
'Type to store the angles (for graphics)
TYPE DIM
TAngles (MaxNoHz) AS Angle
END
DIM fId AS FileId 'File identification

-----
GLOBAL SUB Install
' Description
' Adds the program into the theodolite's PROG menu. The program's
' (application's) name is 'Mean', the global routine to start is
' 'Main' and the program menu item will be named 'MEAN HZ'.

  MMI_CreateMenuItem("Mean", "Main", MMI_MENU_PROGMENU, "MEAN HZ")
END Install

SUB RecordValue (dHz As Angle, ByVal dMean As Angle)
' Description
' Writes the value to data link and file.
' DIM sVal1 As String30

DIM sVal2 As String30
DIM sOut As String255

ON Error Resume Next 'Ignore all errors

MMI_FormatVal(MMI_FFORMAT_HZANGLE, 10, 2, dHz, TRUE, 
MMI_DEFAULT_MODE, sVal1)
MMI_FormatVal(MMI_FFORMAT_HZANGLE, 10, 2, dMean, TRUE, 
MMI_DEFAULT_MODE, sVal2)

sOut = "hz: " + sVal1 + " mean: " + sVal2 'Compute output text

'Write to data link and file
Send(sOut)
Print(fId, sOut)

END RecordValue

'--------------------------------------------------------------------
SUB GetAngleHz ( dHz AS Angle, lValid AS Logical)

' ----------
' Description
' Measures the horizontal angle 'valid' indicates if the dHz is
' valid.
',
' Parameters
' OUT: dHz, OUT, lValid
',
DIM theoAngle AS TMC_Angle_Type 'The measured values
DIM iInfo AS Integer 'Return code

ON Error Resume Next 'Ignore all errors

'TMC_GetAngle( theoAngle, iInfo )

IF (Err = RC_OK) THEN
lValid = TRUE
 dHz = theoAngle.dHz
ELSE
lValid = FALSE
END IF

END GetAngleHz

'--------------------------------------------------------------------
SUB ShowGraphics( byVal iNoPoints AS Integer, angles AS TAngles, 
byVal dMean AS Angle )

' ------------
' Description
' Displays the measured and the mean horizontal angles
' graphically.
',
' Parameters
' IN: iNoPoints, angles, dMean
',
DIM iX AS Integer 'x coordinate
DIM iY AS Integer 'y coordinate
DIM iButton AS Integer 'button id

CONST CX = 90 'display center x coordinate
CONST CY = 24 'display center y coordinate
CONST DL = 20 'length of line
CONST HELPTEXT = "Visualizes the angles with lines from the station. " +
"The computed mean angle is shown by the longer line. " +
"The north angle is 0."

MMI_CreateGraphDialog( CaptionShort, "PICTURE", HELPTEXT )

'Draw center and circle
MMI_DrawCircle( CX, CY, 3, 3, MMI_NO_BRUSH, MMI_PEN_BLACK )
MMI_DrawCircle( CX, CY, DL, DL, MMI_NO_BRUSH, MMI_PEN_BLACK )

'Draw lines for angles (there are iNoPoints angles)
DO WHILE iNoPoints > 0
  'compute the line
  iX = INT( DL * SIN(angles(INT(iNoPoints))) )
iY = INT( DL * COS(angles(INT(iNoPoints))) )
  MMI_DrawLine( CX, CY, CX+iX, CY-iY, MMI_PEN_BLACK )
iNoPoints = iNoPoints - 1
LOOP

'Draw line for dMean
iX = INT( (DL+4) * SIN(dMean) )
iY = INT( (DL+4) * COS(dMean) )
MMI_DrawLine( CX, CY, CX+iX, CY-iY, MMI_PEN_DASHED )

'Wait for key press and finish dialog
MMI_AddButton( MMI_F5_KEY, "END" )
MMI_GetButton( iButton, FALSE )
MMI_DeleteDialog()

END ShowGraphics

'--------------------------------------------------------------------
GLOBAL SUB Main
'
' Description
' Reads the number of points to be measured. Measures these points,
' calculates the mean value and shows the result or moves (if
' motorized) the TPS tocalculated position.
'
DIM iNoPoints AS Integer 'number of points to measure
DIM iCurrNo AS Integer 'current point number
DIM lNoOk AS Logical 'TRUE if no of points are valid
DIM lHzOk AS Logical 'TRUE if measured hz is valid
DIM dHz AS Angle 'measured hz
DIM storeHz AS TAngles 'array of measured angles
DIM dMean AS Angle 'calculated mean angle
DIM lKeyPressed AS Logical 'TRUE if button pressed
DIM iButton AS Integer 'id of pressed button
DIM Family AS TPS_Fam_Type 'this data structure is used to store information about the system

ON Error Resume Next 'ignore errors

'check which type of instrument is active and open file
CSV_GetInstrumentFamily( Family )
IF ( Family.lSimulator ) THEN
  Open( "C:\results.txt", "Append", fId, 0 )
ELSE
  Open( "A:\results.txt", "Append", fId, 0 )
END IF

' set up dialog and input iNoPoints
MMI_CreateTextDialog ( 6, "MEAN", "HZ MEAN VALUE",
"Compute mean HZ for a number of measurements." )

' *******************************
' * read in iNoPoints *
' *******************************
iNoPoints = 3
lNoOk = TRUE
MMI_PrintStr( 0, 0, "No of points:", TRUE )
MMI_AddButton( MMI_F1_KEY, "CONT" )
MMI_AddButton( MMI_SHF6_KEY, "QUIT" )
MMI_InputInt( 26, 0, 2, 1, MaxNoHz, MMI_DEFAULT_MODE, iNoPoints, lNoOk, iButton )

' setup rest of dialog
iCurrNo = 1
MMI_PrintStr( 0, 1, "Curr. point:", TRUE )
MMI_PrintVal( 26, 1, 2, 0, iCurrNo, TRUE, MMI_DEFAULT_MODE )
MMI_PrintStr( 0, 2, "HZ:", TRUE )
MMI_AddButton( MMI_F3_KEY, "REC" )

'init mean value
dMean = 0.0

'get iNoPoints points (abort if ESC or QUIT is pressed)
DO WHILE (iCurrNo <= iNoPoints) AND (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY)
  MMI_PrintVal( 26, 1, 2, 0, iCurrNo, lNoOk, MMI_DEFAULT_MODE )
  MMI_CheckButton( lKeyPressed )
  iCurrNo = iCurrNo + 1
  dMean = dMean + dHz
  lKeyPressed = FALSE
END DO
IF lKeyPressed THEN
    MMI_GetButton( iButton, FALSE )
    SELECT CASE iButton
        CASE MMI_F3_KEY, MMI_F1_KEY
            GetAngleHz( dHz, 1HzOk )
            storeHz(iCurrNo) = dHz
dMean = dMean + dHz
        'if REC pressed record values
        IF iButton = MMI_F3_KEY THEN
            RecordValue(dHz, dMean/iCurrNo)
        END IF
        iCurrNo = iCurrNo + 1
    END SELECT
ELSE
    'update display
    GetAngleHz( dHz, 1HzOk )
    MMI_PrintVal( 20, 2, 8, 3, dHz, 1HzOk, MMI_DEFAULT_MODE )
END IF
END LOOP

**************************
* show results *
**************************

'if execution should proceed
IF (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY) THEN
    'setup new buttons
    MMI_DeleteButton( MMI_F1_KEY )
    MMI_DeleteButton( MMI_F3_KEY )
    MMI_AddButton( MMI_F3_KEY, "SHOW" )
    MMI_AddButton( MMI_F4_KEY, "EXIT" )
    MMI_AddButton( MMI_F5_KEY, "GOTOM" )

    'compute mean value
dMean = dMean / iNoPoints
    MMI_PrintStr( 0, 3, "Mean HZ :", TRUE )
    MMI_PrintVal( 20, 3, 8, 3, dMean, TRUE, MMI_DEFAULT_MODE )
DO WHILE (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY) AND (iButton <> MMI_F4_KEY)
    MMI_GetButton( iButton, FALSE )
SELECT CASE iButton
    CASE MMI_F3_KEY
        ShowGraphics( iNoPoints, storeHz, dMean )

        'move theo to the computed mean horizontal angle
        CASE MMI_F5_KEY
            BAP_PosTelescope(BAP_POSIT_HZ, BAP_POS_MSG, dMean, 0, 0.1, 0.1)
    END SELECT
END SELECT
LOOP
END IF

'clean up text dialog
MMI_DeleteDialog()

'close output file
Close(fid)

END Main
END Mean

10.2 SAMPLE PROGRAMS

These code samples gives you some help for building your first applications. Each of them should give you some hints in a specific problem domain.

- **appinfotest.gbs** This example shows the use of the application information section in the GeoBASIC source file.
- **codefunc.gbs** An example of a program which will be called, when the Code-key has been pressed.
- **cursor.gbs** Cursor control in a dialog.
- **error_ha.gbs** This program shows how error handling changes execution of a program.
- **language.gbs** Take this program as an example to support multiple language applications. Two language files and its text databases are provided to see how multilingual support works.
- **meanhz.gbs** This sample shows the calculation of the mean value of horizontal angle measurements, see Chapter 10.1.
<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>meas.gbs</code></td>
<td>A simple example how to measure with BAP-functions, including Quick-Coding</td>
</tr>
<tr>
<td><code>meas_od.gbs</code></td>
<td>A simple example how to measure and how to record data in an own data-format, including Quick-Coding</td>
</tr>
<tr>
<td><code>stringer.gbs</code></td>
<td>This example shows in which situations typical errors may occur.</td>
</tr>
<tr>
<td><code>test.gbs</code></td>
<td>An empty frame for building up a GeoBASIC application.</td>
</tr>
<tr>
<td><code>tracking.gbs</code></td>
<td>This program shows possible techniques to take advantage of the measurement facilities.</td>
</tr>
<tr>
<td><code>menu.gbs</code></td>
<td>A simple menu handler.</td>
</tr>
<tr>
<td><code>dirlst.gbs</code></td>
<td>This example shows how to get PC card information and how to read a directories content.</td>
</tr>
<tr>
<td><code>inclmain.gbs</code></td>
<td>This example shows the usage of an include file.</td>
</tr>
<tr>
<td><code>iac.gbs</code></td>
<td>An example for an interapplication call.</td>
</tr>
</tbody>
</table>
11 PORTING A TPS1000 ORIGINATED PROGRAM

The implementation of the TPS1100 theodolite series includes several new concepts compared to the firmware of TPS1000 theodolites. To follow up these new concepts and to take care of functionality that has been changed or removed in the implementation of TPS1100 firmware, GeoBASIC programs, once developed for TPS1000 hardware, cannot be compiled without changing the source code.

In this chapter we will cover this subject and we try to give some guidelines to help the developer to port the source code onto the new platform. During the design phase of GeoBASIC for TPS1100 systems we took certain care to make the migration as smooth as possible. Although all programs’ source code has to be changed, the effort to port it will be for the most applications not that high.

In the very end this means also that the developer has to maintain two source code bases.

11.1 TPS1100 HARDWARE RELATED CHANGES

11.1.1 Display Line Length

The TPS1100 series instruments use a different liquid crystal display. The difference means also that one can use only 29 characters per line. To be ‘independent’ of the display length we defined the string type DisplayLine. It does not contain the string length in the name, hence this should help in future to port applications. To be compatible with older, TPS1000 GeoBASIC programs we did not change all String30 declarations. Of course only 29 characters will be printed out to the display.

11.1.2 Keyboard

The number of keys has been reduced, there is no CONT-Key any longer. Remove all MMI_CONT_KEY appearances in the source code. We deleted the definition of this constant to make it more obvious to the programmer that he has to change the source code and think about any button assignments.
11.2 CHANGES TO THE SIMULATOR
Now TPSSim supports GeoBASIC programs larger than 64 KB. A restriction, which turned out in the past, bothered the most of the GeoBASIC program developers. We would like to point out that the SWTheo extension enables the programmer to influence the execution of a program. With specific dialogs the programmer gets the possibility to set or change certain (measurement) values. We hope this helps a lot to simulate a more realistic TPS environment and makes it almost obsolete to have an instrument at your hand to test your application. Of course, still the final test of an application has to be done on an instrument. See also the documentation of TPSSim for further explanations.

11.3 NEW CONSTRUCTS IN GB_1100
Due to some requests we added a few new constructs to GeoBASIC for TPS1100 instruments.

11.3.1 #include Statement
It is now possible to include a GeoBASIC source file in another one. Nevertheless only one level of inclusion is allowed.

11.3.2 MID$ statement
Mid$’s implementation has been extended. Now Mid$ can be used to assign a character or a substring to another string at a certain position. In this way single characters of a string can be set or replaced.
Examples:
\[ T = \text{“abcdef”} \]
\[ \text{Mid$}(t, 2, 1) = \text{“+”} \quad \text{results in “a+cdf”} \]
\[ \text{Mid$}(t, 4) = \text{“---------”} \quad \text{results in “a+c---------”} \]

11.3.3 Application Info
A general concept of configurability has been introduced for the TPS1100 family of instruments. This gives totally new customisation possibilities into the hand of
the developer and more to the customer support. Up to a certain degree GeoBASIC supports this configurability. For example an assignment of a GeoBASIC program to a menu item can be changed by the new configuration utilities. Or it can be assigned to a function key.

To support these new features we extended the concept of the program by a section that describes the attributes of it.

This (informational) section can be appended optionally at the end of the source file. See the extra explanation of it to get further information about it.

11.4 GEOBASIC SOURCE CHANGES

Many GB programs have a similar structure. Therefore it does not surprise that many programs have to be rewritten in the same way to be compilable and executable for TPS1100 GeoBASIC.

11.4.1 General Dialog Changes

The CONT key does not exist any more on the TPS1100 instruments. Scan your source code for MMI_CONT_KEY and replace it by a function key. The TPS1100 guidelines use MMI_F1_KEY normally for the CONT key functionality. This might make it necessary to change your function key layout. Look at the existing dialogs to get an idea and to be more consistent to the built-in dialogs, to which function keys which functionality has been assigned.

In certain circumstances, where no function keys were left, the ESC key was the only way to leave a dialog. Normally ESC leaves a dialog with leaving values untouched.

MMI_SHIFT_ESC_KEY will not be supported any more. Instead one has to assign QUIT to (normally) Shift-F6. Quit leaves the whole application.

Note: 'Old' versions of constants and functions are left aligned. Newer versions or replacements have been shifted to right. The listed changes are ordered in an assumed importance.
Please notice that GB-TPS1000 supports conceptually 2(3) dialogs at once; a text or a graphics dialog and in parallel a customisable measurement dialog - MDlg.

A typical application may create a text dialog and link a graphics dialog to a menu button. Notice, that both dialogs exist at the same time and distinguish this situation from another, where the text dialog will be deleted before the graphical dialog will be created. In the former case one can go back to the text dialog without recreating it. In the latter the text dialog has to be rebuilt. In GB_TPS1100 text and measurement dialog are mutually exclusive.

See the following scheme for a graphical explanation. "()" denotes a dialog.

<table>
<thead>
<tr>
<th>TPS1000</th>
<th>TPS1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Text) and (MeasDlg)</td>
<td>(Text or MDlg)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(Graphic)</td>
<td>(Graphic)</td>
</tr>
</tbody>
</table>

Graphic overrides Text and may have its own buttons. The other way around is not possible. At the same time a MeasDlg may be defined. Graphic overrides Text or MDlg. Text and MDlg are mutually exclusive. Only one can be defined at once. All three dialog types may have their own buttons.
### 11.4.2 Recording Format Settings

<table>
<thead>
<tr>
<th>Deleted:</th>
<th>Replaced by (extended):</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_GetRecFormat()</td>
<td>GSI_GetRecMask ()</td>
</tr>
<tr>
<td>GSI_SetRecFormat()</td>
<td>GSI_SetRecMask ()</td>
</tr>
</tbody>
</table>

### 11.4.3 System Dialog Calls

Replacements for old dialog invocation calls:

<table>
<thead>
<tr>
<th>GSI_CommDlg ()</th>
<th>CSV_SysCall (CSV_EFNC_GeoComSetup, Caption)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_SelectTemplateFiles() and GSI_Setup ()</td>
<td>CSV_SysCall (CSV_EFNC_Setup, Caption)</td>
</tr>
<tr>
<td>GSI_StationData ()</td>
<td>CSV_SysCall (CSV_EFNC_SetStation, Caption)</td>
</tr>
<tr>
<td>GSI_TargetDlg ()</td>
<td>CSV_SysCall (CSV_EFNC_TargetData, Caption)</td>
</tr>
</tbody>
</table>
11.4.4 EDM Mode Changes

Replacement for `EDM_MODE` by the extended `BAP_SetMeasPrg`.

<table>
<thead>
<tr>
<th>Old EDM modes</th>
<th>New defined modes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EDM_SINGLE_STANDARD</code></td>
<td><code>BAP_RED_TRK_DIST</code></td>
</tr>
<tr>
<td><code>EDM_SINGLE_EXACT</code></td>
<td><code>BAP_SINGLE_REF_STANDARD</code></td>
</tr>
<tr>
<td><code>EDM_SINGLE_FAST</code></td>
<td><code>BAP_SINGLE_REF_FAST</code></td>
</tr>
<tr>
<td><code>EDM_CONT_STANDARD</code></td>
<td><code>BAP_SINGLE_REF_VISIBLE</code></td>
</tr>
<tr>
<td><code>EDM_CONT_EXACT</code></td>
<td><code>BAP_SINGLE_RLESS_VISIBLE</code></td>
</tr>
<tr>
<td><code>EDM_CONT_FAST</code></td>
<td><code>BAP_CONT_REF_STANDARD</code></td>
</tr>
<tr>
<td><code>EDM_UNDEFINED</code></td>
<td><code>BAP_CONT_REF_FAST</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_CONT_RLESS_VISIBLE</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_AVG_REF_STANDARD</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_AVG_REF_VISIBLE</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_AVG_RLESS_VISIBLE</code></td>
</tr>
</tbody>
</table>

11.4.5 Interface Changes

The following routines got a new interface.

- `GSI_ImportCoordDlg ()`
- `GSI_ManCoordDlg ()`

Refer to the reference manual to get the new interfaces.

11.4.6 Deleted and Added Identifiers and Types:

<table>
<thead>
<tr>
<th>TPS1000</th>
<th>TPS1100</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deleted:</strong></td>
<td><strong>New:</strong></td>
</tr>
<tr>
<td><code>CSV_MAX_USERS</code></td>
<td><code>CSV_WITH_REFLECTOR</code></td>
</tr>
<tr>
<td><code>CSV_ILLEGAL_USERNR</code></td>
<td><code>CSVWITHOUT_REFLECTOR</code></td>
</tr>
<tr>
<td><code>RC_CSV_ILLEGAL_USERNR</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deleted</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EDM_COMERR</code></td>
<td><code>EDM_NOSIGNAL</code></td>
</tr>
<tr>
<td>EDM_PPM_MM</td>
<td>EDM_METER_FEET</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>New:</td>
<td></td>
</tr>
<tr>
<td>MMI_SHIFT_CODE_KEY</td>
<td>For MMI_SetAngleRelation()</td>
</tr>
<tr>
<td>MMI_UNDEF_LANG</td>
<td>For MDIg routines:</td>
</tr>
<tr>
<td>MMI_DATE_JP</td>
<td></td>
</tr>
</tbody>
</table>

| Deleted:            | New:          |           |           |
| MMI_MENU_EXTRA      | MMI_MENU_PROGRAMS |          |           |
| MMI_MENU_CONFIG     | MMI_MENU_PROGMENU |          |           |
| MMI_MENU_AUTOEXEC   | MMI_MENU_AUTOEXEC |          |           |

<table>
<thead>
<tr>
<th>New GSI_ID values:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_SHZ</td>
<td>GSI_ID_CD_DSC</td>
<td>GSI_ID_PTCD_DSC</td>
<td>GSI_ID_PV_CD</td>
</tr>
<tr>
<td>GSI_ID_PV_PTD</td>
<td>GSI_ID_ACT_PTID</td>
<td>GSI_ID_BACKID</td>
<td>GSI_ID_APP_DATA0</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA1</td>
<td>GSI_ID_APP_DATA2</td>
<td>GSI_ID_APP_DATA3</td>
<td>GSI_ID_APP_DATA4</td>
</tr>
<tr>
<td>GSI_ID_APP_DATA5</td>
<td>GSI_ID_APP_DATA6</td>
<td>GSI_ID_APP_DATA7</td>
<td></td>
</tr>
</tbody>
</table>
GSI_ID_APP_DATA8
GSI_ID_APP_DATA9
GSI_ID_APP_DATA10
GSI_ID_APP_DATA11
GSI_ID_FS_SCALE

New GSI_POINT_TYPE:
GSI_BACKSIGHT
GSI_POINT_CODE

GSI_PAR_* parameters
see GSI system functions.

Deleted:
TPS1100
TPS1700
TPS1800
TPS5000
TPS2003

New:
TPS1102
TPS1103
TPS1105

Old TPS_FAM_Type:
iClass
1EDMBuiltIn
1EDMTypeII
1Motorized
1ATR
1EGL
1DBVersion
1DiodeLaser
1LaserPlummet
1Simulator

New TPS_FAM_Type:
iClass
1EDMBuiltIn (always TRUE)
1EDMTypeII (always FALSE)
1EDMTypeIII (always TRUE)
1EDMReflectorless
1Motorized
1ATR
1EGL
1LaserPlummet
1AutoCollimation
1Simulator

New:
BAP_PRISM_MINI
### 11.4.7 Changes in System Functions

Deleted, because there is no equivalent function at the TPS1100 series instruments:

- BAP_GetFunctionality ()
- BAP_SetFunctionality ()
- BAP_GetFunctionalityDlg ()
- CSV_GetCurrentUser ()
- CSV_SetCurrentUser ()
- CSV_GetDL ()
- CSV_SetDL ()
- CSV_GetUserInstrumentName ()
- CSV_SetUserInstrumentName ()
- CSV_GetUserName ()
- CSV_SetUserName ()
- GSI_GetStdRecMask ()
- GSI_GetStdRecMaskAll ()
- GSI_GetStdRecMaskCartesian ()

Replaced by equivalent functions:

- GSI_WiDlg ()
- GSI_StartDisplay ()
- GSI_GetStdDialogMask ()

Enhanced in certain ways. See the extended identifiers and constants above or refer to the reference manual:

**WI-values**

- CSV_GetPrismType ()
- CSV_SetPrismType ()
- CSV_GetInstrumentFamily ()
- GetMemoryCardInfo ()
- MMI_GetAngleRelation ()
- MMI_SetAngleRelation ()
- MMI_SetDateFormat ()
- MMI_GetDateFormat ()

New functions see reference manual for further details:
Interapplication and system calls

CSV_SysCallAvailable ()
CSV_SysCall ()
CSV_LibCall ()
CSV_LibCallAvailable ()

11.4.8 Returncodes

Their definitions have been coupled totally to the definitions of the TPS1100 firmware. Please refer to the Appendix F in the reference manual for a detailed listing.
12 Changes in GeoBASIC Release 1.30

The Release 1.30 of GeoBASIC contains several new subroutines. It reflects user requests and improvements in the TPS1100 Series firmware Release 2.0.

Note: This GeoBASIC Release needs at least the TPS1100 Series firmware Release 2.0.

The following paragraph shows the changed items. For a detailed explanation, please see the “GeoBASIC Reference Manual”

12.1.1 New functions in Release 1.30

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SearchPrism</td>
<td>search prism</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>returns if an alternative user task was running (i.e. FNC or PROG was pressed)</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>returns the internal instrument temperature</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>resets the “WasRunning”-flag</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>returns if distance tracking is running</td>
</tr>
<tr>
<td>GSI_ExecQCoding</td>
<td>executes Quick-Coding with/without recording</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>starts a distance measurement after changing the distance mode (new buttons in FNC menu)</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>returns the current measurement display number</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>returns if a valid code-list for Quick-Coding is selected</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>returns the current recording mask</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>returns the recording order measurement-code or code-measurement block</td>
</tr>
<tr>
<td>GSI_GetWiEntryText</td>
<td>Get coding text-data from the Theodolite data pool</td>
</tr>
</tbody>
</table>
GSI_SelectCode
select a code-list-code, but without recording it (allows the recording in another format)

GSI_SetMDlgNr
changes the measurement dialog (used i.e. for >DISP buttons)

GSI_SetQCodeMode
enables Quick-Coding

GSI_SetRecMaskNr
changes the recording mask

gS1_SetRecOrder
defines the recording order

MMI_GetVAngleMode
returns if the V-angle is running (even if a valid distance is available)

MMI_SetVangleMode
defines the V-angle mode

TMC_GetAtmCorr
Gets the atmosphere part of distance measurement corrections

TMC_GetGeomProjection
Gets the projection part of distance measurement corrections

TMC_GetGeomReduction
Gets the reduction to the reference part of distance measurement corrections

TMC_GetInclineStatus
returns the inclination status (i.e. ready for recording)

TMC_SetAtmCorr
Sets the atmosphere part of distance measurement corrections

TMC_SetGeomProjection
Sets the projection part of distance measurement corrections

TMC_SetGeomReduction
Sets the reduction to the reference part of distance measurement corrections
12.1.2 New constants in Release 1.30

GSI_GET_NEXT
GSI_MAX_DLG_LINES
GSI_MAX_MDLG_MASKS
GSI_MAX_REC_MASKS
GSI_MAX_REC_WI
GSI_MULTI_REC
GSI_NO_FILE_CHANGE
GSI_SEARCH_FROM_END
TPS1101

12.1.3 New datatypes in Release 1.30

HzAngle
VAngle
TMC_GEOM_PROJECTION_Type
TMC_GEOM_REDUCTION_Type
TMC_ATM_TEMPERATURE_Type

12.1.4 New CSV_SysCall constants in Release 1.30

CSV_SFNC_CheckOrientation
CSV_SFNC_CurrentSetPpmDlg
CSV_SFNC_DefSearchAreaDlg
CSV_SFNC_LoadApplDlg
CSV_SFNC_LoadSysLangDlg
CSV_SFNC_SetDefaultSearchRange
CSV_SFNC_ToggleMeasPrgFastRapidTrk
CSV_SFNC_ToggleMeasPrgRefRL
CSV_SFNC_ToggleMeasPrgStdTracking
CSV_SFNC_ToggleSearchArea
CSV_SFNC_ToggleVAngleMode
1 Introduction ........................................................................................................ 1-3
2 Installation ....................................................................................................... 2-1
  2.1 Setup .......................................................................................................... 2-1
3 Creating a GeoBASIC Application .................................................................. 3-1
  3.1 GBStudio development environment ......................................................... 3-1
  3.2 Typical Development Cycle ...................................................................... 3-7
  3.3 Project Handling ....................................................................................... 3-3
  3.4 Common Problems .................................................................................. 3-4
  3.5 Compiler Limitations ............................................................................. 3-5
4 Executing a GeoBASIC Program on the theodolite .................................... 4-6
  4.1 Loading a GeoBASIC program .................................................................. 4-6
5 Executing a GeoBASIC Program on the Simulator .................................. 5-1
  5.1 General ...................................................................................................... 5-1
  5.2 User Interface .......................................................................................... 5-1
  5.3 Loading and executing GeoBASIC programs ......................................... 5-2
  5.4 Configuration of the Simulator .............................................................. 5-3
  5.5 GeoCom Mode ....................................................................................... 5-4
  5.6 SWTheo Mode ....................................................................................... 5-4
  5.7 Commonly asked questions and answers .............................................. 5-7
6 Additional Debugging Functions ................................................................. 6-1
7 Multiple Language Support .......................................................................... 7-1
  7.1 Text Utility ............................................................................................... 7-2
8 Typical GeoBASIC Programming .................................................................. 8-1
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>The Text Dialog</td>
<td>8-1</td>
</tr>
<tr>
<td>8.2</td>
<td>The Graphics Dialog</td>
<td>8-5</td>
</tr>
<tr>
<td>8.3</td>
<td>Naming conventions</td>
<td>8-9</td>
</tr>
<tr>
<td>9</td>
<td><strong>Refined GeoBASIC Concepts</strong></td>
<td>9-1</td>
</tr>
<tr>
<td>9.1</td>
<td>Units</td>
<td>9-1</td>
</tr>
<tr>
<td>9.2</td>
<td>The User Measurement Dialog</td>
<td>9-2</td>
</tr>
<tr>
<td>9.3</td>
<td>TPS1100 Configurability</td>
<td>9-5</td>
</tr>
<tr>
<td>9.4</td>
<td>Interapplication-Call</td>
<td>9-8</td>
</tr>
<tr>
<td>9.5</td>
<td>System Function Call</td>
<td>9-9</td>
</tr>
<tr>
<td>9.6</td>
<td>System Event Generation</td>
<td>9-10</td>
</tr>
<tr>
<td>10</td>
<td><strong>GeoBASIC Sample Programs</strong></td>
<td>10-1</td>
</tr>
<tr>
<td>10.1</td>
<td>MeanHz — Mean Value of Horizontal Angle Measurements</td>
<td>10-1</td>
</tr>
<tr>
<td>10.2</td>
<td>Sample Programs</td>
<td>10-8</td>
</tr>
<tr>
<td>11</td>
<td><strong>Porting a TPS1000 Originated Program</strong></td>
<td>11-1</td>
</tr>
<tr>
<td>11.1</td>
<td>TPS1100 Hardware Related Changes</td>
<td>11-1</td>
</tr>
<tr>
<td>11.2</td>
<td>Changes to the Simulator</td>
<td>11-2</td>
</tr>
<tr>
<td>11.3</td>
<td>New constructs in GB_1100</td>
<td>11-2</td>
</tr>
<tr>
<td>11.4</td>
<td>GeoBASIC Source Changes</td>
<td>11-3</td>
</tr>
<tr>
<td>12</td>
<td><strong>GeoBASIC Releases</strong></td>
<td>12-1</td>
</tr>
<tr>
<td>12.1</td>
<td>Changes in GeoBASIC Release 1.30</td>
<td>12-1</td>
</tr>
<tr>
<td>12.2</td>
<td>Changes in GeoBASIC Release 2.10</td>
<td>12-3</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

GeoBASIC is a programming language for LEICA theodolites and their simulation on personal computers. The core language appears similar to today's common Windows BASIC dialects, thereby it is easy to learn and use. However, GeoBASIC's main power lies in its ability to use many of the existing theodolite subsystems and dialogs, just by calling an appropriate built-in function: for setting parameters, measuring, geodesy mathematics, and many things more. These tools at hand, the programmer can quickly and flexibly build sophisticated geodesy applications.

The user manual first describes the installation of GeoBASIC on a PC (Chapter 2). Then, after learning how to create a GeoBASIC application (Chapter 3), it will be shown how to actually load and execute a program on a LEICA theodolite (Chapter 4) and on the Windows simulation (Chapter 5).

As these technicalities are mastered, the main topic is programming in GeoBASIC. This manual will give you several hints on typical GeoBASIC programming (Chapter 8), and introduces you to the design and programming of the theodolite user interface and refined GeoBASIC concepts (Chapter 9).

Finally, GeoBASIC example programs are presented (Chapter 10). The reader will find a sample code for measuring and computing the mean value of several horizontal angles. Moreover some introductory examples are given to tell how special problems can be treated.

| Note | All the details of the GeoBASIC language and system functions are composed in the "GeoBASIC Reference Manual". |

---

TPS1100-Version 2.10 1-3
2 INSTALLATION

The requirements for using GeoBASIC are a Personal Computer based on an Intel 486 processor or higher and at least 8MB of main memory. The installation of the whole development environment occupies about 10 MB of disk space, excluding the PDF version of the manual. The delivered software needs Microsoft Win95, Win98 or WinNT to run successfully.

2.1 SETUP

The following directory structure is created during the installation per default. Notice that the location of this directory tree is user definable. Hence it is not a granted to be exactly that location. Notice also that the CodeConverter application is installed in a separate Setup installation procedure.

```
...+-SurveyOffice
    |   
    +--UserTools
        |   
        +--TPS1100Tools
            |   
            +-- CodeConverter
            |   
            +-- GBSamples
```

Content of the directories (only the main objects are listed):

- **TPS1100Tools\**
  - TPS1100.exe  
    TPS Simulator for TPS1100 Series
  - GBStudio.exe  
    GeoBASIC IDE application
  - GBI_1100_xxx.prg  
    GeoBASIC Interpreter for TPS1100 series *)
  - ...  
  - CodeConverter\**
    - CGB_Dlg.exe  
      CODE to GeoBASIC converter
    - Code_ex1.cod  
      CODE sample
    - GBC_xxx.exe  
      GeoBASIC Compiler for TPS1000 series *)
    - GBI_xxx.prg  
      GeoBASIC Interpreter for TPS1000 series *)
    - GBI_1100_xxx.prg  
      GeoBASIC Interpreter for TPS1100 series *)
• …

Several TPS1100Sim specific directories which contain language files, code lists, configurations and things like that.

* xxx means: i.e. 210 for Release 2.10

**Loading the GeoBASIC Interpreter:**

The GeoBASIC Interpreter will be loaded automatically with the loading of the first application into the theodolite using the Software Upload for TPS1100. Hence you have to copy the GeoBASIC Interpreter (GBI_TPS1100_xxx.prg) into the same directory as the application before loading it. Otherwise you will get an error message. (For details, please see Chapter 4.1 Loading a GeoBASIC program or 5.3 Loading and executing GeoBASIC programs)
3 Creating a GeoBASIC Application

Starting from the specification of a GeoBASIC application, several steps have to be performed until the program can be executed on the theodolite or by simulation:

1. Write the program,
2. compile the program,
3. load the program, either onto the simulation or the theodolite, and
4. start the execution of it.
5. if the execution fails, start a debugging session.

3.1 GBSTUDIO DEVELOPMENT ENVIRONMENT

GBStudio is an integrated development environment and includes a source editor, compiler, project handling and a source level debugger. It is able to debug GeoBASIC 2.10 applications for TPS1100 series total stations. Both, the TPS simulator and the TPS device as the execution platform are supported.
GBStudio contains several views for different purposes. The main source view is for showing/editing source files. The ‘Open Files’-tab can be used to switch quickly between different source windows. Toolbars help the user to start actions with one mouse click. The ‘Build/Output’-window is used to display informative messages of the compiler and during the debugging session for the user.

Use the integrated help system to get more descriptive explanations of what can be done with GBStudio. You can invoke the Help documentation by either using the context-help-cursor (Edit toolbar) or the shortcut F1, which opens the content page.

3.1.1 The Editor

It establishes a modern programming language editor, which supports syntax and keyword highlighting, multilevel undo/redo, Intellisense and Tooltip info, Bookmarks, indent and outdent of a block of source lines, and several other features.

The ‘Workspace Preferences’-dialog can be used to customize the features, which should be active during debugging.
To choose a different font use the ‘Font …’-buttons in the ‘Font’-tab, which will offer a dialog to choose one of the installed fonts on the system. Fonts can be chosen separately for the Editor window, Build/Debug output window and for the Watch Variable window.

3.1.2 The Compiler

The source-file has to be compiled before it can be loaded and executed. Compiling the source file with the GeoBASIC compiler results into 3 files, one for the executable object itself (file extension “.gba”; i.e. sample.gba), one for the language data (file extension “.lng”; i.e. sample.lng) and a debug-info file (file extension “.gbd”; i.e. sample.gbd). The first two files are necessary to execute the program, either on a LEICA theodolite or with the simulator on a personal computer. The debug-info file is necessary for debugging a program using GBStudio. See the following diagram:

The compiler is fully integrated in the development environment. The compilation of the source file is just one mouse click away. If an error occurs the editor will place the cursor automatically at the position of the error in the source window. Use Ctrl-F1 to get a more descriptive explanation of what caused the failure of the compilation process.

Depending on the compiler settings also the debug info file is generated which is necessary for debugging the application.
Depending on the selected project type, use either the ‘Default Project Preferences’-dialog or the ‘Project Preferences’-dialog to set the build options for the compiler.

The compiler understands the following options:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>The language on which the resulting application is based on. The default is ENGLISH, other languages are FRENCH, GERMAN, etc.</td>
</tr>
<tr>
<td>Character Set</td>
<td>The character set on which the application is based on. The default character set is 0.</td>
</tr>
<tr>
<td>Output File</td>
<td>The name of the resulting applications file name. If it is empty, the resulting files get the same file base name as the source code file.</td>
</tr>
</tbody>
</table>
Output Path  The path where the compiler places the generated application files. The default is the source directory, where the compiler gets the GeoBASIC source file. The path has to be absolute and has to end with a "\" character.

Include Paths  Set one or more directory-paths for include files. The directory path must not have a "\" character at the end.

Generate Statistics  Enable this flag, if you want the compiler to generate some statistical information about the compiled application.

Generate Debug Info  Enable this flag, if you want the compiler to generate a debug info file, which is necessary to debug the application.

3.1.3 The Debugger

The debugger enables the programmer to debug GeoBASIC applications at source level. Operations like Step, Step Over, Run, Set breakpoints and watching the values of variables and some more operations are implemented.

To find errors in the source code an error catcher has been implemented which stops the execution of the application once the Err-variable changes its value. The error catching mechanism can be enabled and disabled during the debugging session at the needs of the developer.

The generated files include time stamp information. With this information GBStudio is able to check if all involved objects are synchronous to each other. This feature also enables GBStudio to debug an application, which may be in use for some time already. The only precondition, which has to be met is, that all files have to be saved for this purpose. Once the source code file changes debugging can only be started if the application is compiled anew. This means also that the application has to be loaded freshly onto hardware, which then initializes all its values. This feature is very valuable if a tested application shows error only after weeks or months of usage.

Depending on the selected project, use either the “Default Project Preferences” dialog or the “Project Preferences” dialog to set the build options for the debugging session.
For debugging the following values can be set:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>This setting determines the execution platform and if TPS over a serial line is served, which COM port should be used for communication.</td>
</tr>
<tr>
<td>Baud Rate</td>
<td>Is available only if one of the serial communication lines has been chosen. Choose an appropriate Baud rate.</td>
</tr>
<tr>
<td>System Idents</td>
<td>Determines the location of the system specific symbols file. Click on the “Browse…” button to get a file chooser dialog.</td>
</tr>
</tbody>
</table>
Entry Point
Since every loadable application on the TPS may have more than one entry point, one has to select a valid entry point of the application. This value can be entered before the debug info has been loaded, or after the debug info load operation. In the latter case choose the entry point by selecting an item from the drop down list.

Catch Runtime Errors
Enable this flag to catch runtime errors.

Number of Watch Variables
Select a value between 1 and 1000 watch variables. The number determines the table size on the server side. This value heavily influences the performance of certain debug operations. If you don’t a big number, then choose a smaller number for better performance.

Size of Shadow Memory
Select a value between 100 and 10000 Bytes. This will be the size of the shadow memory, where the server will keep a backup copy of the registered variables.

3.1.4 The Interpreter and the Firmware

Both have been adapted to provide all the additional functionality. Hence only firmware releases 2.10 and newer support GeoBASIC debugging with GBStudio. Please notice, that GBStudio cannot handle the TPS device state “Sleep Mode” correctly. Please disable the sleep feature of the TPS firmware if you want to avoid tedious timeout errors in GBStudio.

3.2 TYPICAL DEVELOPMENT CYCLE

3.2.1 Open or Create a GeoBASIC main source file

Use the Open File command to open an existing GeoBASIC main source file or create a new file with the document type GBS.
If you choose to open an already existing project, then the defined main source file should be opened automatically.

### 3.2.2 Edit the application.

Type in or change an existing GeoBASIC application source code. Please, refer also to the GeoBASIC reference manual for a complete description of syntax and semantics of GeoBASIC and how to write applications in GeoBASIC.

The editor is capable of automatically correcting the case of keywords. If one types a blank after a keyword this features take place automatically. Switch this feature off in the Workspace Preferences dialog if you don’t want to use this feature.

CTRL-SPACE opens a drop down list of system-defined functions. This can be used to quickly select a system function. When the opening parenthesis is typed the parameter list will be showed as a tool tip and a reminder what the compiler expects. Use SHIFT-CTRL-SPACE anytime to open up this tool tip again. The displayed parameter list depends on the cursor position and moreover on the system function identifier just before the current cursor.

**Note:** Define also an entry point (GLOBAL SUB definition) of the application, which you can choose later to debug. This is the only identifier in a GeoBASIC application, which is case-sensitive. Make sure this entry point is linked to a menu item on the TPS user interface. Otherwise it will not be possible to debug the application (with the exception of the “BasicCodeProgram” type of application).

Save your changes by using CTRL-S or the Save command from the File menu.

### 3.2.3 Build the application

Press function key F7 or use the Build command from either the Build menu or Build toolbar.
If an error occurs, then the editor will place the cursor automatically near the location of the error. Correct the error and recompile it. Repeat these steps until your application compiles without any errors. Use CTRL-F1 if you want to get some more information on the last error occurred.

| Note | The usage of the compiler is protected by a hardware key. Without the right hardware key it is not possible to execute the compiler successfully. If the hardware key is not installed properly or it does not contain the license for the compiler then an error message will be displayed and execution will be terminated. |

### 3.2.4 Start debugging

To start the debug session, choose the platform (TPS simulator or TPS instrument) and specific settings, you want to use, in the Project Preferences dialog. Make also sure the entry point of the application is set properly in the preferences dialog.

1. Switch on the debugging platform.
2. When using the TPS device:
   - Load the GeoBASIC interpreter.
3. Load the application you want to debug. (For details, please see Chapter 4.1 Loading a GeoBASIC program or 5.3 Loading and executing GeoBASIC programs)

| Note: | The application must have been build with ‘Generate Debug Info’ enabled. |
| Note: | GBStudio uses the TPS device when the **GSI settings are active**. The GeoCOM online mode is **not supported** during the debugging process. Make sure the GSI communication settings are:
|       | 19200 Baud,  
|       | No-Protocol,   
|       | 8 Data Bits,   
|       | No-Parity,     
|       | CR/LF as terminator.  
|       | GBStudio **cannot** handle the sleep state of the TPS device correctly. Make sure the ”Sleep after ...”-mode is disabled. |

The application source and the generated files must be synchronous, hence a source file, which has been changed, after the application has been built, cannot be debugged.

Start debugging by pressing the Start button on the Build toolbar or use the corresponding menu located command.

Start the application on the platform. The editor should now get a small mark (in the shape of an right sided arrow) on the left edge of the main source file window, which points to the very first executable statement of this entry point of the application.

### 3.2.5 Debugging

Use the commands of the Debug menu or toolbar to step through the application, set breakpoints, catch errors and watch variables as they change during the debugging process.

In the watch variable view you will be able to edit either the identifier of the watch variable entry or the value itself, if the debugging process is in a HALT state.
3.2.6 Stop debugging

Choose the Stop Debugging command to stop the debugging process. Just in case the application is executing a system function, then the debug server will not be able to terminate the application immediately. Instead the application will be terminated after the system call returns. Nevertheless, GBStudio can terminate the debugging session on the client side.

3.2.7 Watch Variables and Quick Watches

Watch variables can be added to the Watch Variable view by selecting a variable identifier and pressing the shortcut Ctrl-W.

Use the Quick Watch command if you don’t want to add the variable to the Watch Variable view. Instead the value will be printed into the Debug Output window.

Once added to the window it is possible to change either the identifier name or the value of it (if the point of execution is in the scope of the variable). Use a Double-Click on the identifier or the value to enter the edit mode.

### Note:
Changing the value of a string reference parameter is possible too. Since the actual, maximum length of the variable (behind the reference) is unknown, the debugger is unable to protect the memory area following the string variable. Hence, if you change the value of a string reference parameter, be sure that the number of added characters is less than or equal to the declared length.
Note: The identifier name is bound to the current context, which is determined by the selection you made. To choose the same identifier name from a different context one has to select the identifier in the correct context.

Valid watch variable expressions may be of the following form only:

<table>
<thead>
<tr>
<th>Variable Expression</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VariableIdent</td>
<td>s, Err, line</td>
</tr>
<tr>
<td>StructIdent.Element</td>
<td>CurrPt.dHz, GMCircle.Center.dHeight, ArrayIdent(NumConstant) arr(2), field(17,3)</td>
</tr>
</tbody>
</table>

All other possible text strings cannot be handled correctly in the current implementation and will be rejected for registration therefore.

Include exclusively expressions with numerical constants.

### 3.3 PROJECT HANDLING

GBStudio knows two different categories of projects, which are valid exclusively. First the default project, which is valid for any valid GBS-file. And second the so-called 'named' projects, which have the application specific information stored in a file. It should be emphasized that the default project only stores the settings of one project (similar to one main source file) at a time. Once the user chooses another main source file, he has to make sure that the default preferences are set appropriately. E.g., if the two source files have different application entry points, the user has to set it up accordingly.

The default project is active if the user doesn’t choose a project explicitly. Instead the user will just open a plain GeoBASIC source code file.
3.4 COMMON PROBLEMS

The most common problems, which may arise, are:

- GBStudio is not able to establish a connection to the GeoBASIC Debug Server.
  Solution: In the case of debugging with the simulator make sure the TPS simulator is running and “Switched On”. In the case of the TPS device make sure the right COM port has been chosen, the cables are connected and the communication settings are equal on both sides. Notice, that GBStudio only supports serial settings with 8 Bit, 1 Stop Bit, no Parity Bit and CR/LF as a packet terminator. Only the Baud Rate may vary.
- The application, which should be debugged, and/or the interpreter are not loaded. Solution: Load interpreter and/or application first, before you start debugging.
- The program source files are out of synchronicity with the compiled application. Solution: Recompile and reload the GeoBASIC application.
- The Debug Session cannot be started, because the system predefined symbol file could not be found. Solution: Use the “Project Preferences” dialog, Debug-Tab, to specify path and file name of the system predefined symbols.
- The Debug Session cannot be started, because no valid entry point has been chosen. Solution: Use the “Project Preferences” dialog, Debug-Tab, to specify a valid entry point. Valid entry points are defined in the source code as “GLOBAL SUB …” procedure names. Notice: the predefined entry points Install, Init and Stop are not valid entry points.
- During debugging a Step-Into an Include source file doesn’t open the source file and show the next statement. Or the compiler reports the error that he can’t open an Include file. Solution: Make sure that the “Project Preferences” dialog, Build-Tab, field “Include Directories”, contains the right path, where GBStudio can find the include source file.
- The second registering of a variable doesn’t show the associated value. Notice, a variable can be registered only once.
- During debugging the code source cannot be edited. We disabled this during the debug session to keep the source and the loaded application...
synchronous. Stop the debug session to be able to edit the code source again.

- **The debug session hangs.** Conceptually it may happen that a notify message gets lost from the server to the client. Then it might be possible that the “Stop Debug” and “Break” buttons are enabled only. Since the debug server has sent the notify message it waits for the next command. And because the client has missed the notification, it thinks the last command is still being under execution and waits for the never incoming notification.

  Solution. Use the “Break” button to check the current state. If the last command has been finished and above situation was the reason then this initiate a new notification of the current state.

### 3.5 COMPILER LIMITATIONS

The GeoBASIC programmer has to keep some limitations for his applications:

- One simple procedure or function may not contain more than 10 kB of code.
- The maximum size of an application (including memory space) is limited by the free memory size of the theodolite only. If no other applications are loaded there should be free memory up to several hundred kB on a theodolite.
- An application may not have more than 64kB of string literal in total.
- The number of global identifiers is limited to 3000.
- The overall maximum number of identifiers limits the number of local identifiers, which are about 60000.
4 EXECUTING A GEOBASIC PROGRAM ON THE THEODOLITE

As described in the Chapter 3.1.2 The Compiler, compiling a GeoBASIC program results in at least two files, the executable program itself and the language data. Before a program can be executed, these two files have to be loaded into the theodolite first. With the help of the Leica Survey Office Software Upload the two files can be loaded into TPS-memory and run automatically the install procedure of the GeoBASIC program. The install procedure has to take care of adding an item to a menu which links an external procedure of the GeoBASIC program (Global Sub) to an item in a menu list. Additional to this static link there is a more flexible concept to install an application via a user (definable) configuration. For further explanations how to install an GeoBASIC application read Chapter 9.3. If the menu item is added to a menu you can choose it to run a GeoBASIC program.

4.1 LOADING A GEOBASIC PROGRAM

GeoBASIC programs can be loaded into the theodolite using the Software Upload program from the Open Survey Suite. The procedure for loading a GeoBASIC application is as follows:

1. Verify that a serial link between PC and theodolite is established.
2. Switch theodolite into GeoCOM online mode.
3. Start Software Upload program.
4. Press <Transfer Files...> in <Utilities> menu of Software Upload.
5. Choose <Application Program> as Component Type.
6. Select directory which contains the loadable program (*.gba).
7. Choose language if the application supports multiple languages.
8. Select the application in the <Components> window.

Detailed explanations may be found in the documentation of Leica Survey Office - Software Upload.
GeoBASIC programs can also be loaded from the PC-Card into the theodolite using the build-in application loader. For details, please see description in the theodolite documentation.

**Note**

Loading a program with identical names for module and external procedures as an already loaded program replaces this program and all its associated text modules in memory and the items in the menu list. Hence, transferring of more than one program with the same application name may cause unwanted effects.

**Note**

For the build-in loader from the PC-Card, the files (*.GBA und *.lng) must be stored in the PC-Card folder “\TPS\APPL”. If necessarily, the GeoBASIC interpreter (gbi_xxx.prg) is loaded automatically from the same folder.
5 EXECUTING A GEOBASIC PROGRAM ON THE SIMULATOR

5.1 GENERAL
The TPS1100 simulation supports, among other features, the execution and debugging of GeoBASIC applications. The simulation may run in one of two modes:

- GeoCOM mode
- SWTheo mode

Running in GeoCOM mode the simulation operates the (hardware) theodolite connected to the PC via a serial port and uses it as a sensor device. In SWTheo mode, user triggered commands are redirected to the software simulation of the theodolite.

5.2 USER INTERFACE
The TPS1100 simulation main window contains two windows and a dialog box on start-up: the "TPS1100" window and the "Debug" window (see below). The TPS1100 window contains a replication of the (hardware) TPS1100 theodolite’s user interface. In the "Debug" window, debug information are displayed. It is recommended to have always the debug window opened because some of the statements in the GeoBASIC source code (like the WRITE statement) might cause printing text into the "Debug" window.

The dialog box is called “Virtual Theodolite” and is used to type in raw measurement data for the simulation of measurements. See also section 5.6.2 for further explanations.
5.3 LOADING AND EXECUTING GEOBASIC PROGRAMS

The procedure for loading a GeoBASIC application is as follows:
1. Make sure the simulation is turned on.
2. Choose the „Load Basic Application“ entry from the „File“ menu.
3. Choose a desired GeoBASIC executable (extension .gba) and press the „Open“ button.

If the application could be loaded successfully, it can be executed by choosing the menu item (or in the special case of a code program the CODE button in MEAS-mode), which has been added by the Install routine of the application. There is also a more flexible possibility to install the application via a user (definable) configuration. Refer to Chapter 9.3.2 for more information.

If the menu item “Load Basic Application …” is disabled (grey) then make sure no GeoBASIC application is running and maybe it’s necessary to press once or twice the ESC button of the TPS simulator.
5.4 CONFIGURATION OF THE SIMULATOR

The simulation is configurable via the „Configuration“ menu of the simulation main window. Here, the beep may be toggled using the „Beep On“ entry. A check mark left to the „Beep On“ indicates whether it is turned on or off. The „Instr. Connection …“ entry opens a dialog to configure the communication parameters for GeoCOM mode and to switch between GeoCOM and SWTheo mode as shown in the following figure.

Paths can be set for text management, GSI data, code list, GeoBASIC programs and configuration data in the dialog opened by the „Data Path“ menu entry.

It is highly recommended to set the paths, if they are not already set, to the following values:

<table>
<thead>
<tr>
<th>Path</th>
<th>Recommended value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Files</td>
<td>TPS1100Tools\TextDB</td>
</tr>
<tr>
<td>GSI and Log Files</td>
<td>TPS1100Tools\GSI</td>
</tr>
<tr>
<td>Internal Code List</td>
<td>TPS1100Tools\CodeList</td>
</tr>
<tr>
<td>External Code List</td>
<td>TPS1100Tools\CodeListPcCard</td>
</tr>
<tr>
<td>Basic Programs Path</td>
<td>TPS1100Tools\GBSamples</td>
</tr>
<tr>
<td>Configuration Data Path</td>
<td>TPS1100Tools\Config</td>
</tr>
</tbody>
</table>
5.5 GEOCOM MODE

5.5.1 Running the simulation in GeoCom mode

To switch to and run in GeoCOM mode follow this procedure:
1. Switch off simulation by single clicking under the down cursor of the TPS1100 window if not already off.
2. Verify that a serial link between PC and theodolite is established.
3. Switch off hardware theodolite if not already off or switch into GeoCOM online mode.
4. Select the appropriate communication parameters and „GeoCom“ in „Instr. Connection …“ dialog (see above) of the simulation. Confirm with the „OK“ button.
5. Start the simulation again using the „ON“ button of the TPS1100 window. The simulation now tries to communicate with the theodolite. If a connection can be established, and the port you have chosen was „COM1“, the title of the TPS1100 window will be „TPS 1100 <running, GeoCom on com1:>“.

Otherwise a dialog enables the user to choose whether other communication configurations should be tested or not. Notice that this may take up to one minute. If no connection could be established, the SWTheo is activated instead of GeoCOM after displaying a message box.

5.6 SWTHEO MODE

The software theodolite (Virtual Theodolite, SWTheo) is an emulation of a (hardware) theodolite. Its properties may be accessed via the „Meas Data Input…” entry in the „Configuration“ menu while the simulation is running in SWTheo mode. Otherwise this menu entry is disabled.

5.6.1 Running the simulation in SWTheo mode

The procedure for switching to and running the simulation in SWTheo mode is as follows:
1. Switch off the simulation by single clicking under the down cursor of the TPS1100 window if it is not off already.
2. Open the GeoCOM dialog via the „Configuration“ menu.
3. Disable the GeoCOM enable box. Confirm with the „Ok“ button.
4. Start the simulation using the „ON“ button in the TPS1100 window.

5.6.2 User Interface

There are two dialogs to access the SWTheo from the simulation. The first one is called SWTheo dialog with the caption „Virtual Theodolite“ contains fields to change raw sensor data of the SWTheo as well as station data. This dialog is opened from the “Configuration” menu as stated above. The second dialog called SWTheo properties dialog (caption „Virtual Theodolite Properties“) may be triggered from the SWTheo dialog.

5.6.2.1 SWTheo Dialog

The dialog acts as the connection between the SWTheo and its virtual environment. Here, horizontal angle (Hz), vertical angle (V), and slope distance (Dist) to a virtual reflector as well as station data (N0, H0, E0), reflector (Hr) and instrument height (Hi) may be set. User input has to be confirmed using the “Set Data“ button to take effect. Pressing the “Properties” button opens the Subsystems dialog.

Notice also that it is possible to define several sets of values. Choose a set by selecting the corresponding number off the measurement set. The values will be stored until they are changed.
5.6.2.2 SWTheo properties dialog

The SWTheo properties dialog is a tabbed dialog as shown below. Here you can set some basic values.

![Virtual Thendolite Properties](image)

The „Units“ tab depicted in the last figure enables the user to choose between several display units for the SWTheo dialogs. Please notice these values do not change the settings of the simulation.

“Jittering” is supported for angles and distances. This functionality is applied by alternately adding and subtracting random values in a range depending on the angle and distance sliders, respectively. The jittering amplitude increases from left to right position of the slider. If the sliders are in their leftmost position, there is no jittering applied to the virtual sensor data.
5.7 COMMONLY ASKED QUESTIONS AND ANSWERS

Q:
After starting the simulation and turning on in SWTheo mode, the text „xxx“ will be displayed as the title of some or all of the function buttons. How can I avoid this problem?

A:
Some or all of the text data base files are not contained in the directory referenced by „Text Management Data Path“. Use the „Data Paths“ entry of the „Configuration“ menu to set it accordingly.

Q:
After loading a GeoBASIC program, the expected menu item does not appear in the dialog. What did I wrong?

A:
The menu manager needs an event to reread the menu definition. Press the ESC key to rebuild the menu.
6 ADDITIONAL DEBUGGING FUNCTIONS

There are a few additional features, which may be helpful while debugging the program.

For the simulator:

• The command *Write* writes the given argument to the debug window. This will have no effects on the TPS.
• The same is valid for *Send*, because it will be redirected to the debug window. But, of course, on TPS it will send data over the data link.
• If an error occurs then a message will be written to the debug window, showing the error code and the name of the system routine, which caused the error.

For the simulator and the TPS:

• *MMI_PrintStr* can be used to display and track results and errors.

See also the list of return codes in the appendix of the Reference Manual.
7 MULTIPLE LANGUAGE SUPPORT

The TPS 1100 series system software supports internationalisation in such a way that text fragments are handled extra to an application. Accessing these fragments will be done internally by tokens. GeoBASIC supports this technique in certain system calls. Anytime a system routine is called which needs a _Token instead of a string then this token will be added to the text token database. The compiler handles this automatically for the programmer and produces the already mentioned lng-file.

This text token database is the basis for supporting multiple languages. With the Text Utility you can produce new text token databases (mxx-files) in other languages. Loading the derived lxx-files on the TPS system for enabling the user to choose between the provided languages. (’xx’ stands for the language abbreviation.)

Diagram: Generate language files.
Strings which are not passed to a \texttt{*_Token} parameter cannot be handled with the Text Utility. They are hard coded into program object code. The only way to internationalise them is to use \texttt{MMI\_GetLangName} to select an appropriate text string in GeoBASIC code separated by a conditional statement. See sample file “language.gbs”.

7.1 TEXT UTILITY

The TPS1100/1000 Text Utility (Text Translation Tool) supports GeoBASIC text files. This section describes the most important steps of generating multiple language files. The following picture shows the Text Utility after the import of a GeoBASIC text file:

7.1.1 Generating new language files

For creating a multiple language application, the following steps are necessary:

1. After starting the Text Utility press the \textbf{-} button, select GeoBASIC Text Files (\texttt{*.l??}) in the choice list “File of type:” and open the generated \texttt{*.lng} file (i.e. \texttt{sample.lng}). Answer the question “Do you want to convert this file?” with YES. In the next dialog you can specify the path and the version of the text database which is generated from the \texttt{*.lng} file (i.e. \texttt{sampl100.mng}). The version is automatically included at the end of the file name. Press OK to start the conversion.

2. Press the \textbf{-} button, select a language in the choice list “New language”, enter the path of the new language database and press OK to start the
generation of the new language database (i.e. sampl100.mge). Now translate the text in column “Text”.

**Note** Do not edit the first token with the text “\n\X1\n”. This string is needed by the GeoBASIC Interpreter. Also the special strings for MMI_INVERSE_ON ("\aR+\a") and MMI_INVERSE_OFF ("\aR-\a") must be left unchanged.

After the translation press the —button, select the path and enter the name of the loadable language file and press OK to start the generation of the file (i.e. sample.ige).

### 7.1.2 Updating translated language files

After changing the GeoBASIC source file and re-compiling it, the following steps for updating the translated language files are necessary:

1. Press the —button again and open the generated *.lng file (i.e. sample.lng). The version of the text database which is generated must be increased (i.e. sampl101.mng).

2. Press the —button and open the target language you want to update (i.e. sampl100.mge). Edit the target language text column (indicated with T1). After updating the whole column press —button to generate the new loadable language file.
8 TYPICAL GEOBASIC PROGRAMMING

In this chapter some advice is given on how to program in GeoBASIC. The main attention is given to the user dialog — which is probably the most theodolite-specific part in GeoBASIC programming (besides using the system functions). Afterwards a proposal for naming conventions for GeoBASIC identifiers is given.

Note To make programs easy and intuitive to use, the programmer should follow the given "standards" rather strictly. Moreover (s)he should have a basic understanding of the way how topographical surveying and mapping is actually performed.

8.1 THE TEXT DIALOG

8.1.1 The objects of the text dialog

The following text dialog is not a practical example, it shows only the most important text dialog objects:

<table>
<thead>
<tr>
<th>Dialog line</th>
<th>Object name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;BASIC\Text Dialog Objects&gt;</td>
<td>Caption line: It is composed of the short caption &quot;BASIC&quot; and the caption &quot;Text Dialog Objects&quot;.</td>
</tr>
<tr>
<td>&lt;I am a text dialog object.&gt;</td>
<td>String</td>
</tr>
<tr>
<td>&lt;10587&gt;</td>
<td>Integer value</td>
</tr>
<tr>
<td>90479.568</td>
<td></td>
</tr>
<tr>
<td>50,000 g</td>
<td></td>
</tr>
<tr>
<td>List Item 1</td>
<td></td>
</tr>
</tbody>
</table>
Double (floating point) value without unit
<50.000 g> Double (floating point) value with unit: If the type of the double value is Angle, Distance, Subdistance, etc. the according unit is printed automatically
<List Item 1 ▶> List: It is for selecting an item among several with the cursor keys
<CONT> Button: The buttons inform the user about the functionality of the function key (F1..F6).

8.1.2 Creating a text dialog

A new text dialog is created by `MMI_CreateTextDialog`.

`MMI_CreateTextDialog(6, "BASIC", "Text Dialog Objects", "My help text.")`

A text dialog with a short caption, here "BASIC", and a caption "Text Dialog Objects" is created. There is a total of 27 characters for the three parts, i.e. short caption, separation character (\ printed automatically) and caption. 6 lines (start counting from the first line below the caption – which is 0 – up to line 5) can be used. All lines are empty after the creation. The help text is set to "My help text." — it is shown when the user presses Shift-F1 and the help functionality of the theodolite is enabled.

8.1.3 Representation of the dialog objects

For every input and output the position on the display must be specified. The display is organized in lines and columns. The left upper position has line and column number 0. The line number is rising down and the column number is rising to the right. A display line is 29 characters wide. At most 6 lines are visible at any time, if the dialog contains more lines (up to 12 are possible) it is scrolled when necessary.

For floating point input/output a kind (for instance horizontal angle, distance, etc.) can be specified. Data is automatically transformed to the unit associated to the
kind according to the theodolite settings. Unit conversions are done by the system, all values with units defined in basic are considered to have to SI units. (See Chapter 9.1)

All numeric output appears right aligned in their field (specified by coordinates and length). String output appears left aligned.

Each input/output routine needs a parameter lValid which defines if the value of the object is valid or not. If a value is not valid five dashes are displayed instead of the value.

Every numeric input/output needs a parameter iLen which determines the total character length of the field. If the length is to short for the representation of the numeric value, the field will be filled with the character ‘x’.

### 8.1.4 Output in text dialog

- **Strings:**
  
  ```basic
  MMI_PrintStr(0, 0, "I am a text dialog object.", TRUE)
  ```

  Parameters: column, line, string, lValid

- **Integer values:**
  
  ```basic
  MMI_PrintInt(10, 1, 10, 10578, TRUE)
  ```

  Parameters: column, line, iLen, integer value, lValid

- **Double (floating point) values without unit:**
  
  ```basic
  MMI_PrintVal(10, 2, 10, 3, 90478.568, TRUE, MMI_DEFAULT_MODE)
  ```

  Parameters: column, line, iLen, decimals, double value, lValid, Mode

- **Double (floating point) values with unit:**
  
  ```basic
  DIM hz AS Angle
  hz = PI/4
  MMI_PrintVal(10, 3, 8, 3, hz, TRUE, MMI_DIM_ON)
  ```

  Parameters: column, line, iLen, decimals, double value, lValid, Mode

### 8.1.5 Input in text dialog

Input is roughly dual to the output, except that the input functions return the button id of the button that terminated the edit process. For all numeric values there are the minimum and maximum values defined. The value is only valid, if it is between them.
- **Strings:**
  
  ```basic
  MMI_InputStr(17, 3, 10, sInput, lValid, iButtonId)
  ```
  
  Parameters: column, line, string variable, lValid, button

- **Integer values:**
  
  ```basic
  MMI_InputInt(24, 4, 4, 100, 200, iValue, lValid, iButtonId)
  ```
  
  Parameters: column, line, iLen, minimum value, maximum value, integer variable, lValid, button

- **Double (floating point) values without unit:**
  
  ```basic
  MMI_InputVal(19, 4, 8, 2, 0, 399.99, MMI_DEFAULT_MODE, dValue, lValid, iButtonId)
  ```
  
  Parameters: column, line, iLen, decimals, minimum value, maximum value, mode, double variable, lValid, button

- **Double (floating point) values with unit:**
  
  ```basic
  MMI_InputVal(19, 4, 8, 2, 0, 399.99, MMI_DIM_ON, dValue, lValid, iButtonId)
  ```
  
  Parameters: column, line, iLen, decimals, minimum value, maximum value, mode, double variable, lValid, button
• **List:** Lists take a variable of a predefined type as parameter.

```geo
TYPE ListArray (25) AS String30 END
```

This definition determines the maximum number of entries in a list to be 25, each one is a string of type String30. We create a list with 4 items and use the second entry as default (initial selection).

```geo
DIM aList AS ListArray
DIM iIndex AS Integer

aList(1) = "List Item 1"
(aList(2) = "List Item 2"
(aList(3) = "List Item 3"
(aList(4) = "List Item 4"
iIndex = 2
MMI_InputList(8, 4, 12, 4, MMI_DEFAULT_MODE, aList, 
iIndex, lValid, iButtonId)
```

Parameters: column, line, iLen, number of items, mode, list variable, index, lValid, button

### 8.2  THE GRAPHICS DIALOG

#### 8.2.1 Positioning on the display

Every graphics function needs the position on the display. The graphics display is organized in x- (horizontal) and y-pixels (vertical). The left upper position has x-pixel and y-pixel number 0. The x-pixel number is rising to the right and the y-pixel number is rising down. The size of the display is 232 times 48 pixels.

#### 8.2.2 Creating a graphics dialog

Calling `MMI_CreateGraphDialog` creates a new graphics dialog.

```geo
MMI_CreateGraphDialog("BASIC", "Graphics Dialog", 
"My help text.")
```

A graphics dialog with short caption "BASIC" and caption "Graphics Dialog" is created. The help text is set to "My help text." — it is shown when the user presses Shift-F1 and the help functionality of the theodolite is enabled.
8.2.3 Graphics functions

After having created the graphics dialog, the graphics functions may be used. (E.g. \texttt{MMI\_DrawLine}, \texttt{MMI\_DrawCircle}, \texttt{MMI\_DrawText}, etc. See the "Reference Manual" for a detailed description.)

8.2.4 Deleting a dialog

When a dialog is not used any more it must be deleted. The name of the dialog deletion procedure is for text, measurement and graphics dialogs the same:

\texttt{MMI\_DeleteDialog()}

8.2.5 Mixing text and graphics dialogs

There can be only one text dialog at a time, i.e. an existing text dialog must be deleted with \texttt{MMI\_DeleteDialog} before a new one can be created with \texttt{MMI\_CreateTextDialog}.\textsuperscript{1} The same holds for a graphics dialog (with the appropriate creation procedures).

But a graphics dialog may be opened while a text dialog is active. (Note: The reverse is not the case: a text dialog may not be opened while a graphics dialog is open.) If a text dialog and a graphics dialog are open, the graphics dialog has priority, i.e. all future function calls are related to the graphics dialog (until it is closed). For example, \texttt{MMI\_AddButton} (see below) will add the button to the graphics dialog, and all the display functions must be for graphic dialogs (such as \texttt{MMI\_DrawCircle}, etc.).

8.2.6 Adding buttons

The user may add buttons to a dialog. (These buttons will be added to the \emph{defined buttons} of the dialog.) When adding a button it must be specified what text should be displayed for that button. Such a text can be up to five characters long and is displayed centred above the button.

Each button has an identification associated. This button id is needed

\textsuperscript{1} An existing text dialog is deleted automatically if a new text dialog is created.
for specifying which button is to add in MMI_AddButton, and
• checking what button was pressed or that is returned from a system function.

Example:

We add the F1-button to the currently opened dialog, giving the meaning "CONT" to it.

```
MMI_AddButton( MMI_F1_KEY, "CONT" )
```

**Note**  The button id's are defined as constants in the compiler.

### 8.2.7 Responding to buttons

There are two procedures for coping with button presses:

• MMI_CheckButton queries whether there was a button pressed or not, and
• MMI_GetButton retrieves a pressed button. If there was no button pressed it waits until one is pressed. The second parameter to MMI_GetButton (the in-parameter bAllKey) determines what buttons are accepted:
  - If it is TRUE, any button is accepted.
  - If it is FALSE, only ESC, or a defined button (added with MMI_AddButton) are accepted.
Example:

The example does some work in a loop until Shift-F6 is pressed. As long as there is no button pressed, the display is constantly updated (e.g. the current angles from the theodolite are displayed). If there is a button pressed, this button is handled.

```basic
'bDone must be initialized
bDone = FALSE
DO WHILE NOT bDone    'as long as the job is not done
    'check for defined buttons and get its id
    MMI_GetButton( buttonId, FALSE )
    SELECT CASE buttonId    'handle it
        CASE MMI_F4_KEY
            'handle MMI_F4_KEY
        CASE MMI_SHF6_KEY
            bDone = TRUE    'that’s it,
            'terminate loop
        CASE ...
            'here go the other handled keys
        ELSE
            'here go the unhandled keys
        END SELECT
    'update the display
LOOP
```

8.2.8 Standard key binding

It is clear that for the user it is important that the same name\(^2\) — and moreover the same key — always has the same meaning associated (at least conceptually). An exception is the F1-key, its meaning is not the same in a measurement dialog and in a configuration dialog. In the following table there are the standard key bindings with the caption, the text which is displayed above the keys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Caption</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 in measurement dialog</td>
<td>ALL</td>
<td>Does first DIST, then REC. (See below)</td>
</tr>
<tr>
<td>F1 in configuration dialog</td>
<td>CONT</td>
<td>Continues to the logically following dialog.</td>
</tr>
</tbody>
</table>

\(^2\) For instance, the user of a LEICA theodolite assumes that DIST takes the distance (with the common dialogs), ALL does DIST and then REC, etc.
### 8.3 NAMING CONVENTIONS

We propose some naming conventions for GeoBASIC. More extensive conventions can be found in the naming conventions for Microsoft Access (which are tied closely to Visual Basic conventions).³

#### 8.3.1 Variable names

Variable names of simple types (i.e., all the scalar types and strings) may be tagged to indicate their type. Prefixes are always lowercase so your eye goes past them to the first uppercase letter — where the base name begins. If the base name consists of more than one word, upper case letters within the name are used to distinguish its parts.

**Note** These naming conventions carry only a semantics for the programmer, not for the compiler.

---

The base name succinctly describes the object. For example, PointNumber or just PointNo for the number of a point. Object tags are short abbreviations and simplifications describing the type of the object. For example, the tag 'i' in iPointNo denotes that the type of the variable is Integer. The following table lists the tags for the GeoBASIC types.

<table>
<thead>
<tr>
<th>type</th>
<th>tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>i</td>
</tr>
<tr>
<td>Logical</td>
<td>l</td>
</tr>
<tr>
<td>Double</td>
<td>d</td>
</tr>
<tr>
<td>Distance</td>
<td>d</td>
</tr>
<tr>
<td>Subdistance</td>
<td>d</td>
</tr>
<tr>
<td>Angle</td>
<td>d</td>
</tr>
<tr>
<td>Pressure</td>
<td>d</td>
</tr>
<tr>
<td>Temperature</td>
<td>d</td>
</tr>
<tr>
<td>String</td>
<td>s</td>
</tr>
</tbody>
</table>

Note that all types which represent floating point numbers are tagged by 'd'. This is because operations valid for the type Double are also valid for the other d-tagged types.

If there are several similar object names, a qualifier may follow the name and further clarify it. For example if we kept two special point numbers, one for the first point and one for the last, the variable names would be the (qualified) variables iPointNoFirst and iPointNoLast.

Structure types do not have a default prefix, if needed the (abbreviated) type name could be used. For arrays the base name itself could contain the information that the variable names an array.

For global variables an additional prefix 'g' might be useful.

### 8.3.2 Constants and user-defined types

Constants begin with an upper case character. If constants contain only upper case characters (as most of the predefined constants do) the underscore '_' is used to separate parts of the name. Often constants can be grouped together, then a prefix is used to denote their common criterion. For example the return codes use RC, as in RC_OK, RC_ABORT, etc.
Mostly constants are globally defined. For *local constants* an additional prefix 'loc' might be useful.

**User defined types** begin with an upper case character. Use the postfix '_TYPE', '_Type' or 'Type' (according to the naming convention used for the type name itself) appended to the type name to denote that it is a type structure. Alternatively, you can use a prefix 'T'. (For types these conventions are useful since GeoBASIC is not case sensitive. Hence, for example, if there is a type Date no variable can be named date. If the type has the name TDate or Date_Type or DateType, there can.) As for local constants, *local types* might be prefixed with 'loc'.

### 8.3.3 Procedures

A procedure name begins with an upper case letter and succinctly describes the action that is performed. Variables that denote parameters passed to a function or subroutine (in the parentheses after the function/subroutine name) should be well documented, also indicating whether they act as *input*, *output*, or *input and output* parameters.

### 8.3.4 Keywords

GeoBASIC keywords are all in upper case letters. For example, DIM, FOR, LOOP, FUNCTION, etc.

### 8.3.5 Labels

For error labels (ON ERROR GOTO) we use the function/subprocedure name with the qualifier '_Err' appended.
SUB LabelExample ()
'code of the procedure
LabelExample_Err:
SELECT CASE ERR
'handle specific errors here
CASE ELSE
'generic error handler here
END SELECT
END LabelExample

8.3.6 Remark on naming conventions

Naming conventions never replace the judicious use of comments in your GeoBASIC program code. Naming conventions are an extension of, not a replacement for, good program-commenting techniques.

Formulating, learning, and applying a consistent naming style require a significant initial investment of time and energy. However, you will be amply rewarded when you return to your application a year later to do maintenance or when you share your code with others. Once you implement standardised names, you will quickly grow to appreciate the initial effort you made.

To complete the discussion about naming conventions, we mention the use of program headers:

In every function/subprocedure there should be a header describing, at a minimum, purpose, and parameters passed and/or returned. (In addition there might be comments, the author's name, last revision date, notes, etc.)
9 Refined GeoBASIC Concepts

In GeoBASIC several concepts are implemented to utilise and standardise programming and applications.

9.1 Units

Working with units always gives rise to the problem that different users want to work with different units. In geodesy, take the vertical angle as an example: some surveyors measure in Gon, some in radians, others in percentages. And, in addition to the unit problem, there is the question where to fix the zero point of some scale. Again for the vertical angle example: some surveyors want to have zenith angles, some nadirs, some something in between.

To cope with this situation there is a fine automatic unit handling system built in the theodolite system, and the GeoBASIC programmer can take full advantage of it. All that has to be done in a GeoBASIC program, is to keep all values in SI units and, when a value has to be displayed specify what kind of value it is: a horizontal angle, a vertical angle, a distance, a temperature, etc. All the formatting, together with choice of the right representation (the user may define this in his theodolite system configuration with which the GeoBASIC programmer is not concerned), and displaying the unit after the value are handled automatically. (Of course the programmer can also decide not to use this automation and handle everything on his own. But values obtained from the system will be in SI units anyway.)

9.1.1 What the GeoBASIC programmer has to do

- Use SI units throughout the program. All computations are done with values in SI units.
- When displaying, specify the correct data type i.e. Distance for the value is displayed. See description of the MMI_PrintVal function in the "Reference Manual".

We will give an example of measuring an horizontal angle, computing the difference to a given angle, and displaying the difference on the display. (Note that we use the GetAngleHz routine from the MeanHz program (see 10.1), and we assume that a text dialog has been opened properly. The angle difference is normalised to the range 0 to 2\pi.)
Example

```basic
DIM dHz1 AS Angle 'first horizontal angle
DIM dHz2 AS Angle 'second horizontal angle
DIM lValidHz2 AS Logical 'indicator if second
' angle is valid
DIM dDiffHz AS Angle 'the difference of the
' angles

'assume dHz1 is initialized here to an angle
'in radians
GetAngleHz( dHz2, lValidHz2 )

dDiffHz = dHz1 - dHz2
GM_AdjustAngleFromZeroToTwoPi( dDiffHz )

MMI_PrintVal( 20, 0, 8, 3, dDiffHz, lValidHz2,
MMI_DIM_ON )
```

The output is as follows:

- If the GetAngleHz routine returned a valid angle, also the difference
  dDiffHz will be valid (this is why lValidHz2 is used in the
  MMI_PrintVal function). In this case the angle will be formatted in an 8
  character wide field with 3 decimals, afterwards the unit according the
  theodolite system configuration will be displayed.
  Assume that gon is set and the angle difference was 1.5473452 radians,
  then at position 20 in line 0 the output will be « 98,507 g».

- If the angle returned from GetAngleHz was not valid, five dashes will be
  displayed « ----- g».

9.1.2 What the user/surveyor has to do

The user has to set up the units, in which he want to work, in the theodolite system
configuration. All outputs that use the theodolite system will automatically be
formatted according to this setting.

9.2 THE USER MEASUREMENT DIALOG

The User Measurement Dialog (sometimes referred as MDlg) standardises the
visualisation of the measurement values in GeoBASIC. Each value (i.e. vertical
angle, horizontal distance) has a predefined output format. Thus the GeoBASIC
programmer has only to define, on which line a value should be displayed. All lines begin with a brief description of the value.

For example (Output of the horizontal distance):

```
«Horiz.Dist: 158.287 m»
```

Additionally the measurement parameters and (self-definable) application parameters can be displayed in the measurement dialog. Thus a user is able to change measurement parameters immediately and without leaving the dialog. All measurement values and measurement parameters are saved in the theodolite’s data pool as system parameters.

We distinguish between measurement and application parameters. The former are defined by the system in its meaning and data type. The latter can be defined freely by the user. Please refer to Appendix H in the reference manual for a list of all system and application parameters, which can be used in a measurement dialog.

### 9.2.1 Configuration of the User Measurement Dialog

Before using the measurement dialog we have to define its contents. There are 3 types of possible entries:

- **System parameters:**
  The routine `GSI_SetLineMDlg` places a system parameter (measurement value or measurement settings) on a line.

- **Pure text line:**
  The routine `GSI_SetLineMDlgText` places any text on a line.

- **Application parameters:**
  The routine `GSI_SetLineMDlgPar` places a (self-definable) application parameter on a line.

**Note** The user measurement dialog configuration is automatically initialised with the entries of the first system measurement dialog.

Thus all lines which are not configured by the GeoBASIC programmer shows the same parameters as the first system measurement dialog. For further explanations how to configure the user measurement dialog read the description of the 3 system functions (`GSI_SetLineMDlg`, `GSI_SetLineMDlgText`, `GSI_SetLineMDlgPar`) in the reference manual.
9.2.2 Creating the User Measurement Dialog

After the definition of the content GSI_CreateMDlg analogous to the creation of a text dialog creates the user measurement dialog. For adding buttons to the dialog use MMI_AddButton.

9.2.3 Executing the User Measurement Dialog

In the following example a measurement dialog is created with the horizontal angle on line 2 and the buttons “DIST” on F2-key and “QUIT” on SHIFT-F6-key. All other lines are predefined by the system. After the creation of the dialog the measured values will be updated in a loop:

```plaintext
'Change line 2
GSI_SetLineMDlg(2, GSI_PAR_AngleHz)
GSI_CreateMDlg(2, "MEAS", "Measurement Test",
               "Measurement Help...")

'Addition of buttons
MMI_AddButton(MMI_F2_KEY, "DIST")
MMI_AddButton(MMI_SHF6_KEY, "QUIT")
lDone = FALSE
DO WHILE NOT lDone
    GSI_UpdateMeasurement(TMC_AUTO_INC, WAITTIME,
                          lRecValid, iCode, FALSE)
    GSI_UpdateMDlg(iButton)
    SELECT CASE iButton
        CASE MMI_F2_KEY
            'DIST Button --> meas a distance and angles
            BAP_MeasDistAngle(iDistMode, dHz, dV, dDist, TRUE,
                               MEAS)
        CASE '..
            'handle other keys
        CASE MMI_ESC_KEY, MMI_SHF6_KEY
            'done --> exit this routine
            lDone = TRUE
    END SELECT
END DO WHILE
```

The routine GSI_UpdateMeasurement updates the measurement values in the theodolite data pool. GSI_UpdateMDlg updates the user measurement dialog with the new values and returns the pressed button. For further explanations read the description of these system routines in the reference manual.
If the user measurement dialog is not used any more it must be deleted with \texttt{MMI\_DeleteDialog}.

See the example program \texttt{MEAS.GBS} for a typical usage of the user measurement dialog.

### 9.2.4 Mixing the User Measurement Dialog with Other Dialogs

There can be only one user measurement dialog at a time, i.e. an existing user measurement dialog must be deleted with \texttt{MMI\_DeleteDialog} before a new one can be created with \texttt{GSI\_CreateMDlg}. If a user measurement dialog is active, no text dialog can be opened and vice versa.

But a graphics dialog may be opened while a user measurement dialog is active.

\begin{tabular}{|l|l|}
\hline
\textbf{Note} & The reverse is not the case: a user measurement dialog may not be opened while a graphics dialog is open. If a user measurement dialog and a graphics dialog are open, the graphics dialog has priority, i.e. all future function calls are related to the graphics dialog until it is closed. \\
\hline
\end{tabular}

### 9.3 TPS1100 CONFIGURABILITY

In general, each part of an application, which should be accessible from outside, has to be of the form `GLOBAL SUB`. These points are known as entry points and can be used in two ways. First they can be linked to a menu item (of the a system), and second they can be described as configuration item.

#### 9.3.1 Adding the program in a System Menu

The easier way to access an entry point of an application is to link it to a menu item during the installation phase. Please refer to the Reference Manual \texttt{MMI\_CreateMenuItem} for further explanations.
9.3.2 Import the program in a User Configuration

The TPS1100 series theodolites support the concept of individual configurations. In a configuration the user can define his own dialogs or menus and link them to certain events (i.e. pressing the PROG key or Power ON). If the event occurs then the linked dialog or the menu will be displayed. The user can create and change his configuration on the PC with the Customisation Tool.

The import of a GeoBASIC program in a user configuration means, that an external GeoBASIC routine is linked with an item of a user defined menu, a button of a user defined dialog or directly with an event. If either the event occurs or the button is pressed or the menu item is selected, then the linked external routine is executed. For the import of a GeoBASIC program the Customisation Tool needs a special file named APPInfo-file with the necessary information about the program.

The usage of the APPInfo-file in the Customisation Tool:

- Start the Customisation Tool
- Open a configuration file, appropriate text- and definition files
- Choose Import Application from the file menu
- Check the box named with the program name (i.e. AppInfoExample)
- Press the OK button

Now the globally accessible subroutines may be added to menus, buttons, etc. simply by using drag and drop.

**Generate the AppInfo-file**

The AppInfo-file is automatically generated during compilation, if there is a application information (short AppInfo) section in the GeoBASIC source file.

| Note | The AppInfo-section has to occur at the end of the source code. The AppInfo-section is optional; if there is no AppInfo-section in the GeoBASIC source file, the AppInfo-file generation is omitted. The global routine "Install" is optional, since any global routine may be associated with a menu entry, using the AppInfo-file via the Customisation Tool. |

The following GeoBASIC sample code illustrates the usage of the AppInfo-section in a GeoBASIC source file. See also the sample program AppInfoTest.gbs.
PROGRAM AppInfoExample

'-------------------------------------------------------
GLOBAL SUB GlobalSub1
  Dim dummy As Integer
  MMI_WriteMsgStr("AppInfoExample.", "GlobalSub1 in AppInfoExample called", MMI_MB_OK, dummy)
END GlobalSub1

'-------------------------------------------------------
GLOBAL SUB GlobalSub2
  Dim dummy As Integer
  MMI_WriteMsgStr("AppInfoExample.", "GlobalSub2 in AppInfoExample called", MMI_MB_OK, dummy)
END GlobalSub2

END AppInfoExample

<Application Information for Config Tool

APPINFO

GENERAL
  SET Author "Leica AG, CH - Heerbrugg"
  SET Desc "AppInfo Example Application"
  SET TheoModel "TCA1100"
END GENERAL

ENTRYPOINT GlobalSub1
  SET CapLg "Global Sub 1"
  SET CapSh "GSUB1"
  SET Desc "test of appinfo subroutine 1"
END GlobalSub1

ENTRYPOINT GlobalSub2
  SET CapLg "Global Sub 2"
  SET CapSh "GSUB2"
  SET Help "displays a message and exits"
END GlobalSub2

END APPINFO

The global subroutines GlobalSub1 and GlobalSub2 are indicated as entry points for the import in a user configuration. Refer to Chapter 2.11 in the Reference Manual for a description of the syntax in BNF-form.
The following figure depicts the whole scenario, from the generation of the AppInfo file over the import in a user (definable) configuration to the loading of the configuration into the theodolite:

---

**9.4 INTERAPPLICATION-CALL**

The inter-application-call makes it possible to call a subroutine in another GeoBASIC program. With this concept the GeoBASIC programmer can use the same subroutine in several programs.
9.4.1 Definition of a subroutine for Interapplication-Call

If a subroutine should be called by another application, it must be defined as a global subroutine.

Example:

```
PROGRAM IAC2
GLOBAL SUB InterAppEntry
  DIM iButton AS INTEGER
  MMI_WriteMsgStr("Welcome in IAC2","IAC2", MMI_MB_OK, iButton)
END InterAppEntry
END IAC2
```

9.4.2 Call the global subroutine

Before calling the global subroutine, the GeoBASIC programmer has to check with `CSV_LibCallAvailable` if the subroutine is available. That usually means if it is loaded or not. If the subroutine available, he can invoke it with `CSV_LibCall`.

Example:

```
DIM lAvailable AS LOGICAL
  'Check if global subroutine is available
  CSV_LibCallAvailable("IAC2","InterAppEntry", lAvailable)
  IF lAvailable
    'available, call global subroutine
    CSV_LibCall("IAC2", "InterAppEntry", "BASIC")
  END IF
```

See the example program IAC.GBS and IAC2.GBS for a typical usage of inter-application-call. For further explanations read the description of `CSV_LibCall` and `CSV_LibCallAvailable` in the reference manual.

9.5 SYSTEM FUNCTION CALL

If a theodolite user creates his own configuration on the PC with the Customisation Tool, he has a wide selection of predefined system functions which he can add to menus, buttons, etc. After the loading of the configuration he calls the system functions by selecting the appropriate menu item or button.

The GeoBASIC programmer has the same possibilities. With the routine `CSV_SysCall` he can call the system functions in his programs. Because some system functions do not run on every theodolite type, there is a routine
CSV_SysCallAvailable, which returns if the system function can be executed.

Example:

```geo
DIM lAvailable AS Logical
CSV_SysCallAvailable(CSV_SFNC_PositCompassDlg, lAvailable)
IF lAvailable
    CSV_SysCall(CSV_SFNC_PositCompassDlg)
END IF
```

If the system function CSV_SFNC_PositCompassDlg can be executed (RCS mode is active), then the dialog RCS orientation with a compass is displayed. For further explanations read the function descriptions of CSV_SysCall and CSV_SysCallAvailable in the reference manual. In Appendix H of the reference manual there is a list of all system functions.

### 9.6 SYSTEM EVENT GENERATION

Every configuration for a TPS1100 series theodolite is event driven. The user or the system itself generates an event (e.g. the user has pressed the PROG key or the initialisation sequence is finished) and the configuration functionality executes then the linked action (menu, dialog, macro, application or system function).

A GeoBASIC program can generate all events, which can occur in the theodolite system software, also. To generate a system event the same functions can be used as for calling system functions. The routine CSV_SysCall is used for the generation of system events. The routine CSV_SysCallAvailable returns TRUE, if there is an action linked to the requested event and the action can be executed.

Example:

```geo
DIM lItemDefined AS Logical
CSV_SysCallAvailable(CSV_EFNC_CompensatorSetting, lItemDefined)
IF lItemDefined
    CSV_SysCall(CSV_EFNC_CompensatorSetting)
END IF
```

If a configuration item is defined for the system event CSV_EFNC_CompensatorSetting (compensator setting event; usually connected to a compensator setting dialog) CSV_EFNC_CompensatorSetting is generated and the appropriate system function, application, macro, dialog or menu is
executed. For further explanations read the function description of CSV_SysCall and CSV_SysCallAvailable in the reference manual. In Appendix H of the reference manual there is a list with all system events.
10 GEOBASIC SAMPLE PROGRAMS

10.1 MEANHZ — MEAN VALUE OF HORIZONTAL ANGLE MEASUREMENTS

10.1.1 Program description

The program "MeanHz" measures a number of horizontal angles and computes its arithmetic mean value. The measured angles and the mean angle can then be displayed graphically.

Program flow:

First, the user may enter the number of horizontal angles he wants to measure. (The number of angles must be within a certain range.) Then the angles are measured — each time the REC key is pressed the current horizontal angle is recorded.

As soon as the requested number of angles is measured, the mean angle is computed and displayed. Now the user has the choice either to display the angles graphically, to move the theodolite to the computed mean angle or to quit the program. (The program can be terminated with the ESC button or the QUIT button on shift-F6 at any time.)
Input the number of angles to measure

Measure the angles

Compute and display mean angle

Show Graphics?
  Yes → Draw the angles
  No → Move theo to mean angle?
  Yes → Position theodolite on mean angle
  No → End
10.1.2 Source code listing

See example file "meanhz.gbs"

PROGRAM Mean
  '' Sample application for building the mean value of angles
  ''-------------------------------------------------------------
  '' Measures a user defined number of horizontal angles and calculate
  '' the mean angle. The measured and the mean angle can also be
  '' displayed graphically.
  '' GeoBASIC 1.0 for TPS1100 Series Instruments
  '' (c) Leica AG, CH - Heerbrugg 1998
  ''-------------------------------------------------------------
  '' Global Declarations
  CONST MaxNoHz = 9        'Maximum number of angles that can be
                           'measured
  CONST CaptionShort = "MEAN" 'Short caption (displayed lefthand, in
                               'top line)

  'Type to store the angles (for graphics)
  TYPE DIM
    TAangles (MaxNoHz) AS Angle
  END

  DIM fId AS FileId       'File identification

------
GLOBAL SUB Install
  '------
  ' Description
  ' Adds the program into the theodolite's PROG menu. The program's
  ' (application's) name is 'Mean', the global routine to start is
  ' 'Main' and the program menu item will be named 'MEAN HZ'.
  MMI_CreateMenuItem( "Mean", "Main", MMI_MENU_PROGMENU, "MEAN HZ")
END Install

SUB RecordValue (dHz As Angle, ByVal dMean As Angle)
  '------
  ' Description
  ' Writes the value to data link and file.
  ' DIM sVal1 AS String30

TPS1100-Version 2.10
DIM sVal2 As String30
DIM sOut As String255

ON Error Resume Next 'Ignore all errors

MMI_FormatVal(MMI_FORMAT_HZANGLE, 10, 2, dHz, TRUE, MMI_DEFAULT_MODE, sVal1)
MMI_FormatVal(MMI_FORMAT_HZANGLE, 10, 2, dMean, TRUE, MMI_DEFAULT_MODE, sVal2)

sOut = "hz: " + sVal1 + "mean: " + sVal2 'Compute output text

'Write to data link and file
Send(sOut)
Print(fId, sOut)

END RecordValue

'--------------------------------------------------------------------
SUB GetAngleHz ( dHz AS Angle, lValid AS Logical)' ----------' Description' Measures the horizontal angle 'valid' indicates if the dHz is ' valid.' Parameters' OUT: dHzOUT, lValid'DIM theoAngle AS TMC_Angle_Type 'The measured valuesDIM iInfo AS Integer 'Return code

ON Error Resume Next 'Ignore all errors

'TMC_GetAngle( theoAngle, iInfo )

IF (Err = RC_OK) THEN
lValid = TRUE
 dHz = theoAngle.dHz
ELSE
lValid = FALSE
END IF

END GetAngleHz

'--------------------------------------------------------------------
SUB ShowGraphics( byVal iNoPoints AS Integer, angles AS TAngles, 'byVal dMean AS Angle )

' Description
' Displays the measured and the mean horizontal angles graphically.' Parameters
' IN: iNoPoints, angles, dMean

DIM ix AS Integer 'x coordinate
DIM iY AS Integer 'y coordinate
DIM iButton AS Integer 'button id

CONST CX = 90 'display center x coordinate
CONST CY = 24 'display center y coordinate
CONST DL = 20 'length of line
CONST HELPTEXT = "Visualizes the angles with lines from the station."
  "The computed mean angle is shown by the longer line."
  "The north angle is 0."

MMI_CreateGraphDialog( CaptionShort, "PICTURE", HELPTEXT )

'Draw center and circle
MMI_DrawCircle( CX, CY, 3, 3, MMI_NO_BRUSH, MMI_PEN_BLACK )
MMI_DrawCircle( CX, CY, DL, DL, MMI_NO_BRUSH, MMI_PEN_BLACK )

'Draw lines for angles (there are iNoPoints angles)
DO WHILE iNoPoints > 0
  'compute the line
  iX = INT( DL * SIN(angles(INT(iNoPoints))) )
  iY = INT( DL * COS(angles(INT(iNoPoints))) )
  MMI_DrawLine( CX, CY, CX+iX, CY-iY, MMI_PEN_BLACK )
  iNoPoints = iNoPoints - 1
LOOP

'Draw line for dMean
iX = INT( (DL+4) * SIN(dMean) )
iY = INT( (DL+4) * COS(dMean) )
MMI_DrawLine( CX, CY, CX+iX, CY-iY, MMI_PEN_DASHED )

'Wait for key press and finish dialog
MMI_AddButton( MMI_F5_KEY, "END" )
MMI_GetButton( iButton, FALSE )
MMI_DeleteDialog()

END ShowGraphics

GLOBAL SUB Main

' Description
' Reads the number of points to be measured. Measures these points,
' calculates the mean value and shows the result or moves (if
' motorized) the TPS to calculated position.

DIM iNoPoints AS Integer 'number of points to measure
DIM iCurrNo AS Integer 'current point number
DIM lNoOk AS Logical 'TRUE if no of points are valid
DIM lHzOk AS Logical 'TRUE if measured hz is valid
DIM dHz AS Angle 'measured hz
DIM storeHz AS TAngles 'array of measured angles
DIM dMean AS Angle 'calculated mean angle
DIM iKeyPressed AS Logical 'TRUE if button pressed
DIM iButton AS Integer 'id of pressed button
DIM Family AS TPS_Fam_Type 'this data structure is used to store information about the system

ON Error Resume Next 'ignore errors

'check which type of instrument is active and open file
CSV_GetInstrumentFamily( Family )
IF ( Family.lSimulator ) THEN
  Open( "C:\results.txt", "Append", fId, 0 )
ELSE
  Open( "A:\results.txt", "Append", fId, 0 )
END IF

'set up dialog and input iNoPoints
MMI_CreateTextDialog ( 6, "MEAN", "HZ MEAN VALUE", "Compute mean HZ for a number of measurements." )

' ******************************************
' * read in iNoPoints *
' ******************************************
iNoPoints = 3
lNoOk = TRUE
MMI_PrintStr( 0, 0, "No of points: ", TRUE )
MMI_AddButton( MMI_F1_KEY, "CONT" )
MMI_AddButton( MMI_SHF6_KEY, "QUIT" )
MMI_InputInt( 26, 0, 2, 1, MaxNoHz, MMI_DEFAULT_MODE, iNoPoints, lNoOk, iButton )

'setup rest of dialog
iCurrNo = 1
MMI_PrintStr( 0, 1, "Curr. point: ", TRUE )
MMI_PrintVal( 26, 1, 2, 0, iCurrNo, lNoOk, MMI_DEFAULT_MODE )
MMI_PrintStr( 0, 2, "HZ: ", TRUE )
MMI_AddButton( MMI_F3_KEY, "REC" )

'init mean value
dMean = 0.0

'get iNoPoints points (abort if ESC or QUIT is pressed)
DO WHILE (iCurrNo <= iNoPoints) AND (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY)
  MMI_PrintVal( 26, 1, 2, 0, iCurrNo, lNoOk, MMI_DEFAULT_MODE )
  MMI_CheckButton( lKeyPressed )
  dMean = dMean + dHz
  iKeyPressed = NOT iKeyPressed
  iCurrNo = iCurrNo + 1
END DO

MMI_PrintVal( 26, 1, 2, 0, iCurrNo, lNoOk, MMI_DEFAULT_MODE )
MMI_CheckButton( iKeyPressed )
IF lKeyPressed THEN
        MMI_GetButton( iButton, FALSE )
        SELECT CASE iButton
            CASE MMI_F3_KEY, MMI_F1_KEY
                GetAngleHz( dHz, lHzOk )
                storeHz(iCurrNo) = dHz
                dMean = dMean + dHz
                'if REC pressed record values
                IF iButton = MMI_F3_KEY THEN
                    RecordValue(dHz, dMean/iCurrNo)
                END IF
                iCurrNo = iCurrNo + 1
            END SELECT
        ELSE
            'update display
            GetAngleHz( dHz, lHzOk )
            MMI_PrintVal( 20, 2, 8, 3, dHz, lHzOk, MMI_DEFAULT_MODE )
        END IF
    END LOOP

'**************************
* show results *
'**************************

'if execution should procede
IF (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY) THEN
    'setup new buttons
    MMI_DeleteButton( MMI_F1_KEY )
    MMI_DeleteButton( MMI_F3_KEY )
    MMI_AddButton( MMI_F3_KEY, "SHOW" )
    MMI_AddButton( MMI_F4_KEY, "EXIT" )
    MMI_AddButton( MMI_F5_KEY, "GOTOM" )

    'compute mean value
    dMean = dMean / iNoPoints
    MMI_PrintStr( 0, 3, "Mean Hz : ", TRUE )
    MMI_PrintVal( 20, 3, 8, 3, dMean, TRUE, MMI_DEFAULT_MODE )

    DO WHILE (iButton <> MMI_ESC_KEY) AND (iButton <> MMI_SHF6_KEY)
        AND (iButton <> MMI_F4_KEY)
            MMI_GetButton( iButton, FALSE )
        END IF
SELECT CASE iButton
    CASE MMI_F3_KEY
        ShowGraphics( iNoPoints, storeHz, dMean )
        'move theo to the computed mean horizontal angle
        CASE MMI_F5_KEY
            BAP_PosTelescope(BAP_POSIT_HZ, BAP_POS_MSG, dMean, 0, 0.1, 0.1)
    END SELECT
LOOP
END IF
'clean up text dialog
MMI_DeleteDialog()
'close output file
Close(fId)
END Main

10.2 SAMPLE PROGRAMS

These code samples gives you some help for building your first applications. Each of them should give you some hints in a specific problem domain.

- **appinfotest.gbs** This example shows the use of the application information section in the GeoBASIC source file.
- **codefunc.gbs** An example of a program which will be called, when the Code-key has been pressed.
- **cursor.gbs** Cursor control in a dialog.
- **error_ha.gbs** This program shows how error handling changes execution of a program.
- **language.gbs** Take this program as an example to support multiple language applications. Two language files and its text databases are provided to see how multilingual support works.
- **meanzhz.gbs** This sample shows the calculation of the mean value of horizontal angle measurements, see Chapter 10.1.
<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>meas.gbs</td>
<td>A simple example how to measure with BAP-functions, including Quick-Coding</td>
</tr>
<tr>
<td>meas_od.gbs</td>
<td>A simple example how to measure and how to record data in an own data-format, including Quick-Coding</td>
</tr>
<tr>
<td>stringer.gbs</td>
<td>This example shows in which situations typical errors may occur.</td>
</tr>
<tr>
<td>test.gbs</td>
<td>An empty frame for building up a GeoBASIC application.</td>
</tr>
<tr>
<td>tracking.gbs</td>
<td>This program shows possible techniques to take advantage of the measurement facilities.</td>
</tr>
<tr>
<td>menu.gbs</td>
<td>A simple menu handler.</td>
</tr>
<tr>
<td>dirlist.gbs</td>
<td>This example shows how to get PC card information and how to read a directories content.</td>
</tr>
<tr>
<td>inclmain.gbs</td>
<td>This example shows the usage of an include file.</td>
</tr>
<tr>
<td>iac.gbs</td>
<td>An example for an interapplication call.</td>
</tr>
</tbody>
</table>
11 PORTING A TPS1000 ORIGINATED PROGRAM

The implementation of the TPS1100 theodolite series includes several new concepts compared to the firmware of TPS1000 theodolites. To follow up these new concepts and to take care of functionality that has been changed or removed in the implementation of TPS1100 firmware, GeoBASIC programs, once developed for TPS1000 hardware, cannot be compiled without changing the source code. In this chapter we will cover this subject and we try to give some guidelines to help the developer to port the source code onto the new platform. During the design phase of GeoBASIC for TPS1100 systems we took certain care to make the migration as smooth as possible. Although all programs’ source code has to be changed, the effort to port it will be for the most applications not that high.

In the very end this means also that the developer has to maintain two source code bases.

11.1 TPS1100 HARDWARE RELATED CHANGES

11.1.1 Display Line Length

The TPS1100 series instruments use a different liquid crystal display. The difference means also that one can use only 29 characters per line. To be “independent” of the display length we defined the string type DisplayLine. It does not contain the string length in the name, hence this should help in future to port applications. To be compatible with older, TPS1000 GeoBASIC programs we did not change all String30 declarations. Of course only 29 characters will be printed out to the display.

11.1.2 Keyboard

The number of keys has been reduced, there is no CONT-Key any longer. Remove all MMI_CONT_KEY appearances in the source code. We deleted the definition of this constant to make it more obvious to the programmer that he has to change the source code and think about any button assignments.
11.2 CHANGES TO THE SIMULATOR

Now TPSSim supports GeoBASIC programs larger than 64 KB. A restriction, which turned out in the past, bothered the most of the GeoBASIC program developers. We would like to point out that the SWTheo extension enables the programmer to influence the execution of a program. With specific dialogs the programmer gets the possibility to set or change certain (measurement) values. We hope this helps a lot to simulate a more realistic TPS environment and makes it almost obsolete to have an instrument at your hand to test your application. Of course, still the final test of an application has to be done on an instrument. See also the documentation of TPSSim for further explanations.

11.3 NEW CONSTRUCTS IN GB_1100

Due to some requests we added a few new constructs to GeoBASIC for TPS1100 instruments.

11.3.1 #include Statement

It is now possible to include a GeoBASIC source file in another one. Nevertheless only one level of inclusion is allowed.

11.3.2 MID$ statement

Mid$’s implementation has been extended. Now Mid$ can be used to assign a character or a substring to another string at a certain position. In this way single characters of a string can be set or replaced.

Examples:

$$T = \text{"abcdef"}$$
$$\text{Mid$(t, 2, 1) = \text{"+"}$ results in $\text{"a+cdef"}$
$$\text{Mid$(t, 4) = \text{"--------"}$ results in $\text{"a+c--------"}$

11.3.3 Application Info

A general concept of configurability has been introduced for the TPS1100 family of instruments. This gives totally new customisation possibilities into the hand of
the developer and more to the customer support. Up to a certain degree GeoBASIC supports this configurability. For example an assignment of a GeoBASIC program to a menu item can be changed by the new configuration utilities. Or it can be assigned to a function key.

To support these new features we extended the concept of the program by a section that describes the attributes of it.

This (informational) section can be appended optionally at the end of the source file. See the extra explanation of it to get further information about it.

11.4 GEOBASIC SOURCE CHANGES

Many GB programs have a similar structure. Therefore it does not surprise that many programs have to be rewritten in the same way to be compilable and executable for TPS1100 GeoBASIC.

11.4.1 General Dialog Changes

The CONT key does not exist any more on the TPS1100 instruments. Scan your source code for MMI_CONT_KEY and replace it by a function key. The TPS1100 guidelines use MMI_F1_KEY normally for the CONT key functionality. This might make it necessary to change your function key layout. Look at the existing dialogs to get an idea and to be more consistent to the built-in dialogs, to which function keys which functionality has been assigned.

In certain circumstances, where no function keys were left, the ESC key was the only way to leave a dialog. Normally ESC leaves a dialog with leaving values untouched.

MMI_SHIFT_ESC_KEY will not be supported any more. Instead one has to assign QUIT to (normally) Shift-F6. Quit leaves the whole application.

**Note**

'Old' versions of constants and functions are left aligned. Newer versions or replacements have been shifted to right. The listed changes are ordered in an assumed importance.
Please notice that GB-TPS1000 supports conceptually 2(3) dialogs at once: a text or a graphics dialog and in parallel a customisable measurement dialog - MDlg.

A typical application may create a text dialog and link a graphics dialog to a menu button. Notice, that both dialogs exist at the same time and distinguish this situation from another, where the text dialog will be deleted before the graphical dialog will be created. In the former case one can go back to the text dialog without recreating it. In the latter the text dialog has to be rebuilt. In GB_TPS1100 text and measurement dialog are mutually exclusive.

See the following scheme for a graphical explanation. "()" denotes a dialog.

<table>
<thead>
<tr>
<th>TPS1000</th>
<th>TPS1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Text) and (MeasDlg)</td>
<td>(Text or MDlg)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>(Graphic)</td>
<td>(Graphic)</td>
</tr>
</tbody>
</table>

Graphic overrides Text and may have its own buttons. The other way around is not possible. At the same time a MeasDlg may be defined.

Graphic overrides Text or MDlg. Text and MDlg are mutually exclusive. Only one can be defined at once. All three dialog types may have their own buttons.
## 11.4.2 Recording Format Settings

### Deleted:
- GSI_GetRecFormat()
- GSI_SetRecFormat()

### Replaced by (extended):
- GSI_GetRecMask ()
- GSI_SetRecMask ()

## 11.4.3 System Dialog Calls

### Replacements for old dialog invocation calls:

<table>
<thead>
<tr>
<th>Old Call</th>
<th>New Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_CommDlg ()</td>
<td>CSV_SysCall (CSV_EFNC_GeoComSetup, Caption)</td>
</tr>
<tr>
<td>GSI_SelectTemplateFiles() and GSI_Setup ()</td>
<td>CSV_SysCall (CSV_EFNC_Setup, Caption)</td>
</tr>
<tr>
<td>GSI_StationData ()</td>
<td>CSV_SysCall (CSV_EFNC_SetStation, Caption)</td>
</tr>
<tr>
<td>GSI_TargetDlg ()</td>
<td>CSV_SysCall (CSV_EFNC_TargetData, Caption)</td>
</tr>
</tbody>
</table>

### Deleted:
- GSI_CreateMeasDlg()
- GSI_DefineMeasDlg()
- GSI_DeleteMeasDlg()
- GSI_GetDialogMask()
- GSI_SetDialogMask()
- GSI_UpdateMeasDlg()

### Replaced by a more general concept
- See the reference manual for GSI_*MDlg-* routines.

### New routines are:
- GSI_SetLineMDlg ()
- GSI_SetLineMDlgPar ()
- GSI_SetLineMDlgText ()
- GSI_GetLineSysMDlg ()
- GSI_CreateMDlg ()
- GSI_UpdateMDlg ()
11.4.4 EDM Mode Changes

Replacement for `EDM_MODE` by the extended `BAP_SetMeasPrg ()`.

<table>
<thead>
<tr>
<th>Deleted EDM modes:</th>
<th>New defined modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EDM_SINGLE_STANDARD</code></td>
<td><code>BAP_RED_TRK_DIST</code></td>
</tr>
<tr>
<td><code>EDM_SINGLE_EXACT</code></td>
<td><code>BAP_SINGLE_REF_STANDARD</code></td>
</tr>
<tr>
<td><code>EDM_SINGLE_FAST</code></td>
<td><code>BAP_SINGLE_REF_FAST</code></td>
</tr>
<tr>
<td><code>EDM_CONT_STANDARD</code></td>
<td><code>BAP_SINGLE_REF_VISIBLE</code></td>
</tr>
<tr>
<td><code>EDM_CONT_EXACT</code></td>
<td><code>BAP_SINGLE_RLESS_VISIBLE</code></td>
</tr>
<tr>
<td><code>EDM_CONT_FAST</code></td>
<td><code>BAP_CONT_REF_STANDARD</code></td>
</tr>
<tr>
<td><code>EDM_UNDEFINED</code></td>
<td><code>BAP_CONT_REF_FAST</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_CONT_RLESS_VISIBLE</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_AVG_REF_STANDARD</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_AVG_REF_VISIBLE</code></td>
</tr>
<tr>
<td></td>
<td><code>BAP_AVG_RLESS_VISIBLE</code></td>
</tr>
</tbody>
</table>

11.4.5 Interface Changes

The following routines got a new interface.

GSI_ImportCoordDlg ()
GSI_ManCoordDlg ()

Refer to the reference manual to get the new interfaces.

11.4.6 Deleted and Added Identifiers and Types:

<table>
<thead>
<tr>
<th>TPS1000</th>
<th>TPS1100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleted:</td>
<td>Deleted:</td>
</tr>
<tr>
<td><code>CSV_MAX_USERS</code></td>
<td><code>CSV_MAX_USERS</code></td>
</tr>
<tr>
<td><code>CSV_ILLEGAL_USERNR</code></td>
<td><code>CSV_ILLEGAL_USERNR</code></td>
</tr>
<tr>
<td><code>RC_CSV_ILLEGAL_USERNR</code></td>
<td><code>RC_CSV_ILLEGAL_USERNR</code></td>
</tr>
<tr>
<td>Deleted</td>
<td>CSV_WITH_REFLECTOR</td>
</tr>
<tr>
<td><code>EDM_COMERR</code></td>
<td>CSV_WITHOUT_REFLECTOR</td>
</tr>
<tr>
<td><code>EDM_NOSIGNAL</code></td>
<td></td>
</tr>
<tr>
<td>EDM_PPM_MM</td>
<td>New:</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>EDM_METER_FEET</td>
<td>MMI_SHIFT_CODE_KEY</td>
</tr>
<tr>
<td>EDM_ERR12</td>
<td>For MMI_SetAngleRelation()</td>
</tr>
<tr>
<td>EDM_DIL99</td>
<td>MMI_HANGLE_CLOCKWISE_SOUTH</td>
</tr>
<tr>
<td></td>
<td>Changed to return code:</td>
</tr>
<tr>
<td></td>
<td>MMI_UNDEF_LANG</td>
</tr>
<tr>
<td></td>
<td>For MDIg routines:</td>
</tr>
<tr>
<td></td>
<td>MMI_FFORMAT_STRING</td>
</tr>
<tr>
<td></td>
<td>New date format:</td>
</tr>
<tr>
<td></td>
<td>MMI_DATE_JP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deleted:</th>
<th>New:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_MENU_EXTRA</td>
<td>MMI_MENU_PROGRAMS</td>
</tr>
<tr>
<td>MMI_MENU_CONFIG</td>
<td>MMI_MENU_PROGMENU</td>
</tr>
<tr>
<td></td>
<td>MMI_MENU_AUTOEXEC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>New GSI_ID values:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_SHZ</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_CD_DSC</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PTCD_DSC</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PV_CD</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PV_PTCD</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_ACT_PTID</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_BACKID</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA0</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA1</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA2</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA3</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA4</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA5</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA6</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_APP_DATA7</td>
<td></td>
</tr>
</tbody>
</table>
New GSI_POINT_TYPE:
GSI_BACKSIGHT
GSI_POINT_CODE

GSI_PAR_* parameters
see GSI system functions.

Deleted:
TPS1100
TPS1700
TPS1800
TPS5000
TPS2003

New:
TPS1102
TPS1103
TPS1105

Old TPS_FAM_Type:
iclass
lEDMBuiltIn
lEDMTypeII
lMotorized
lATR
lEGL
lDBVersion
lDiodeLaser
lLaserPlummet
lSimulator

New TPS_FAM_Type:
iclass
lEDMBuiltIn (always TRUE)
lEDMTypeII (always FALSE)
lEDMTypeIII (always TRUE)
lEDMReflectorless
lMotorized
lATR
lEGL
lLaserPlummet
lAutoCollimation
lSimulator

New:
BAP_PRISM_MINI
Deleted, because there is no equivalent function at the TPS1100 series instruments:
BAP_GetFunctionality (), BAP_SetFunctionality ()
BAP_SetFunctionalityDlg ()
CSV_GetCurrentUser (), CSV_SetCurrentUser ()
CSV_GetDL (), CSV_SetDL ()
CSV_GetUserInstrumentName ()
CSV_SetUserInstrumentName ()
CSV_GetUserName (), CSV_SetUserName ()
GSI_GetStdRecMask ()
GSI_GetStdRecMaskAll ()
GSI_GetStdRecMaskCartesian ()

Replaced by equivalent functions:
GSI_WiDlg ()
GSI_StartDisplay ()
GSI_GetStdDialogMask ()

Enhanced in certain ways. See the extended identifiers and constants above or refer to the reference manual:
WI-values
CSV_GetPrismType (), CSV_SetPrismType ()
CSV_GetInstrumentFamily ()
GetMemoryCardInfo ()
MMI_GetAngleRelation (), MMI_SetAngleRelation ()
MMI_SetDateFormat (), MMI_GetDateFormat ()

New functions see reference manual for further details:
11.4.8 Returncodes

Their definitions have been coupled totally to the definitions of the TPS1100 firmware. Please refer to the Appendix F in the reference manual for a detailed listing.
12 GEOBASIC RELEASES

12.1 CHANGES IN GEOBASIC RELEASE 1.30
The Release 1.30 of GeoBASIC contains several new subroutines. It reflects user requests and improvements in the TPS1100 Series firmware Release 2.0.

Note: This GeoBASIC Release 1.30 needs at least the TPS1100 Series firmware Release 2.0.

The following paragraph shows the changed items. For a detailed explanation, please see the “GeoBASIC Reference Manual”

12.1.1 New functions in Release 1.30

<table>
<thead>
<tr>
<th>Subroutine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SearchPrism</td>
<td>search prism</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>returns if an alternative user task was running (i.e. FNC or PROG was pressed)</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>returns the internal instrument temperature</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>resets the &quot;WasRunning&quot;-flag</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>returns if distance tracking is running</td>
</tr>
<tr>
<td>GSI_ExecQCoding</td>
<td>executes Quick-Coding with/without recording</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>starts a distance measurement after changing the distance mode (new buttons in FNC menu)</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>returns the current measurement display number</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>returns if a valid code-list for Quick-Coding is selected</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>returns the current recording mask</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>returns the recording order measurement-code or code-measurement block</td>
</tr>
<tr>
<td>GSI_GetWiEntryText</td>
<td>Get coding text-data from the Theodolite data pool</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_SelectCode</td>
<td>select a code-list-code, but without recording it (allows the recording in another format)</td>
</tr>
<tr>
<td>GSI_SetMDlgNr</td>
<td>changes the measurement dialog (used i.e. for DISP buttons)</td>
</tr>
<tr>
<td>GSI_SetQCodeMode</td>
<td>enables Quick-Coding</td>
</tr>
<tr>
<td>GSI_SetRecMaskNr</td>
<td>changes the recording mask</td>
</tr>
<tr>
<td>GSI_SetRecOrder</td>
<td>defines the recording order</td>
</tr>
<tr>
<td>MMI_GetVAngleMode</td>
<td>returns if the V-angle is running (even if a valid distance is available)</td>
</tr>
<tr>
<td>MMI_SetVangleMode</td>
<td>defines the V-angle mode</td>
</tr>
<tr>
<td>TMC_GetAtmCorr</td>
<td>Gets the atmosphere part of distance measurement corrections</td>
</tr>
<tr>
<td>TMC_GetGeomProjection</td>
<td>Gets the projection part of distance measurement corrections</td>
</tr>
<tr>
<td>TMC_GetGeomReduction</td>
<td>Gets the reduction to the reference part of distance measurement corrections</td>
</tr>
<tr>
<td>TMC_GetInclineStatus</td>
<td>returns the inclination status (i.e. ready for recording)</td>
</tr>
<tr>
<td>TMC_SetAtmCorr</td>
<td>Sets the atmosphere part of distance measurement corrections</td>
</tr>
<tr>
<td>TMC_SetGeomProjection</td>
<td>Sets the projection part of distance measurement corrections</td>
</tr>
<tr>
<td>TMC_SetGeomReduction</td>
<td>Sets the reduction to the reference part of distance measurement corrections</td>
</tr>
</tbody>
</table>
12.1.2 New constants in Release 1.30

GSI_GET_NEXT
GSI_MAX_DLG_LINES
GSI_MAX_MDLG_MASKS
GSI_MAX_REC_MASKS
GSI_MAX_REC_WI
GSI_MULTI_REC
GSI_NO_FILE_CHANGE
GSI_SEARCH_FROM_END
TPS1101

12.1.3 New datatypes in Release 1.30

HzAngle
VAngle
TMC_GEOM_PROJECTION_Type
TMC_GEOM_REDUCTION_Type
TMC_ATM_TEMPERATURE_Type

12.1.4 New CSV_SysCall constants in Release 1.30

CSV_SFNC_CheckOrientation
CSV_SFNC_CurrentSetPpmDlg
CSV_SFNC_DefSearchAreaDlg
CSV_SFNC_LoadApplDlg
CSV_SFNC_LoadSysLangDlg
CSV_SFNC_SetDefaultSearchRange
CSV_SFNC_ToggleMeasPrgFastRapidTrk
CSV_SFNC_ToggleMeasPrgRefRL
CSV_SFNC_ToggleMeasPrgStdTracking
CSV_SFNC_ToggleSearchArea
CSV_SFNC_ToggleVAngleMode

12.2 CHANGES IN GEOBASIC RELEASE 2.10

The Release 2.10 of GeoBASIC contains the first edition of the integrated development environment GBStudio.

It contains also a few minor bug fixes.
<table>
<thead>
<tr>
<th>Note:</th>
<th>This GeoBASIC Release 2.10 needs at least the <strong>TPS1100 Series firmware Release 2.10</strong> or the TPS1100 Series Simulator 2.10.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>GeoBASIC applications, compiled with GeoBASIC 1.30, are also executable on the <strong>TPS1100 Series firmware Releases 2.10</strong>. For running these applications, the GeoBASIC interpreter 1.30 must be loaded. There is no debugging-support for GBStudio! <strong>Different Releases</strong> of GeoBASIC applications on the same instrument are <strong>not supported</strong>!</td>
</tr>
</tbody>
</table>
1. CONTENT OF REFERENCE MANUAL

1. Content of Reference manual
2. GeoBASIC Constructs
3. TPS1100 system and GeoBASIC
4. Remarks on the Description
5. Standard Functions
6. System Functions
A — GEOBASIC SYNTAX
B — GLOSSARY
C — LIST OF RESERVED WORDS
D — DERIVED MATHEMATICAL FUNCTIONS
E — GEOFONT
F — SYSTEM RETURN CODES
G — GEODESY MATHEMATICAL FORMULAS
H — CSV_SYSCALL CONSTANTS
I — CALLABLE C-APPLICATION FUNCTIONS
J — LIST OF PREDEFINED IDENTIFIERS
2. **GeoBasic Constructs**

2.1 General

2.1.1 Syntax and Notation - BNF

2.1.2 Examples

2.1.3 Declarations and Statements

2.1.4 Comments

2.1.5 Names

2.1.6 Numbers

2.1.7 Strings and Tokens

2.1.8 Logical Values

2.2 Data Types

2.2.1 Simple data types

2.2.2 Composite data types

2.2.3 Declaration of Arrays

2.2.4 Declaration of Structures

2.2.5 Predefined Structured Types

2.3 Data Declarations

2.3.1 Declaration of Constants

2.3.2 Declaration of Variables

2.4 Variables

2.5 Expressions

2.5.1 Type Compatibility

2.6 Statements

2.6.1 Sequential Statements

2.6.2 Selection Statements

2.6.3 Iteration Statements

2.7 Routines

2.7.1 Routine Declaration

2.7.2 Routine Calls
2.8 Error Handling ................................................................. 2-38
2.9 The Program ............................................................... 2-41
2.10 output to the display ...................................................... 2-42
  2.10.1 Write .................................................................. 2-42
2.11 AppInfo-definition ...................................................... 2-43
  2.11.1 Syntax in BNF ...................................................... 2-43
2.12 In-/Output to Files ....................................................... 2-45
  2.12.1 Summarising Lists of Types and Procedures ............. 2-46
  2.12.2 File Operation Data Structures ............................. 2-47
  2.12.3 Open ................................................................ 2-48
  2.12.4 Close ................................................................ 2-51
  2.12.5 Input ................................................................ 2-52
  2.12.6 Print ................................................................ 2-54
  2.12.7 Get – values ....................................................... 2-55
  2.12.8 Put – values ....................................................... 2-58
  2.12.9 Tell ................................................................ 2-60
  2.12.10 Seek ................................................................. 2-61
  2.12.11 Eof() (standard function) .................................... 2-63
  2.12.12 CurDir$ ............................................................. 2-64
  2.12.13 ChDir ............................................................... 2-65
  2.12.14 MkDir .............................................................. 2-66
  2.12.15 RmDir .............................................................. 2-67
  2.12.16 Kill ................................................................ 2-68
  2.12.17 GetMemoryCardInfo ........................................ 2-69
  2.12.18 GetFileStat ....................................................... 2-70
  2.12.19 GetDirectoryList .............................................. 2-71
  2.12.20 FileCopy ......................................................... 2-72
  2.12.21 RenameFile ...................................................... 2-73
  2.12.22 RenameDir ....................................................... 2-74
2.13 COMmunication Functions ........................................... 2-75
  2.13.1 Send ................................................................. 2-75
2.1 GENERAL

2.1.1 Syntax and Notation - BNF

The syntax and semantics of GeoBASIC are based on modern Basic implementations (like Visual Basic from Microsoft). The syntax in this manual is given in BNF - Bachus Naur Normal Form.

BNF knows the following elements to describe a syntax definition:

- **Reserved words, operators and delimiters:**
  They are printed in **BOLD** letters and enclosed in double quotes " ". They have to be written as given (except that upper and lower case letters are equivalent).

- **Square brackets [ ]:**
  They designate an **optional** part, hence such a part may be omitted.

- **Curly braces { }:**
  Enclose elements which may occur 0 or more times.

- **Round parentheses ( ):**
  They contain a list of **alternatives** separated by a vertical bar |, from which one has to be chosen.

- **The abstraction character ::=:**
  This sign binds a concrete structure of syntactical elements to an abstract concept of it.

For example see the following syntax description:
This syntax describes all possible variants of variable declarations. It contains reserved words ("STRING"), delimiters ("("), alternative and optional parts. Examples of concrete sentences are:

```
DIM i AS Integer
DIM a(10) AS Double
DIM s AS String*10
```

Reserved words in the text are written in **BOLD** letters, but without quotes. References to GeoBASIC code are written in **Courier**.

### 2.1.2 Examples

In some examples, definitions made in preceding examples, are used. Variable declarations are used before they are introduced formally, details can be found in Section 2.3.2 on Declaration of Variables.

### 2.1.3 Declarations and Statements

Declarations and statements are normally terminated by "end-of-line" (carriage return) or by a comment (see next Section 2.1.4); nevertheless, long declarations and statements may be spread over several lines. Type (structure) and routine declarations and structured statements will always occupy several lines. A single line may never contain more than one declaration or statement.

### 2.1.4 Comments

Comments may be added at the end of a statement line. A comment is introduced by an apostrophe (’), and all characters to the right of it up to the end of the line are ignored by the compiler. The comment is terminated by the end of the line; for
longer comments, simply use another apostrophe on the next line. Comments may stand by themselves on a line.

**Examples:**

- Comments may take the whole line.
  ```plaintext
  'This is a comment line.
  'The comment may continue on the next line.
  ```

- Typically comments give more meaning to the program code. (The exact meaning of the GeoBASIC code is not of importance here, you will learn about it later in this manual.)
  ```plaintext
  'declare variables
  DIM iFirstPoint AS Integer 'the number of the first point
  DIM lButtonPressed AS Logical 'indicates whether a button was pressed

  'initialize the variables
  iFirstPoint = 1 'the first point has the number 1
  ```

- Comments may give additional information and structure the program code.
  ```plaintext
  '----------------------------------------------------
  'this comment says that this is the last example
  'for comments
  '-----------------------------------------------
  ```

**Note** Comments should explain what is going on in the program without having to work through the program code. They are intended for humans trying to understand the program.
2.1.5 Names

Names (identifiers) may be up to 40 characters long. They must begin with a letter and may contain letters, digits, the $-sign, and the underscore character ( _ ). Upper and lower case letters are not distinguished. The reserved words cannot be used as names (see Appendix C for the list of reserved words and Appendix E for predefined identifiers). All user-defined names must be declared before they are used in a program.

The scope of names follows the usual rules for block structured languages, i.e. all names declared at the program level are known and unable from the point of their declaration, unless an object is hidden by a locally defined object of the same name. Names declared at the local (subroutine or function) level are known and unable inside the subroutine or function only, from the point of their declaration through the end of the routine.

In general global objects with the same name as local objects are hidden by the local objects and not visible within the local scope. Despite this rule variable and constant names may not get the same name as global type names.

Field names within structures are local to the structure and can be accessed only through the name of the structure variable; thus, for field names there can never be a name conflict with either globally or locally declared objects, or indeed with field names of other structures.

In the following syntax definitions, all terms containing "Name", such as VariableName, TypeName, etc. signify a name according to this definition.

Note In certain cases the length of names should be no longer than 18 characters. E.g. for using MMI_CreateMenuItem the programmer has to provide a global program name (the application name) and a subroutine name.

If you plan to use the program with other languages than the default language, then you have to use a tool to edit and translate the tokens which are used in the program. This tool supports only names up to 18 characters for the application name. Hence the application name and global subroutine names have been limited to 18 characters.
2.1.6 Numbers

Numeric constants are written in the usual way, i.e.
1. **integers** consist of digits only, and
2. **floating point numbers** of any type contain a decimal point and/or an exponent part (so-called scientific notation or E-format). The exponent part consists of the letter 'E' or 'e' followed by a – possibly signed – integer value.

**Examples:**

- **Integer**
  - Integer | meaning
  - 0 | 0
  - 4711 | 4711
  - 49882 | 49882
  - 0001 | 1

- **Floating point**
  - Floating point | meaning
  - 0.0 | 0.0
  - 3.141593 | 3.141593
  - .25 | 0.25
  - 6. | 6.0

- **Floating point (E-format)**
  - Floating point (E) | meaning
  - 6E3 | 6000.0
  - 7.2e-5 | 0.000072
  - .62e+3 | 620.0
  - 3.E2 | 300.0

**Note** Numbers without a comma are of type Integer, numbers with a comma or E in it are of a floating point type. Hence 0 and 0.0 are of different types.
Numbers which may get only positive values are not supported in GeoBasic. Hence distance variables may get negative values also. The programmer has to take care of that.
2.1.7 Strings and Tokens

Strings (of characters) may be 0 to 255 characters long and are enclosed in a pair of double quotes (" "). Any printable character may be included; lower and upper case letters are distinguished. If a double quote is to be part of the string, it must be written twice. The character-set is described in Appendix E.

Special characters are supported by the notation \d255 which represents one character that has the decimal value composed by the three digits. The special character \d000 is not part of the supported character set, because it’s internal use is to terminate the string. Only decimal values of characters between 1 and 255 are supported.

Due to the notation of special characters a \ has to be written as \\.

Examples:

♦ The smallest string is the empty string. Then follow one character strings.

""  "the empty string"
" "  "a string containing one blank"
"a"  "a string containing the character a"

♦ Normally, strings are somewhat larger.

"This is a string."  "a string with 17 characters"

♦ Strings can contain special characters.

"Slope distance: \d001"  "a string with a special character"

♦ Strings can also contain quotes.

"The states are \"0\" and \"1\""  "a string containing double quotes"

♦ The last example prints as «The states are "0" and "1"».

Token

The TPS-1100 series system software implements a special facility to support different natural languages for the user interface. This feature is based on token processing. With GeoBASIC we can simulate this by passing tokens to system software routines. In the documentation parameters of this type are denoted by the
data type _Token. Actual values of such parameters must be of type string literal or string constant.

**Note** Neither variables nor string expressions are allowed as actual values for parameters of type _Token.

**Examples:**

♦ A typical example would be to create a dialog with graphical output capabilities.

```geo
'a string constant
CONST Help_Token = "This function defines " +
    "the standard " +
    "graph dialog."

MMI_CreateGraphDialog ("GRAPH",
    "Graphical Sit.",
    Help_Token)
```

♦ Variables and string expressions are not allowed as actual parameters. Therefore the following example is multiple erroneous in the call of CreateGraphDialog, because there are tokenizable strings allowed only.

```geo
DIM Help_Token AS String255
DIM capt AS String20

capt = "GRAPH"
HelpToken = "This function defines the "
    "standard graph dialog." 'a string
    'constant
MMI_CreateGraphDialog(capt, "Graphical"+-" sit.",
    Help_Token) 'error!!!
```

### 2.1.8 Logical Values

Logical values are written as TRUE or FALSE. They are predefined names (not reserved words) and can be used wherever logical constants are allowed. As usual for names, upper and lower case letters are not distinguished.
2.2 DATA TYPES
There are two kinds of data types in GeoBASIC: simple and composite.

2.2.1 Simple data types
The simple data types are:
1. Integer
2. Logical
3. Double, Distance, Subdistance, Angle, VAngle, HzAngle, Pressure, Temperature

- The values of type Integer are the signed 31-bit integer numbers, from -2147483648 to 2147483647.
- Variables of type Logical can take on the values TRUE and FALSE. They are used in logical expressions, they can be assigned, and they can be passed as parameters.
- The other predefined simple types are all the same as Double; their values are the floating point numbers. The different names are provided for correct displaying of its units and dimension. Within the theodolite Firmware SI units are used (Meter, radians, hPa and Celsius).

2.2.2 Composite data types
In addition to the predefined (simple) types, there are three composite data types available:
1. String
2. Array
3. Structure

A variable of type String can contain a string of some maximum length which is specified in the declaration of the variable (see Section 2.3.2 on Declaration of Variables). The values of type String are described in Section 2.1.7 on Strings.
2.2.3 Declaration of Arrays

An array consists of a fixed number of values of the same type, organised in one or more dimensions (vector, matrix, three-dimensional array, etc.) and is declared as follows.

Syntax:

```
ArrayDeclaration ::= "TYPE" "DIM" Name SubscriptList "AS"
                   "END" [ Name ]
DataType ::= ( DataTypeName | "STRING" "*" Length )
SubscriptList ::= ( UpperBound { "," UpperBound } )
UpperBound ::= IntegerConstant
Length ::= IntegerConstant
```

- A variable of type "Name" will consist of an array of as many dimensions as there are bounds specified. The upper bounds must be positive integer constants.
- Subscripting starts at 1; thus each dimension has "UpperBound" entries. Each element of the array will be of the data type specified.
- An individual element is accessed by giving its subscripts (coordinates) as expressions (see Section 2.4 on Variables).
- For assignment and parameter passing, the variable may also be used as a whole. Other operations can only be performed on the individual elements; in particular, comparison of entire arrays is not possible.

Examples:

- Declare a type for an array that contains two integers, and a variable of that type.
  ```
  TYPE DIM MyFirstArrayType(2) AS Integer END
  DIM MyFirstArray AS MyFirstArrayType
  ```

- Now we can access the two components as individual variables.
  ```
  MyFirstArray(1) = 10
  MyFirstArray(2) = 20
  MyFirstArray(1) = MyFirstArray(2) DIV MyFirstArray(1)
  ```
The first element of the array now contains the value $\frac{20}{10} = 2$.

- We can also use variables for the index; assume we had declared an integer variable `iIndex`.

  ```vbnet
  DIM iIndex AS Integer
  iIndex = 2
  MyFirstArray( iIndex ) = 5
  ```

- And even more complicated, the index variable may of course be an indexed variable.

  ```vbnet
  iIndex = 1
  MyFirstArray( iIndex ) = MyFirstArray( MyFirstArray( iIndex ) )
  ```

**Note** For keeping track of value changes it is often convenient to draw a table with pencil and paper. But as a rule, a program should always be written and commented so well that is immediately clear what is done when reading the program.

<table>
<thead>
<tr>
<th>State</th>
<th>MyFirstArray(1)</th>
<th>MyFirstArray(2)</th>
<th>iIndex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>20</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>20</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

- Array variables of the same type can be assigned as a whole, no matter how complex they are. This is equivalent to assigning all elements separately.

  ```vbnet
  DIM A1 AS MyFirstArrayType
  DIM A2 AS MyFirstArrayType
  A1(1) = 1
  A1(2) = 2
  A2 = A1  'equivalent to
  ' A2(1) = A1(1)
  ```
' A2(2) = A1(2)

**Note** Neither the compiler nor the interpreter does any index-overflow checking. Hence overwriting of data outside an array may occur and may cause severe errors, if indexes are used that is bigger than the defined upper bounds.

- Arrays cannot be compared directly — it must be done element by element. Often it is useful to declare constants for the upper bound of an array. (For a description of the **IF** and **WHILE** statement see Sections 2.6.2.1 and 2.6.3.1, respectively.)

```plaintext
CONST MaxNoOfHeights AS Integer = 10 'want to have 10 heights

TYPE DIM HeightArrayType(MaxNoOfHeights) AS Double END

DIM HeightArray1 AS HeightArrayType 'first array of heights
DIM HeightArray2 AS HeightArrayType 'second array of heights
DIM iIndex AS Integer 'index for comparing
DIM lEqual AS Logical 'indicator for comparing

'lEqual = TRUE 'so far everything was equal
'iIndex = 1 'start with the first element
'n now compare the arrays
DO WHILE lEqual AND (iIndex <= MaxNoOfHeights)
  lEqual = (HeightArray1( iIndex ) = HeightArray2( iIndex ))
  iIndex = iIndex + 1
LOOP

' do some action according to the result of the comparison
IF lEqual THEN
  'yes, they are equal
ELSE
  'no, they are not equal;
  'the first difference is at position iIndex - 1
END IF

♦ Now declare some larger arrays.

```plaintext
TYPE DIM DoubleArrayType ( 20 ) AS Double END
```
The last example shows that arrays can be nested: the five elements of \texttt{ArrayArrayType} are arrays itself. But there is also a direct way of declaring multidimensional arrays.

\begin{verbatim}
TYPE DIM MatrixType (5,20) AS Angle END
\end{verbatim}

A variable of \texttt{MatrixType} will denote a 5 by 20 matrix of angles (floating point).

In closing let us compare the access to elements of the two multidimensional arrays.

\begin{verbatim}
DIM ArrayArray AS ArrayArrayType
DIM Matrix AS MatrixType

ArrayArray(1)(1) = 1.0
ArrayArray(1)(20) = 20.0
ArrayArray(5)(20) = 100.0

Matrix(1,1) = 1.0
Matrix(1,20) = 20.0
Matrix(5,20) = 100.0
\end{verbatim}

### 2.2.4 Declaration of Structures

A structure (a structured type, also known as a "record" in other languages) consists of a number of values of possibly different types and is declared as follows:

**Syntax:**

\[
\text{TypeDeclaration} ::= \text{"TYPE" } \text{Name} \\
\quad \quad \quad \{ \text{ElementName} \text{ "AS" } \text{DataTypeName} \} \\
\quad \quad \quad \text{"END" [ Name ]}
\]

- A variable of type "Name" will consist of elements (fields, components) which can be accessed by their element name as given in the type declaration (see Section 2.4 on Variables).
• For assignment and parameter passing, the variable may also be used as a whole. Other operations can only be performed on the individual elements; in particular, comparison of entire structures is not possible.

Example:

♦ We declare a type for Cartesian coordinates in the space.

```plaintext
TYPE CartesianPointType
  iNumber AS Integer 'number of the coordinate
dNorth AS Distance 'north coordinate
dEast AS Distance 'east coordinate
dHeight AS Distance 'height coordinate
END CartesianPointType
```

♦ A variable of type `CartesianPointType` will consist of the four components `iNumber`, `dNorth`, `dEast`, and `dHeight`. `iNumber` is an integer for a point number, the others are floating point values (doubles) for the coordinates in the space.

♦ We declare two variables of `CartesianPointType` and initialise the first point’s components to the origin.

```plaintext
DIM Point1 AS CartesianPointType
DIM Point2 AS CartesianPointType

Point1.iNumber = 1
Point1.dNorth = 0.0
Point1.dEast = 0.0
Point1.dHeight = 0.0
```

♦ As with arrays, we can assign a whole structure at once. This is equivalent to assigning each of the components.

```plaintext
Point2 = Point1 'equivalent to
  Point2.iNumber = Point1.iNumber
  Point2.dNorth = Point1.dNorth
  Point2.dEast = Point1.dEast
  Point2.dHeight = Point1.dHeight
```

♦ Now we set `Point2`’s values. Since it is initialised we only need to say where it differs from `Point1`.

```plaintext
Point2.iNumber = 2
Point2.dNorth = 1.0
Point2.dEast = 1.0
```

♦ And we can, for instance, compute the distance between `Point1` and `Point2`. (`Sqr` computes the square root, and `^2` squares its argument.)
DIM dDistance AS Distance

dDistance = Sqr((Point2.dNorth - Point1.dNorth)^2 +
(Point2.dEast - Point1.dEast)^2 +
(Point2.dHeight - Point1.dHeight)^2)

♦ A record type can itself be the type of a record component, or the type of
elements of an array.

```basic
TYPE LineType
  StartPoint AS CartesianPointType
  EndPoint AS CartesianPointType
END LineType

TYPE DIM PointArrayType (5) AS CartesianPointType ENDTYPE

TYPE SomeMeasurementType
  BaseLine AS LineType
  MeasuredPoints AS PointArrayType
END SomeMeasurementType
```

♦ The access to nested structures is done as follows.

```basic
DIM Measurement AS SomeMeasurementType

' set the base line
Measurement.BaseLine.StartPoint = Point1
Measurement.BaseLine.EndPoint = Point2

' set the first point of the measurement
Measurement.MeasuredPoint(1).iNumber = 1
Measurement.MeasuredPoint(1).dNorth = 1.6
Measurement.MeasuredPoint(1).iEast = 5.3
Measurement.MeasuredPoint(1).iHeight = 3.9
```

### 2.2.5 Predefined Structured Types

GeoBASIC provides for the inclusion of system routine calls a set of predefined
structured types (strings, arrays, and structures). The definitions of such predefined
types are implemented in the GeoBASIC compiler and accessible to the
programmer as any other defined types. One example is `GM_Point_Type` which
denotes a GeoMath point data type. Normally they are explained at the beginning
of a subsection.
2.3 DATA DECLARATIONS

2.3.1 Declaration of Constants

Syntax:

```
ConstantDeclaration ::= "CONST" Name [ "AS" DataType ]
"=" Expression
```

The expression is evaluated at compile time and must therefore contain constants only. All GeoBASIC operators may be used, including comparisons and logical operators, but no functions. The name of the constant can subsequently be used wherever a constant of this type is allowed. It is known only inside the unit in which it was declared.

The optional type specification is used to specify an explicit type, e.g. for values of one of the specialities of Double.

In the definitions in the remainder of this document, wherever "Constant" is used in a term, either alone or with a qualifier, such as IntegerConstant etc., either an explicitly written constant as defined in Sections 2.1.6 on Numbers, 2.1.7 on Strings, 2.1.8 on Logical Values, or the name of a declared constant is required.

Examples:

♦ In GeoBASIC the constant Pi is predefined. The definition corresponds to the following constant declaration in the main program.

```
CONST Pi = 3.1415926
```

Note

It is recommended always to specify the type of the constant, even if it is not required by the compiler.

```
CONST Pi AS Double = 3.1415926 'declare Pi as Double explicitely
```

♦ Also string constants can be declared. They may even extend over several lines of code.

```
CONST sProgramTitle = "ATHLETICS DISTANCE MEASURENENT"
```
When declaring constants, the built in arithmetic may be used (but no function calls).

CONSt TwoPi AS Double = 2.0*Pi

2.3.2 Declaration of Variables

Syntax:

VariableDeclaration ::= "DIM" Name [ SubscriptList ] "AS" DataType

DataType ::= ( DataTypeName | "STRING" "*" Length )

SubscriptList ::= "(" UpperBound { "," UpperBound } ")"

UpperBound ::= IntegerConstant

Length ::= IntegerConstant

There are no implicit variable types; all variables used by the program must be explicitly declared to be of a certain data type, whose name may be one of the predefined types (see Section 2.2 on Data Types) or a previously declared array or structure type name (see Section 2.2.3 on Declaration of Arrays, and 2.2.4 on Declaration of Structures). Alternatively, array variables may be declared directly, as explained in the following paragraph.

If a subscript list is specified with the variable name, the variable will denote an array of as many dimensions as there are bounds specified. The upper bounds must be positive integer constants. Subscripting always starts at 1; thus each dimension has "UpperBound" entries. Each element of the array will be of the data type specified.

Variables are known only inside the unit where they are declared.

For string variables and arrays of strings, "Length" specifies the maximum number of characters the variable or the array element is to hold and must be a positive
integer constant. Parts of a string may be accessed and manipulated through standard functions (See 2.7.2.1 Standard Function Calls.) String variables are handled differently if they were declared in global and local scopes. If a string variable is declared globally, then it will be initialised only once, after the program has been loaded. After that point the variable will not be touched again from the environment and it keeps the value the last time assigned to it. A local string variable will be initialised each time the surrounding subroutine (or function) is entered.

**Note**
The declaration of a variable does not assign any value to it. The value of a variable that is read before the first assignment to it has been performed is undefined.

**Examples:**

- First we declare and initialise variables of simple types.
  ```basic
  DIM iSum AS Integer
  DIM dDistance AS Distance
  DIM dHz AS Angle
  iSum = 0
  dDistance = 0.0
  dHz = 100.0
  ```

- Then we declare variables composite types.
  ```basic
  DIM StartPoint AS CartesianPointType
  DIM BaseLine AS LineType
  DIM PointArray AS PointArrayType
  ```

- Arrays can be declared directly.
  ```basic
  DIM NameList ( 8 ) AS String * 50
  DIM AngleMatrix ( 5 , 20 ) AS Angle
  DIM PointArray2 ( 5 ) AS CartesianPoint
  ```

**Note**
If all bounds and the element type of two array variables match, they are considered to be of the same type, hence they can be assigned to each other. For example, the variables `PointArray` and `PointArray2` can be assigned to each other.
2.3.2.1 The Variable Err

The predefined integer variable **Err** can in principle be accessed like any other integer variable. Its main purpose, however, is to contain the error code returned by an external routine called from a GeoBASIC module. Furthermore, at termination of the module's execution, the current contents of **Err** will be passed back to the system as the module's return code. For details on error handling, see Section 2.8 on Error Handling.

2.4 VARIABLES

This section describes the access to variables. Their declaration is described in Section 2.3.2.

Simple variables are accessed by their name. Composite variables (strings, arrays, and structures) can also be accessed by their name, but only for the operations of assignment (see Section 2.6.1.1 on The ... their individual constituents will be selected and operated one by one of the operations available for data of that type.

**Syntax:**

```
Variable ::= VariableName { Selector }
Selector ::= ( ArraySelector | FieldSelector )
ArraySelector ::= ( SubscriptExpression )
FieldSelector ::= . ElementName
SubscriptExpression ::= IntegerExpression
```

An element of a one-dimensional array is accessed with a subscript expression given between parentheses. The expression must be of type Integer and must evaluate to a value between 1 and the upper bound of the array (bounds inclusive).

**Note**

There is no check performed whether the subscript is within bounds, neither at compile time nor at run time.

To access an element of a multidimensional array, as many subscript expressions are needed as there are dimensions.

An element (field) of a structure is accessed by its name.

**Examples for valid variable access (assuming appropriate type definitions)**
Variables of simple types.

<table>
<thead>
<tr>
<th>variable</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSum</td>
<td>Integer</td>
</tr>
<tr>
<td>dAngleDifference</td>
<td>Angle</td>
</tr>
<tr>
<td>dHorizontalDistance</td>
<td>Distance</td>
</tr>
<tr>
<td>lValidPoint</td>
<td>Logical</td>
</tr>
</tbody>
</table>

Variables of compound types.

<table>
<thead>
<tr>
<th>variable</th>
<th>with component/element</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point1</td>
<td>Point1.iNumber</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>Point1.dEastY</td>
<td>Double</td>
</tr>
<tr>
<td>ArrayArray</td>
<td>ArrayArray(1)</td>
<td>DoubleArray</td>
</tr>
<tr>
<td></td>
<td>ArrayArray(1)(1)</td>
<td>Double</td>
</tr>
<tr>
<td>Matrix</td>
<td>Matrix( 1, 1 )</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td>Matrix( x, y )</td>
<td>Double</td>
</tr>
</tbody>
</table>

(with x and y integer variables within the bounds)

For further examples see Sections 2.2.3 on Declaration of Arrays, 2.2.4 on Declaration of Structures, and 2.3.2 on Declaration of Variables.

### 2.5 EXPRESSIONS

Syntax:

- **Expression** ::= LogicalTerm "OR" LogicalTerm
- **LogicalTerm** ::= LogicalFactor "AND" LogicalFactor
- **LogicalFactor** ::= "NOT" LogicalPrimary
- **LogicalPrimary** ::= SimpleExpression [ RelationOperator SimpleExpression ]
- **RelationOperator** ::= "=" | "<" | "<=" | ">" | "<>" | ">=| "<="
- **SimpleExpression** ::= [ AddOperator ] Term
- **AddOperator** ::= "+" | "-"
- **Term** ::= Factor [ MultOperator Factor ]
- **MultOperator** ::= "*" | "/" | "\" | "MOD"
- **Factor** ::= Primary [ "^" Factor ]
- **Primary** ::= ( Variable | Constant | FunctionCall |
The operators have their usual meaning, as found in many programming languages. The logical operators OR, AND, and NOT stand for the inclusive logical or, the logical and, and the logical not. The relational operators =, <>, >, <, >=, <= stand for "equal to", "not equal to", "greater than", "less than", "greater than or equal to", and "less than or equal to", respectively. The arithmetic operators +, -, *, /, \, MOD and ^ stand for addition, subtraction, multiplication, floating point division, integer division, remainder, and power, respectively. Aside from its use as arithmetic addition operator, the + operator is also used for string concatenation.

The syntax for the expressions reflects the precedence of the operators; thus, the logical OR operator has the lowest precedence, since both LogicalTerms are evaluated before the or takes place. The parameters of function calls are evaluated before the function itself. Functions and parenthesised expressions are evaluated before any operations involving them. All operations on the same level are evaluated from left to right, with the exception of powers, which are evaluated from right to left, i.e. \(x^3^2\) is the same as \(x^{(3^2)}\) (= \(x^9\)) and not \((x^3)^2\) (= \(x^6\)). Multiplication, division, and remainder are evaluated before addition and subtraction. Arithmetic operations and string concatenation are performed before comparisons, and comparisons before logical operations. In logical operations, NOT is performed before AND, which is performed before OR.

Note In case of doubt about the precedence, or to make the intention clear to the reader, parentheses are recommended.

Examples

♦ First we declare some variables that will be used.

```
DIM a AS Double
DIM b AS Double
DIM c AS Double
DIM i AS Integer
DIM j AS Integer
DIM k AS Integer
DIM x AS Logical
DIM y AS Logical
DIM z AS Logical
DIM s AS String
```
The implicit precedence of the expression in the left column is shown in the right column explicitly.

<table>
<thead>
<tr>
<th>expression</th>
<th>precedence made explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>a + 3 * b</td>
<td>a + (3*b)</td>
</tr>
<tr>
<td>a / b * c</td>
<td>(a/b) * c</td>
</tr>
<tr>
<td>a ^ 3 ^ b</td>
<td>a^(3^b)</td>
</tr>
<tr>
<td>i \ j \ k</td>
<td>(i \ j) \ k</td>
</tr>
<tr>
<td>x or y and z</td>
<td>x or (y and z)</td>
</tr>
<tr>
<td>x and y = z</td>
<td>x and (y = z)</td>
</tr>
<tr>
<td>a * F((-b) + 1) / 2</td>
<td>(a * (F((-b) + 1)) ) / 2</td>
</tr>
</tbody>
</table>

where \( F \) is a function (see Section 2.7 on Routines; this example is only included for completeness).

Now we show some examples for the type conversion.

<table>
<thead>
<tr>
<th>Expression</th>
<th>value</th>
<th>result type</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 / 3</td>
<td>2.33333333</td>
<td>Double</td>
</tr>
<tr>
<td>7 \ 3</td>
<td>2</td>
<td>Integer</td>
</tr>
<tr>
<td>7 mod 3</td>
<td>1</td>
<td>Integer</td>
</tr>
<tr>
<td>&quot;Geo&quot; + &quot;BASIC&quot;</td>
<td>&quot;GeoBASIC&quot;</td>
<td>String</td>
</tr>
</tbody>
</table>

### 2.5.1 Type Compatibility

Note that not all types of operands can be combined with all operations. The rules are as follows.

#### 2.5.1.1 Addition, subtraction, multiplication (\(+\), \(-\), \(*\)):

Both operands must be of a numeric type (Integer, Double, or any of the various specialities of Double). If both are of the same type, the result is also of that type, otherwise it is of type Double.

**Note** The + operator is also used for string concatenation, see below.

---

1 The actual value depends on the hardware.
2.5.1.2 Division (/):
Both operands must be of a numeric type (Integer, Double, or any of the various specialities of Double). The result is always of type Double. If the value of the denominator is zero, the division is not performed and an error results, which will cause an enabled error handler to become active.

2.5.1.3 Integer division, remainder (\, mod):
Both operands must be of type Integer, and the result is also of type Integer. If the value of the denominator is zero, the division is not performed and an error results, which will cause an enabled error handler to become active.

2.5.1.4 Exponentiation (^):
Both operands must be of a numeric type (Integer, Double, or any of the various specialities of Double). The result is always of type Double. If the exponent is 0, the result is 1.0 for all values of the base. If the base is negative, the exponent must have an integer value, otherwise a domain error occurs.

2.5.1.5 Relational operators ( =, <>, >, <, >=, <=):
Both operands must be either of a numeric type (Integer, Double, or any of the various specialities of Double), or both Logical, or both strings. The result is always of type Logical.
For numerical operands, the relations are the usual. For logical operands, FALSE is less than TRUE. For strings, the ASCII code sequence is used, so that e.g. "0" < "1" < "A" < "Z" < "a" < "z". Comparison of strings proceeds character by character from left to right, and the first unequal pair determines which string is less. Comparison also ends when an "end-of-string" is found; in this case, if both strings are of the same length they are equal, otherwise the shorter is less than the longer. Note that strings of different length can never be equal, but a shorter string can be greater than a longer one.

2.5.1.6 Logical operations:
The logical operators (not, and, or) require their operands (one for not, two for and and or) to be of type Logical. The result is, of course, also of type Logical.

2.5.1.7 String concatenation (+):

Both operands must be string expressions, and the result is again a string, whose length is the sum of the lengths of the two operands and must be less than 256. If string manipulation functions are used in string expressions, all intermediate results from concatenation or string generation must be less than 256 characters long.

Examples

Now we show some examples for string comparison.

<table>
<thead>
<tr>
<th>expression</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Sun&quot; &lt; &quot;Sunny&quot;</td>
<td>TRUE</td>
</tr>
<tr>
<td>&quot;Sun&quot; &gt; &quot;Moon&quot;</td>
<td>TRUE</td>
</tr>
<tr>
<td>&quot;Sun&quot; &lt;&gt; &quot;Sun &quot;</td>
<td>TRUE</td>
</tr>
<tr>
<td>&quot;Sun&quot; &gt; &quot;Sun &quot;</td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Sun&quot; &gt; &quot;Sun&quot;</td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Sun&quot; &lt; &quot;Sun&quot;</td>
<td>FALSE</td>
</tr>
<tr>
<td>&quot;Sun&quot; = &quot;Sun&quot;</td>
<td>TRUE</td>
</tr>
<tr>
<td>* * &gt; * *</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

2.6 STATEMENTS

Syntax:

StatementSequence ::= { [ ErrorLabel ] Statement }
ErrorLabel ::= HandlerLabel ":"  
Statement ::= ( SequentialStatement | SelectionStatement | LoopStatement | OnErrorStatement | ExitStatement | IOSStatement )

The error label is used in conjunction with the ON-ERROR-statement, see Section 2.8; it must be written on a line by itself, i.e. the statement following it must be on a new line.
2.6.1 Sequential Statements

Syntax:

\[
\text{SequentialStatement} ::= ( \text{Assignment} \mid \text{SubroutineCall} )
\]
\[
\text{Assignment} ::= \text{Variable} \mathrm{ \text{"\textendquote}} \text{Expression}
\]

2.6.1.1 The Assignment Statement

The expression is evaluated and the result is assigned to the variable. The type of the variable and the type of the expression must be the same, unless they are of a simple type. In this case they must either be both of a numeric type (\text{Integer}, \text{Double}, or any of the various specialities of \text{Double}), or both of type \text{Logical}. If the variable is of type \text{Integer}, the expression must also be of type \text{Integer}. If the variable is one of the \text{Double} types and the expression is \text{Integer}, the result is converted to \text{Double} before being assigned.

If the variable is an array element, the subscript expression is evaluated before the expression on the right hand side. (This will matter only if functions with side effects are evaluated, which should be avoided.)

A structure variable can be assigned to another one, provided they are both of the same structure type (same name). An array variable can be assigned to another one if both are of the same type (same name) or if they have the same "shape" (the same number of dimensions and the same number of elements in corresponding dimensions) and if their elements are of the same type.

Examples:

♦ Compute the east coordinate of \text{Point1} out of the east coordinate of \text{Point2}.
\[
\text{Point1.dEast} = 2.5 \times \text{Point2.dEast}
\]

♦ The following assignment with \text{i} and \text{j} in the appropriate bounds may occur in some matrix computation.
\[
\text{Matrix}(i, j) = (\text{Matrix}(i+1, j)+\text{Matrix}(i-1, j)) / 2.0
\]

♦ Next, the matrix is assigned to itself. (Note that it is an assignment, not a Boolean expression.)
\[
\text{Matrix} = \text{Matrix}
\]

♦ Often a logical variable (\text{lDone}) has to be set according to some condition. \text{x} and \text{y} must be comparable.
1Done = (x > y)

- In closing a unit is appended to a string s.
  s = s + " cm"

For subroutine calls see Section 2.7.2.

### 2.6.2 Selection Statements

**Syntax:**

```
SelectionStatement ::= ( IfStatement | SelectStatement )
IfStatement ::= "IF" Condition "THEN"
  StatementSequence
  { "ELSEIF" Condition "THEN"
    StatementSequence
  }
  [ "ELSE"
    StatementSequence
  ]
"END IF"
Condition ::= LogicalExpression
SelectStatement ::= "SELECT CASE" Expression
  { "CASE" ConstantList
    StatementSequence
  }
  [ "CASE ELSE"
    StatementSequence
  ]
"END SELECT"
ConstantList ::= Constant { "," Constant }
```

#### 2.6.2.1 The IF-Statement

The conditions are evaluated one after the other. As soon as one is found that results in the value `TRUE`, the statement sequence following the corresponding `THEN` is executed and no further conditions are evaluated. If no condition evaluates to `TRUE`, then the statement sequence after `ELSE` is executed, if there is an `ELSE`, otherwise nothing is done. In any case, execution continues with the statement following `END IF`.

**Examples:**
If a is greater than b, Stat1 will be executed. If a is smaller than b, Stat2 will be executed. The ELSE case means that neither a is greater b, nor a is smaller b — hence a equals b. In that case Stat3 is executed.

```
IF a > b THEN
Stat1
ELSEIF a < b THEN
Stat2
ELSE 'a = b
Stat3
END IF
```

**Note**

In general the branch conditions in the IF-Statement must neither be exclusive nor complete. Hence the compiler will not check if any branch is accessible.

The built-in function `Abs` computes the absolute value of a number, i.e. takes a number and computes its value as a non-negative integer ("forgets its sign"). It can be written as the following program that does nothing if x is already non-negative, and converts x to a positive number if the current value is negative. The empty ELSE case can be omitted.

```
IF x < 0 THEN
  x = -x
END IF
```

Another example is given in the next Section 2.6.2.2 on The SELECT-Statement.

### 2.6.2.2 The SELECT-Statement

The expression is evaluated and compared to the constants. If a constant equal to the value of the expression is found, the corresponding statement sequence is executed. If no constant equals the expression and there is a CASE ELSE, the statement sequence following this is executed, otherwise nothing more is done. Execution then continues with the statement after END SELECT.

The expression and the constants must be of a simple type or strings, and the constants should all have different values. The order of the constants in the list, and the order of the lists in the SELECT-statement is irrelevant as far as the effect of the statement is concerned; however, the constants will be checked for equality in the order in which they appear, so if the most frequent case is put first, this will likely result in faster execution.
There is no check to assure that the constants are all different. If there is more than one constant equal to the value of the expression, the first one will always be selected; the other cases will therefore be inaccessible.

Example:

Assume that the sum of the variables \( a \) and \( b \) denotes an integer, and we want to check if this number is a prime number smaller than 10, a prime number between 10 and 20, or not a prime number at all.

```
SELECT CASE a+b
CASE 2, 3, 5, 7
    Stat1
CASE 11, 13, 17, 19
    Stat2
CASE ELSE
    Stat3
END SELECT
```

Note that if had used a nested IF statement, we would have to write a lot of comparisons that make the code much less readable. (Further, if we do a straight forward transformation from SELECT to IF, the selection expression is evaluated more than once, in the general case.)

```
IF (a+b)=2 OR (a+b)=3 OR (a+b)=5 OR (a+b)=7 THEN
    Stat1
ELSEIF (a+b)=11 OR (a+b)=13 OR (a+b)=17 OR (a+b)=19 THEN
    Stat2
ELSE
    Stat3
END IF
```

2.6.3 Iteration Statements

Syntax:

```
LoopStatement ::= ( WhileLoop | UntilLoop | ForLoop )
WhileLoop ::= "DO" [ "WHILE" Condition ] StatementSequence "LOOP"
UntilLoop ::= "DO" StatementSequence "LOOP" [ "UNTIL" Condition ]
ForLoop ::= "FOR" CounterName ":=" Start "TO"
```
2.6.3.1 The WHILE-Loop

If there is a condition, it is evaluated. If this yields TRUE, the statement sequence is executed once, then the condition is re-evaluated. This continues until the condition evaluates to FALSE, whereupon execution continues with the statement following the loop.

If the condition yields FALSE the first time, the statement sequence is not executed at all, and execution continues immediately with the statement following the loop.

If there is no condition specified, the loop can only be left through an EXIT-statement (see the note on the Exit-Statement at the end of this section), or through the occurrence of a run time error.

An example is given after the description of the UNTIL-loop below.

2.6.3.2 The UNTIL-Loop

The statement sequence is executed, then the condition, if there is one, is evaluated. If this yields FALSE, the statement sequence is executed again, then the condition is re-evaluated. This continues until the condition evaluates to TRUE, whereupon execution continues with the statement following the loop.

If no condition is specified, the loop can only be left through an EXIT-statement (see the note on the Exit-Statement at the end of this section), or through the occurrence of a run time error.

The statement sequence is executed at least once.

Examples:

♦ Assume, for instance, the following variable declarations.

```plaintext
CONST iMaxIndex AS Integer = 10
```
DIM dSum AS Double 'for the summation
DIM iIndex AS Integer 'the running index
DIM iLastIndex AS Integer 'index of last element 
  'to add
DIM NumberArray (iMaxIndex) AS Double 
  'array with the numbers

Then the following WHILE loop sums up iLastIndex(\leq iMaxIndex) numbers of the array NumberArray. The resulting sum will be in dSum.

\[
dSum = 0 \quad \text{'so far the sum is zero}
\]
\[
iIndex = 1 \quad \text{'the first index is 1}
\]

DO WHILE iIndex <= iLastIndex 
  'as long as we are 
    'not at the end
  dSum = dSum + NumberArray(iIndex) 
    'add the 
      'current element
  iIndex = iIndex + 1 
    'compute next index
LOOP

♦ Every WHILE loop can be transformed in an equivalent UNTIL loop and vice versa. Have a look at the following UNTIL version of the summation.

\[
dSum = 0 \quad \text{'so far the sum is zero}
\]
\[
iIndex = 1 \quad \text{'the first index is 1}
\]

DO 
  dSum = dSum + NumberArray(iIndex) 
    'add the current 
      'element
  iIndex = iIndex + 1 
    'next index
LOOP UNTIL iIndex > iLastIndex 
    'until we exceed 
      'the last index

♦ These two loops (the WHILE and UNTIL version) perform exactly the same 
  computation for iLastIndex > 0. But for iLastIndex \leq 0, dSum 
  remains 0 and iIndex remains 1 in the WHILE example, while in the 
  UNTIL version dSum is set to the value of NumberArray(1), and 
  iIndex is incremented once.

2.6.3.3 The FOR-Loop

The three Integer expressions (Start, Finish, Step) are evaluated at the 
outset. If the Step part is omitted, Step is set to +1 by default. The values thus 
obtained for Finish and Step are used throughout execution of the FOR-loop,
which means that they do not change even if their constituent variables should change their values inside the FOR-loop.

**Note** If the value of Step is 0, the loop can only be left through an EXIT-statement (see the note on the Exit-Statement below) or through the occurrence of a run time error.

The Start value is assigned to the counter. Before each execution of the loop, the counter is compared to the Finish value. If the value of Step is positive and the counter is smaller or equal to Finish, or if the value of Step is negative and the counter is greater or equal to Finish, another iteration takes place, otherwise the loop terminates and the statement following it is executed. At the end of each iteration, the counter is incremented by Step (which means a decrement for a negative value of Step). Like the WHILE-loop, a FOR-loop may be executed zero times.

**Note** The counter name must be an Integer variable declared in the same routine as the FOR-loop (i.e. it must be a local variable). Within the loop it can be accessed for reading only; changes to it by the statements inside the loop are not allowed.

The execution of the FOR-loop can be described as follows:

```geo
FOR iIndex = iStart TO iFinish STEP iDelta
  Statements
NEXT iIndex
```

The following WHILE loop is equivalent to the FOR loop.

```geo
' evaluate the bounds at the outset
iIndex = iStart
iFinishEvaluated = iFinish
iDeltaEvaluated = iDelta

DO WHILE (iDeltaEvaluated >= 0 AND
  iIndex <= iFinishEvaluated) OR
  (iDeltaEvaluated < 0 AND
  iIndex >= iFinishEvaluated)
  ' Statements
  iIndex = iIndex + iDeltaEvaluated
LOOP
```

*Example:*
We present the previous example of the **WHILE** loop now as a **FOR** loop. They perform exactly the same calculation, for all values of `iLastIndex`.

```
dSum = 0
FOR iIndex = 1 to iLastIndex
    dSum = dSum + NumberArray(iIndex)
NEXT iIndex
```

**Note on the loop EXIT-Statement**

All three loops — the **WHILE** loop, the **UNTIL** loop, and the **FOR** loop — may contain one or more loop-exit-statements. If one of these is executed, the loop terminates immediately and the statement following it is executed. An EXIT-statement always exits only the innermost loop containing it.

### 2.7 ROUTINES

#### 2.7.1 Routine Declaration

Routines come in two flavours: subroutines and functions. Functions return a value and normally cause no change to the variables of their environment, while subroutines often change their environment. Because they are quite similar, they are described together.

**Syntax:**

```
RoutineDeclaration ::= ( SubroutineDeclaration | FunctionDeclaration )
SubroutineDeclaration ::= [ "GLOBAL" ] "SUB" SubroutineName [ ParameterList ] Body "END" [ SubroutineName ]
FunctionDeclaration ::= "FUNCTION" FunctionName ParameterList "AS" DataTypeName Body "END" [ FunctionName ]
ParameterList ::= "(" [ ParameterSpecification { "," ParameterSpecification } ] ")"
```
ParameterSpecification ::= [ "BYVAL" ] ParameterName "AS" DataTypeName
Body ::= { CVTDeclaration | LabelDeclaration } CodePart
CVTDeclaration ::= ( ConstantDeclaration | VariableDeclaration | TypeDeclaration ) CodePart ::= StatementSequence ExitStatement ::= ( LoopExit | RoutineExit ) RoutineExit ::= "EXIT" ( "SUB" | "FUNCTION" )

Routines that will be called from the TPS-1100-System, so-called modules, must be declared with the keyword GLOBAL. They must be parameter-less subroutines (not functions), and they should return an error code in the predefined integer variable Err. (See also Section 2.3.2.1 on The Variable Err, and Section 2.8 on Error Handling.)

Global subroutine may have a length up to 18 characters.

The names of the parameters in the parameter list can be used inside the routine like variables of the specified type. When the routine is called (executed), actual variables or expressions will be substituted for them. A parameter specified as byVal must not be a structure or an array and can be replaced by a variable or an expression; the parameter behaves like a variable initialised to the value of the expression. Parameters not specified as byVal must be replaced by a variable (of the correct type); any manipulations performed on the parameter are actually performed on the substituted variable.

Functions usually have one or more parameters; if a function has no parameters, the parentheses must still be written. On the other hand, if a subroutine has no parameters, the parentheses may be omitted.

The declaration part of a routine contains local declarations of constants, types, variables, and labels, which will not be known outside the routine.

The code part of a routine contains the statements which are executed when the routine is called.

The code part of a function should contain at least one assignment statement of the form

\[ \text{FunctionName} = \text{Expression} \]

When control returns to the point of call, the value last assigned to the function name will be the value returned by the function. If no such assignment is made before control returns, the return value of the function is undefined.
Both the declaration and the code part may use the names that are known in the environment of the routine, i.e. the globally declared objects, provided their declaration preceded (in the source text) the current routine.

**Note on the routine EXIT-Statement**

The code part of a routine may contain one or more routine-exit-statements, which are written as `EXIT SUB` or `EXIT FUNCTION` for a subroutine or a function, respectively. If one of these is executed, execution of the routine terminates at that point and control passes back to the point where the routine was called. If no such EXIT-statement is executed, control returns to the point of call when the `END` of the routine is encountered.

**Examples:**

- The subroutine `SquareAndCube` takes a `Double` as first argument (the parameter variable `dX`) and returns the square and cube of this first argument in the second (`dSquare`) and third (`dCube`) one.

  ```geo
  SUB SquareAndCube( byVal dX AS Double, dSquare AS Double, dCube AS Double )
  'description: the argument dX is squared and cubed and returned in dSquare and dCube, respectively;
  dSquare = dX * dX
dCube = dX * dSquare
END SquareAndCube
  ```

- The function `AverageAngle` takes a `Matrix` of type `MatrixType` as argument and returns the average of the matrix elements.

  ```geo
  FUNCTION AverageAngle( Matrix AS MatrixType ) AS Angle
  'description: Matrix is a n by m array of Angle
  'return: the average of all its elements
  DIM dSum AS Angle 'sum of the angles
  DIM i AS Integer 'index in the first dimension
  DIM j AS Integer 'index in the second dimension
  dSum = 0 'init the sum to 0
  ```
FOR i = 1 to n  'for all elem. in the first dim.
FOR j = 1 to m 'for all elem. in the second dim.
dSum = dSum + Matrix(i, j) 'sum up the elem.
NEXT j
NEXT i

AverageAngle = dSum / (n*m) 'assign the mean as
' return value
END AverageAngle

♦ The next example shows a possible use of the EXIT SUB statement, and the
difference to the loop EXIT statement.

SUB RoutineWithExit
  'description: demonstrates EXIT SUB and EXIT
  DIM i AS Integer
  DIM lOk AS Logical
  DIM lCond AS Logical...

  lOk = TRUE
  DO WHILE lOk
    FOR i = 1 TO n
      'do something
      IF Error() THEN
        EXIT SUB  ' terminates the subroutine
      END IF
      IF lCond then
        EXIT    ' terminates the loop
      END IF
    NEXT i
    'this will be executed after "EXIT" but
    ' not after "EXIT SUB"
  LOOP
  END RoutineWithExit

2.7.2 Routine Calls

Syntax:

SubroutineCall ::= [ "CALL" ] SubroutineName
                     [ ActualParameterList ]
A subroutine call is a statement by itself and can be written wherever statements are allowed, while a function call is (part of) an expression and can be written wherever expressions are allowed. Standard functions are called like user-defined functions.

When a subroutine or function call is encountered, control passes to the called routine. The parameters of the routine are replaced by the expressions in one of two ways, depending on the specification of the parameter.

If the parameter was specified as `byVal`, the expression is evaluated and the resulting value is passed to the routine as the initial value of the corresponding parameter. If the parameter was not specified as `byVal`, the expression must be a variable of the type specified in the parameter list (possibly an element of a composite variable), and it is passed "by reference", i.e. for this call it takes the place of the parameter in the routine. Any assignment to the parameter becomes an assignment to the actual variable.

Note once again, that variables, including local ones, are not initialised by the compiler. The value of a variable that has not been explicitly assigned a value is undefined.

**Note**

Generic string parameters which are passed by reference are not checked for overwriting length limits. Hence overwriting of subsequent data may happen if the programmer does not care of this limits. E.g. if the program assigns a string which is longer than the data area where the reference is pointing to.

Passing an actual parameter to a typed string parameter (e.g. String30) by reference is limited so far as the actual string parameter has to be of larger or equal length than the formal string parameter. This avoids overwriting of subsequent data.

### 2.7.2.1 Standard Function Calls

A standard function is called like any user-defined function, as part of an expression, returning a value whose type depends on the function and sometimes on the parameters. Unlike user-defined functions, some standard functions are "overloaded", i.e. they can take parameters of different types, or a varying number...
of them. For a list of the available standard functions, see Section Standard functions.

2.7.2.2 External Routine Calls

GeoBASIC provides interfaces to external functions, e.g. system routine calls to get a distance. Such routines can be called like any user defined subroutine. They can takes value and reference parameters of any known type. A speciality of external routines is the fact that they return an error code, which is stored in the predefined variable \texttt{Err} upon return (see Section 2.3.2 on Declaration of Variables). Special actions may be taken by the GeoBASIC module if the error code is not \texttt{RC_OK}; details are given in the following Section 2.8 on Error Handling.

2.8 ERROR HANDLING

Syntax:

\[
\begin{align*}
\text{LabelDeclaration} & ::= \ "LABEL" \ \text{HandlerLabel} \\
\text{OnErrorStatement} & ::= \ "ON ERROR" \ ( \ "RESUME\ NEXT" \ | \ "GOTO" \ ( \ \text{HandlerLabel} \ | \ "0" \ ) \ ) \\
\text{HandlerLabel} & ::= \ \text{Name} \\
\text{ErrorLabel} & ::= \ \text{HandlerLabel} \ ",\"
\end{align*}
\]

An \texttt{ErrorLabel} is used to mark a part of the code and is written on a separate line before the first statement that is to be executed as part of that particular error handler (see also Section 2.6 on Statements). All labels must be declared in the routine in which they label a statement, i.e. the scope of the label is the routine code. An "\texttt{ON ERROR GOTO label}" statement must appear in the same routine as the specified label. The other two "\texttt{ON ERROR}" statements may appear anywhere.

The predefined variable \texttt{Err} is used to signal run-time errors; its value changes in one of three ways.

1) An external TPS-1100 system software routine is called. Upon return \texttt{Err} is always set to the routine's return code. Normally this is 0 (= OK); a non-zero value means that an error has occurred during the execution of the external routine.

2) A run-time error occurs during the execution of GeoBASIC code (e.g. division by zero, illegal instruction).

3) The GeoBASIC module explicitly assigns a value to \texttt{Err}. 
In the first two cases, error handling takes place (if $\texttt{Err} <> 0$) according to the choice then in effect, see below. In the third case, error handling does not take place; execution continues normally, regardless of the error handling choice.

Run-time errors can be handled by the GeoBASIC module in one of the following three ways.

a) Control is passed to an error handler label. This method is chosen by executing \texttt{ON ERROR GOTO LAB}, where \texttt{LAB} is the label of the statement to which control is to be passed. Leaving the active routine will reset the value of \texttt{Err} to Zero.

b) Execution of a GeoBASIC program is terminated immediately after an error occurs. This method is chosen by executing \texttt{ON ERROR GOTO 0}. This is also the default choice, active at the start of the GeoBASIC module.

c) Execution continues with the statement after the call, i.e. the error condition is ignored. This method is chosen by executing \texttt{ON ERROR RESUME NEXT}. The value of \texttt{Err} will be kept if the routine returns to the caller.

In methods a) and c) the variable \texttt{Err} is set to the return code and can be inspected by the program. In method b) \texttt{Err} is set as well, but the program terminates execution. Control and the error code will be passed to the point of the TPS-1100 system where the interpreter has been called.

The activation of an error handler takes place when the execution of an \texttt{ON ERROR} - condition has been passed. \texttt{ON ERROR} - conditions may be defined anywhere in a statement sequence. Passing such a statement resets the value of \texttt{Err} to Zero. In this way, the GeoBASIC programmer has the possibility to control the behaviour of execution depending on the point of execution.

For more information, see the examples below.

\textbf{CAUTION} It is entirely the application programmer's responsibility to make sure that no nonsense results from the use of error handler labels. Particular attention should be paid to the following points.

- If a label is reached in the normal course of code execution, the statements following it will be executed as if the label were not present.
- If "GOTO label" (method a) has been chosen and an error occurs, control will be transferred to that label even when the label is inside a structured statement or in a different routine.
• If control is transferred from outside to a label inside a structured statement, this may have undefined consequences, e.g. in case of a FOR-statement. Such transfers must be avoided.

**Note**  
ERROR, GOTO, and RESUME are not reserved words, but **ON** is.

**Examples:**

♦ First, a simple example. An error will be ignored and passed to the caller.

```basic
SUB ABC
  ON ERROR RESUME NEXT
  ... 'statements
  CALL ExternalSystemRoutine (.).
  ... 'statements
END ABC
```

♦ The next example shows an external system routine call. If an error occurs, then the statements in `ErrLab` may make some changes and try the execution again. If the error occurs a second time, the program aborts immediately.

```basic
SUB Dispatch
  LABEL ErrLab
  ... 'statements
  ON ERROR GOTO ErrLab
  CALL ExternalSystemRoutine (.).
  EXIT
ErrLab:
  ... 'make changes
  ON ERROR GOTO 0  'abort next time
  CALL ExternalSystemRoutine (.).
  ... 'statements
END Dispatch
```
The third example handles an error not caused by an external routine (division by zero).

```basic
SUB MatrixInversion
    LABEL Singular
    DIM i AS Integer
    DIM p AS Double
    ...
    ON ERROR GOTO Singular
    FOR i = 1 TO n
    ...
        p = 1 / a(r, c) 'may divide by zero
    ...
    NEXT i
EXIT SUB
Singular:
    ... 'output error message
END MatrixInversion
```

Please see also the sample program `error_ha.gbs`.

### 2.9 THE PROGRAM

A GeoBASIC program (a loader object) has a structure similar to that of a routine. It has no parameters and no code, but it may contain declarations for common constants, types, and variables, and it contains routine declarations, among them at least one `GLOBAL` subroutine (module).

**Syntax:**

```
Program ::= "PROGRAM" ProgramName
            { CVTDeclaration | RoutineDeclaration }
            "END" [ ProgramName ]
```

The constant, type, and variable declarations (CVTDeclaration) that are global to the entire program are written on this level, as are all routine declarations. These comprise the `GLOBAL` subroutines, i.e. the GeoBASIC modules that can be called from "outside" (from the system), and all local subroutines and functions, which are not accessible from outside.

Global routines (modules) with the names "Stop", "Init", and "Install" have a special function within the TPS-1100-System. ("Stop" and "Init" are reserved names for future using). From the GeoBASIC viewpoint, however, they are declared like any other `GLOBAL` subroutine.
The program name may have up to 18 characters.

2.10 OUTPUT TO THE DISPLAY

Input and output to the display device is not handled by GeoBASIC directly; instead, necessary system routines are called. However, for testing purposes, it is often convenient to have some rudimentary output facilities. GeoBASIC provides a WRITE-statement for this purpose. The simple types (Integer, Double, Logical) and strings can be written one per call.

2.10.1 Write

**Note** During execution of a GeoBASIC program on TPS-1100 system a WRITE-statement has no effect at all. The described behaviour can be observed only if the program is executed on the TPS-Simulator.

**Syntax:**

\[
\text{IOStatement} ::= \text{"WRITE" Expression}
\]

On output, the evaluated expression is written on one line, terminated by return / new line.

- Numeric values are written in a standard format, which for doubles depends on the value. No blanks are output before or after the number.
- Logical values are written as T (true) or F (false), again without surrounding blanks.
- Strings are written as they are, without surrounding quotes or blanks. Output strings may contain any printable characters, including blanks and tabs. A WRITE-call closes the output with CR-LF automatically.

**Examples:**

- We do some output.
WRITE 3 * 6
WRITE 1e3
WRITE 2 > 3
WRITE "this is it"

This will print as
18
1000
false
this is it

2.11 APPINFO-DEFINITION
The AppInfo-definition is an optional compiler-directive which activates the generation of the AppInfo-file during the compilation. The AppInfo-definition has to occur at the end of the source code. Refer to chapter 9.3.2 in the user manual for a description of the AppInfo-functionality.

2.11.1 Syntax in BNF
All entries embraced by curly braces are optional. Also, the AppInfo-section as a whole is optional.

Abbreviations used in the following Syntax:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlobalSubName</td>
<td>Name of a global subroutine</td>
</tr>
<tr>
<td>StringConstant</td>
<td>String constant</td>
</tr>
<tr>
<td>CapLg</td>
<td>Caption Long: Application name for a menu item</td>
</tr>
<tr>
<td>CapSh</td>
<td>Caption Short: Application name for a button</td>
</tr>
<tr>
<td>Desc</td>
<td>Description in Customization Tool</td>
</tr>
<tr>
<td>Help</td>
<td>Help Text</td>
</tr>
<tr>
<td>TheoModel</td>
<td>Theodolite Model</td>
</tr>
<tr>
<td>Author</td>
<td>Author of the GeoBASIC Program</td>
</tr>
</tbody>
</table>

Syntax:
AppInfo ::= "APPINFO"
            [ GeneralSection ]
            [ GlobalSubSection ]
            "END" "APPINFO"

GeneralSection ::= "GENERAL"
            [ GeneralSectionEntry ]
            "END" "GENERAL"

GlobalSubSection ::= "ENTRYPOINT" GlobalSubName
            [ GlobalSubSectionEntry ]
            "END" [ GlobalSubName ]

GeneralSectionEntry ::= "SET"
            GeneralSectionKey
            StringConstant

GlobalSubSectionEntry ::= "SET"
            GlobalSubSectionKey
            StringConstant

GeneralSectionKey ::= "AUTHOR" |
            "DESC" |
            "THEOMODEL"

GlobalSubSectionKey ::= "CAPSH" |
            "DESC" |
            "HELP"

Example:

'Application Information for Config Tool
'---------------------------------------
APPINFO

GENERAL
   SET Author  "Leica AG, CH - Heerbrugg"
   SET Desc   "AppInfo Example Application"
   SET TheoModel  "TCA1100"
END GENERAL

ENTRYPOINT GlobalSub1
   SET CapLg "Global Sub 1"
   SET CapSh "GSUB1"
   SET Desc  "test of appinfo subroutine 1"
END GlobalSub1

ENTRYPOINT GlobalSub2
   SET CapLg "Global Sub 2"
   SET CapSh "GSUB2"
   SET Help   "displays a message and exits"
END GlobalSub2

END APPINFO

See explanations of the example in section 9.3.2 of the user manual.

2.12 IN-/OUTPUT TO FILES

The I/O-routines to files are realised as external routines. Therefore, all the rules explained in chapter 2 have to be applied to the description here too.
2.12.1 Summary of Lists of Types and Procedures

2.12.1.1 Types

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE_EXT_Type</td>
<td>A filename inclusive the extension (exclusive path).</td>
</tr>
<tr>
<td>FILE_STAT_Type</td>
<td>Specific data about a file.</td>
</tr>
<tr>
<td>FileId</td>
<td>File identification.</td>
</tr>
<tr>
<td>FileName</td>
<td>String of 64 characters. Contains a file path and file name.</td>
</tr>
<tr>
<td>MEM_CARD_INFO_Type</td>
<td>Information about the PC card.</td>
</tr>
</tbody>
</table>
### 2.12.1.2 Procedures

<table>
<thead>
<tr>
<th>Procedure Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChDir</td>
<td>Changes the current directory to a given drive and directory.</td>
</tr>
<tr>
<td>Close</td>
<td>Close a file.</td>
</tr>
<tr>
<td>CurDir$</td>
<td>Delivers the current directory including the drive.</td>
</tr>
<tr>
<td>Eof</td>
<td>Examines if end-of-file has been reached.</td>
</tr>
<tr>
<td>FileCopy</td>
<td>Copies a file’s contents to another.</td>
</tr>
<tr>
<td>GetDirectoryList</td>
<td>Get a directory list of entries.</td>
</tr>
<tr>
<td>GetFileStat</td>
<td>Get information about a file.</td>
</tr>
<tr>
<td>GetInt, GetDouble, GetLogical, GetString</td>
<td>Reads a value in binary mode from a file.</td>
</tr>
<tr>
<td>GetMemoryCardInfo</td>
<td>Get information about the current mounted PC card.</td>
</tr>
<tr>
<td>Input</td>
<td>Reads ASCII text from a file in sequential mode.</td>
</tr>
<tr>
<td>Kill</td>
<td>Removes a given file.</td>
</tr>
<tr>
<td>MkDir</td>
<td>Creates a directory in the current directory.</td>
</tr>
<tr>
<td>Open</td>
<td>Open a file in a specific mode.</td>
</tr>
<tr>
<td>Print</td>
<td>Writes ASCII text into a file in sequential mode.</td>
</tr>
<tr>
<td>PutInt, PutDouble, PutLogical, PutString</td>
<td>Writes a value in binary mode into a file.</td>
</tr>
<tr>
<td>RenameDir</td>
<td>Renames a directory.</td>
</tr>
<tr>
<td>RenameFile</td>
<td>Renames a file.</td>
</tr>
<tr>
<td>RmDir</td>
<td>Removes the given directory.</td>
</tr>
<tr>
<td>Seek</td>
<td>Positions the file pointer to a specific byte location.</td>
</tr>
<tr>
<td>Tell</td>
<td>Delivers the current file pointer.</td>
</tr>
</tbody>
</table>

### 2.12.2 File Operation Data Structures

#### 2.12.2.1 MEM_CARD_INFO_Type – PC Card information
2.12.2.2 FILE_STAT_Type – File specific data

TYPE FILE_STAT_Type
sFileName AS FILE_EXT_Type
DateTime AS Date_Time_Type
iSize AS Integer
lReadOnly AS Logical
lSubDir AS Logical
lArchive AS Logical
END FILE_STAT_Type

2.12.3 Open

Description
Opens a file.
Record oriented file operations are not supported yet
Open( ByVal sFileName AS FileName,  
    ByVal sMode AS String20,  
    FileId AS FileId,  
    ByVal iRecLen AS Integer )

Remarks
The Function attempts to open the file given in sFileName with mode sMode. If the procedure is successful a valid file descriptor is returned. This file descriptor is used for all successive operations on the opened file. The device of the PC-Card, which is also the default device, is "A:". An Open will not change the default device nor the default directory. Directories included in sFileName must exist already. The FileId will be determined automatically. There is no need at all to handle the value of FileId directly! No white spaces (spaces, tabs, etc.) may be included in sFileName.

Note
If the device is not mounted, an error code will be returned.

The iRecLen parameter will be ignored, hence it has no effect at all. Its usage is reserved for future purposes.

Access modes may not be mixed, hence opening for Input and Output does not work. A maximum of 20 files can be opened simultaneously.

Parameters

sFileName in File path and name of the file to be opened ("A:\dir\filename.ext", up to 100 characters).

sMode in Access mode
- "Input" - Opens a text file for reading. The file must exist.
- "Output" - Creates a text new file for writing or truncates it to zero if it
− "Append" - Opens an existing text file at the end of it (EOF). If the file does not exist, it will be created
− "InBin" - Opens a binary file for reading.
− "OutBin" - Creates a binary file for writing. If it exists then the file will be truncated to zero length.
− "UpdateBin" - Opens a binary file for reading and writing. After a successful open the file pointer points to the beginning of the file. If the file does not exist it will be created.

```
FileId out
iRecLen in
```

Unique file-id (output).
The record length is set to a default of 1 byte in any case.

See Also
Close, Input, Print

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>file opened successfully</td>
</tr>
<tr>
<td>BAS_FIL_INV_MODE</td>
<td>invalid access mode (see par. sMode)</td>
</tr>
<tr>
<td>BAS_FIL_ILL_NAME</td>
<td>illegal file name specified</td>
</tr>
<tr>
<td>BAS_FIL_TABLE_FULL</td>
<td>the internal file id table is full</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FATAL_ERROR</td>
<td>other fatal error</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>device errors</td>
</tr>
</tbody>
</table>

| RC_FIL_FAT_ERROR    | fatal error in accessing the file allocation table                         |
### RC_FIL_ILLEGAL_DRIVE
illegal drive specified

### RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE
No memory card inserted or it has been removed and put in again. Further file operations are not save.

### RC_FIL_PATTERN_DOES_NOT_MATCH
directory error

### RC_FIL_FILNAME_NOT_FOUND
tried to access a non-existing file

#### Example
Open a file in "Output" access mode for writing.

```basic
DIM FileId AS FileId
Open("A:\test.dat", "Output", FileId, 0 )
```

### 2.12.4 Close

#### Description
Closes a file.

#### Declaration
```basic
Close( byVal fileId AS FileId )
```

#### Remarks
Closes the file as represented by the file descriptor.

#### Parameters
- **FileId** in Unique file-id returned by `Open`.

#### See Also
Open, Print, Input

#### Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>file closed successfully</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non-existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FATAL_ERROR</td>
<td>other fatal error</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td><em>device errors</em></td>
</tr>
<tr>
<td></td>
<td>Fatal error in accessing the file</td>
</tr>
</tbody>
</table>
allocation table.

Illegal drive.

Unspecified error on writing to a file.

No memory card inserted or it has been removed and put in again. Further file operations are not save.

file errors
g file descriptor is not valid. May occur e.g. if closed twice.

Example

Close a file. The fileId has to be returned (by Open):

```
DIM FileId AS FileId

Open("A:\\test.dat", "Output", FileId, 0 ) 'do some work
Close( FileId )
```

### 2.12.5 Input

**Description**

Read a string from file.

**Declaration**

```
Input( ByVal FileId AS FileId,
      sData AS String255,
      iSize AS Integer )
```

**Remarks**

The functions read a string from the file identified by FileId. iSize determines how many characters have to be read from the file at a maximum. If the line terminator occurs before iSize characters has been read, than sData will contain only characters up to the terminator. The current file pointer will be set to the position after the terminator. The line terminator will never be included in the resulting string. The line terminator will be expected as "CR/LF". End-of-file (EOF) can be examined by
calling Eof(). iSize, if greater, will be reset to 255 characters without notification to the caller.

Note

The file must have been opened successfully in access mode "Input".

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileId</td>
<td>in</td>
<td>Unique file-id returned by Open.</td>
</tr>
<tr>
<td>sData</td>
<td>out</td>
<td>The read data.</td>
</tr>
<tr>
<td>iSize</td>
<td>inout</td>
<td>Number of bytes to be read. Number of bytes actually read from file.</td>
</tr>
</tbody>
</table>

See Also

Open
Close
Print

Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>data read successfully</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
<tr>
<td>BAS_FIL_ILL_OPER</td>
<td>Illegal file operation. Operation and access mode do not correspond.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_FILE_DESCR</td>
<td>illegal file descriptor used</td>
</tr>
</tbody>
</table>

Example

Read a string from a file in "Input" access mode.

```
DIM FileId AS FileId
DIM sFileinput AS String255
```
2.12.6 Print

**Description**  
Write a string to a file.

**Declaration**  
Print( ByVal FileId AS FileId,  
       ByVal sData AS String255 )

**Remarks**  
The function writes a string to the file specified by FileId. The actual string determines the numbers of characters which will be written to the file. The printed string will include the line terminator at the end, which will be in any case "CR/LF".

**Note**  
The file must have been opened in access modes "Output" or "Append". Each Print prints the line terminator to the file automatically.

**Parameters**

- **FileId**  
in  
Unique file-id returned by Open.

- **sData**  
in  
The data to be written (of the specified type).

**See Also**

- Open
- Close
- Input

**Error Codes**

- **RC_OK**  
data written
Error in internal memory allocation. May be during open access of a non existing directory.

**device errors**

- **RC_FIL_MEMORY_FAILED**
  - fatal error in accessing the file allocation table

- **RC_FIL_WRITE_TO_MEDIUM_FAILED**
  - unspecified error on writing to a file medium is full

- **RC_FIL_MEDIUM_FULL**
  - No memory card inserted or it has been removed and put in again. Further file operations are not save.

**file errors**

- **RC_FIL_INVALID_FILE_DESCR**
  - illegal file descriptor used

- **RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE**
  - illegal file operation, hence using it on a file which has not been opened in sequential OUTPUT or APPEND mode.

**Example**

Write a string to an "Output" file. The FileId has to be defined (used by Open):

```bas
DIM FileId AS FileId
Open ( "A:\test.txt", "OUTPUT", FileId, 1 )
Print( FileId, "distance measuring" )
Close( FileId )
```

**2.12.7 Get – values**

**Description**

Read a value from file in binary mode.
Declaration

GetByte ( byVal FileId AS FileId, iVal AS Integer )
GetInt ( byVal FileId AS FileId, iVal AS Integer )
GetDouble ( byVal FileId AS FileId, dVal AS Double )
GetLogical( byVal FileId AS FileId, lVal AS Logical )
GetString ( byVal FileId AS FileId, szVal AS String255, iLen AS Integer )

Remarks

These functions read a value from the file identified by FileId. The values will not be interpreted at all. Only logical values will be transformed to the internal coding. iLen gives the maximum number of characters to be read. iLen, if greater, will be reset to 255 characters without notification to the caller. End of file can be recognised by calling Eof(). If end of file has been reached then it is not guaranteed that the returned value is valid.

Note
The file must have been opened successfully in access mode "InBin" or "UpdateBin". The binary values will be interpreted in standard DOS format. GetString reads as many characters as asked. If the read string contains a 0x00-byte (internal terminator) then successive string operations will interpret the string up to this terminator.

Parameters

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileId</td>
<td>in</td>
<td>Unique file-id returned by Open.</td>
</tr>
<tr>
<td>GetByte</td>
<td>iVal</td>
<td>out</td>
</tr>
<tr>
<td>GetInt</td>
<td>iVal</td>
<td>out</td>
</tr>
<tr>
<td>GetDouble</td>
<td>dVal</td>
<td>out</td>
</tr>
<tr>
<td>GetLogical</td>
<td>lVal</td>
<td>out</td>
</tr>
</tbody>
</table>
GetString szVal

ilen In out
ilen characters read
ilen characters to be read. Returns actual length of read data. EOF may be a reason which reduces this value.

See Also
Open
Close
Put - values

Error Codes

RC_OK data read successfully
RC_FIL_MEMORY_FAILED Error in internal memory allocation. May be during open access of a non existing directory.

device errors
RC_FIL_FAT_ERROR fatal error in accessing the file allocation table
RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE No memory card inserted or it has been removed and put in again. Further file operations are not save.

file errors
BAS_FIL_ILL_OPER illegal file operation, hence using it on a file which has not been opened in InBin or UpdateBin mode.

invalid file descriptor used

Example
The example copies a file.

DIM iFId1 AS FileId
DIM iFId2 AS FileId
DIM i AS Integer

Open { "A:\source.txt", "InBin", iFId1, 1 }
Open { "A:\target.txt", "OutBin", iFId2, 1 }
IF EOF(iFId1) THEN
2.12.8 Put – values

**Description**
Put a value to file in binary mode.

**Declaration**
- `PutByte ( byVal FileId AS FileId, iVal AS Integer )`
- `PutInt ( byVal FileId AS FileId, iVal AS Integer )`
- `PutDouble ( byVal FileId AS FileId, dVal AS Double )`
- `PutLogical( byVal FileId AS FileId, lVal AS Logical )`
- `PutString ( byVal FileId AS FileId, szVal AS String255, iLen AS Integer )`

**Remarks**
These functions write a value to the file identified by FileId. The values will not be interpreted at all. Only logical values will be transformed to the external coding. iLen gives the maximum number of characters to be written. If iLen is greater than the actual length, then the string will be filled up with ‘0’-characters. If iLen is greater than 255, then it will be reset to 255. If less than 0 then it will be reset to 0.
The file must have been opened successfully in access mode "OutBin" or "UpdateBin".
The binary values will be written in standard DOS format.

**Parameters**

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Field</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>PutByte</td>
<td>iVal</td>
<td>in</td>
<td>1 byte binary integer, only the lowest order byte will be taken of the input parameter.</td>
</tr>
<tr>
<td>PutInt</td>
<td>iVal</td>
<td>in</td>
<td>4 byte binary integer.</td>
</tr>
<tr>
<td>PutDouble</td>
<td>dVal</td>
<td>in</td>
<td>8 byte binary double float.</td>
</tr>
<tr>
<td>PutLogical</td>
<td>lVal</td>
<td>in</td>
<td>1 byte: FALSE - 0, TRUE - 1</td>
</tr>
<tr>
<td>PutString</td>
<td>szVal</td>
<td>in</td>
<td>String to be written. iLen characters will be written. Note:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>if len(szVal) &gt; iLen then szVal will be cut off.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00 characters will be filled up with 0x00-characters.</td>
</tr>
<tr>
<td>iLen</td>
<td>in</td>
<td></td>
<td>iLen characters to be written.</td>
</tr>
</tbody>
</table>

**See Also**

- Open
- Close
- Get - values

**Error Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>data written successfully</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
</tbody>
</table>

*device errors*
### 2.12.9 Tell

**Description**
Delivers the current position of the file pointer.

**Declaration**
```
Tell( ByVal FileId AS FileId, 
      iPos   AS Integer )
```

**Remarks**
The procedure returns the current byte position of the file pointer which has been set by the last read or write operation. `iPos` will get 1 for the first byte.

**Note**
Other than read and write operations `Tell` do not set the file pointer. Hence after opening a file in APPEND mode `Tell` will yield into 1, since the file pointer has not been set so far.

**Parameters**
- **FileId** `in` Unique file-id returned by `Open`.
- **iPos** `out` The current byte file position.

### File Errors
- **RC_FIL_FAT_ERROR**
  fatal error in accessing the file allocation table
- **RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE**
  No memory card inserted or it has been removed and put in again. Further file operations are not save.
- **BAS_FIL_ILL_OPER**
  illegal file operation, hence using it on a file which has not been opened in `OutBin` or `UpdateBin` mode.
- **RC_FIL_INVALID_FILE_DESCR**
  illegal file descriptor used
- **RC_FIL_NO_MORE_ROOM_ON_MEDIUM**
  memory device is full

**Example**
see Get-values example
See Also

Open
Seek

Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>operation successfully finished</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_FILE_DESCR</td>
<td>illegal file descriptor used</td>
</tr>
</tbody>
</table>

2.12.10 Seek

Description
Sets the current position of the file pointer.

Declaration

```
Seek( ByVal FileId AS FileId,
     ByVal iPos AS Integer )
```

Remarks
The procedure sets the current byte position of the file pointer where the next file operation has to take place. FIL_EOF may be used for iPos to set the file pointer to end-of-file. If iPos is greater than the length of the file no return code will be produced. The file pointer will be set to end-of-file.

Note
Seek may be used on files only which have been opened successfully with access modes "Input", "InBin" or "UpdateBin".

Parameters
**See Also**

Open  
Tell

**Error Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>operation successfully finished</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_FILE_DESCR</td>
<td>illegal file descriptor used</td>
</tr>
<tr>
<td>BAS_FIL_ILLEGAL_POSITION</td>
<td>illegal file position, hence &lt; 1</td>
</tr>
<tr>
<td>BAS_FIL_ILL_OPER</td>
<td>illegal file operation, hence using it on a file opened in sequential OUTPUT or APPEND mode.</td>
</tr>
</tbody>
</table>

**Example**

Getting the length of a text file.

```geo
DIM FileId AS FileId  
DIM nLen AS Integer  
Open ("A:\test.txt", "INPUT", FileId, 1)  
Seek (FileId, FIL_EOF)  
Tell (FileId, nLen) 'one more than the length  
nLen = nLen - 1 'the length of the file  
Close( FileId )
```
### 2.12.11 Eof() (standard function)

**Description**  
Examines if end-of-file has been reached.

**Declaration**  
`Eof( byVal FileId AS FileId ) AS Logical`

**Remarks**  
The function examines if end-of-file has been reached by the last file operation.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileId</td>
<td>Unique file-id returned by Open.</td>
</tr>
</tbody>
</table>

**See Also**  
Open, Input

**Error Codes**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>operation successfully finished</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_FILE_DESCR</td>
<td>illegal file descriptor used</td>
</tr>
</tbody>
</table>
Example

Opens a file in current directory on default drive. Inputs data and examines if EOF has been reached.

```
DIM FileId AS FileId
DIM sIn AS String255
DIM nLen AS Integer

Open ("test.txt", "INPUT", FileId, 1)
DO WHILE NOT Eof(FileId)
  nLen = 255
  Input(FileId, sIn, nLen)
  'process in-data
LOOP
```

2.12.12 CurDir$

Description

Get current directory.

Declaration

```
CurDir$( szcurDir AS FileName )
```

Remarks

The procedure gets the absolute path of the current directory.

Note

Since on TPS only memory card device A: will be supported only paths with drive A: will be returned.

Parameters

```
szcurDir out
```

The current directory and drive.

See Also

ChDir,

MkDir

Error Codes

```
RC_OK
RC_FIL_MEMORY_FAILED
RC_FIL_FAT_ERROR
```

operation successfully finished

Error in internal memory allocation. May be during open access of a non existing directory.

device errors

fatal error in accessing the file
2.12.13 ChDir

Description
Changes the current directory.

Declaration
ChDir( ByVal szName AS FileName )

Remarks
After calling ChDir all subsequent file operations will occur in the current directory if no absolute path is given.

Note
On TPS only the memory card device will be supported. Hence only paths with drive A: will be supported.

Parameters

| szName | in | Name of the next directory. |

See Also
CurDir$, MkDir, RmDir

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>current directory changed</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL.Fat_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
</tbody>
</table>
2.12.14 MkDir

Description
Creates a directory entry.

Declaration
MkDir( byVal szName AS FileName )

Remarks
If szName contains a relative path to the directory then it will be created relative to the current directory. Given an absolute path MkDir will create the directory at the absolute position.

Note
On TPS only the memory card device will be supported.

Parameters
szName in Name of the file to be created.

See Also
CurDir$, ChDir, RmDir

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>directory created</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
<tr>
<td>RC_FIL_NO_MAKE_DIRECTORY</td>
<td>directory could not be created, because, for example, the directory</td>
</tr>
</tbody>
</table>
2.12.15 RmDir

Description
Removes a directory.

Declaration
RmDir( byVal szName AS FileName )

Remarks
The procedure removes a directory with name szName. szName will be interpreted either as relative to current directory or absolute.

Note
The directory must exist and must be empty.

Parameters

| in | szName | Name of the directory. |

See Also
CurDir$,
MkDir

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>directory removed</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
</tbody>
</table>
## 2.12.16 Kill

**Description**  
Deletes an existing file.

**Declaration**  
```basic
Kill( byVal szName AS FileName )
```

**Remarks**  
The name may be given relative to the current directory or absolute.

**Note**  
The file must exist.

**Parameters**  
- `szName`  
in  
Name of the file to be deleted.

**See Also**  
Open  
RmDir

**Error Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>file removed</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. May be during open access of a non existing directory.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>fatal error in accessing the file allocation table</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again. Further file operations are not save.</td>
</tr>
<tr>
<td>RC_FIL_FILNAME_NOT_FOUND</td>
<td>the given file has not been found</td>
</tr>
</tbody>
</table>
2.12.17 GetMemoryCardInfo

Description
Get information about the memory card.

Declaration
GetMemoryCardInfo
(MCInfo AS MEM_CARD_INFO_Type)

Remarks
The function returns the label, the total capacity, the free capacity and the memory medium of the current mounted PC card. It also get the information if the current mounted PC card is write protected or not.

TPS_Sim
On the simulator the requested drive will be derived from the current setting of GSI data path. Since Win95/WinNT support disk sizes larger than 2GB any capacity between 2 and 4 GB will returned as a negative number. Any capacity above 4GB will be returned as –1.

Parameters
MCInfo out
Information about the current mounted PC card.

See Also
-

Error Codes
RC_OK
Successfully completed.
device errors
RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE
No memory card inserted or it has been removed and put in again.

Example
see example dirlist.gbs
2.12.18 GetFileStat

Description  Get specific data about a file.

Declaration  GetFileStat
             ( ByVal sFileName As FileName,
               FStat As FILE_STAT_Type )

Remarks  The function returns data about a file. This function follows the
          same pattern matching rules as GetDirList.

TPS_Sim  DOS handles the root directory differently to
          subdirectories. Therefore calling this function with ".:"
          in the root and ":.." in a subdirectory of root will cause
          an error on the simulator.

Parameters

sFileName  in  Pattern for the requested file.
FStat  out  Specific data of a file which matches
            the pattern given in sFileName.

See Also  -

Error Codes

RC_OK  Successfully completed.
RC_FIL_MEMORY_FAILED  Error in internal memory allocation.
                       Maybe because of an access of a non
                       existing directory or drive.

device errors

RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE  No memory card inserted or it has
                                     been removed and put in again.

RC_FIL_INVALID_PATH  The given file name pattern does not
                     conform to file path rules.

RC_FIL_PATTERN_DOES_NOT_MACTH  The given file name pattern does not
                                match against any directory entry.

Example  see example dirlist.gbs
2.12.19 GetDirectoryList

**Description**
Get a list of entries of the given directory.

**Declaration**

```plaintext
GetDirectoryList
( byVal sPattern As FileName,
  byVal lInclDir As Logical,
  DirList As ListArray,
  iItems As Integer )
```

**Remarks**
The function returns a list filled with directory entries of the given directory which match the given file name pattern. If `lInclDir` is TRUE all subdirectory entries in this directory will be included in the list. The current implementation of `ListArray` contains LIST_ARRAY_MAX_ELEMENT elements. If the directory contains more entries then the last list entry will have "--- more ---" assigned to. Pattern matching characters are all valid file name characters, ",*" and ",?". The former matches one or more characters and the latter matches exactly one character. For further information please refer to a DOS reference guide. For the definition of `ListArray` refer to MMI_InputList.

**Note**
As a valid drive specification only "A:\" is allowed. Hidden and system flagged files will be ignored for the entry list.

**Parameters**

- `sPattern` in
  Pattern for the requested files.
- `lInclDir` in
  TRUE: include subdirectories, FALSE: list files only.
- `DirList` out
  List of directory entries.
- `iItems` out
  Actual number of items, list length.

**See Also**
-

**Error Codes**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successfully completed.</td>
</tr>
</tbody>
</table>
Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. Maybe because of an access of a non existing directory or drive.</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>No memory card inserted or it has been removed and put in again.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_PATH</td>
<td>The given file name pattern does not conform to file path rules.</td>
</tr>
<tr>
<td>RC_FIL_PATTERN_DOES_NOT_MATCH</td>
<td>The given file name pattern does not match against any directory entry.</td>
</tr>
</tbody>
</table>

Example

see example dirlist.gbs

2.12.20 FileCopy

Description
Copies a file’s contents to another.

Declaration
FileCopy( ByVal sSrc As FileName, ByVal sDot AS FileName )

Remarks
This function copies the contents of the source file (first parameter) to the destination file. The source and destination patterns must be absolute for it to be guaranteed to work! No wildcards are allowed in the names of the source or destination paths. The name of the file in the source path is not used for the destination name when the destination only has a directory path: this is an error as no destination file has been explicitly named!

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sSrc</td>
<td>As FileName</td>
<td>Absolute path of source file.</td>
</tr>
<tr>
<td>sDot</td>
<td>As FileName</td>
<td>Absolute path of destination file.</td>
</tr>
</tbody>
</table>

See Also
-

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successfully completed.</td>
</tr>
</tbody>
</table>
GeoBASIC Reference Manual

2. GeoBASIC Constructs

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. Maybe because of an access of a non existing directory or drive.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_PATH</td>
<td>If path is not existent.</td>
</tr>
</tbody>
</table>

Example

FileCopy(“A:\test.gsi”, “A:\GSI\data_1.gsi”)

2.12.21 RenameFile

Description

Renames a file.

Declaration

```
RenameFile( ByVal sOldName As FileName, 
            ByVal sNewName AS FileName )
```

Remarks

This function renames the file sOldName to sNewName. There should be no previous object (file or directory) with the new name. Absolute and relative paths can be used. A file cannot be moved from one directory to another.

Parameters

- `sOldname` in: Existing file name (with path).
- `sNewName` in: New name for file (with path).

See Also

- 

Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successfully completed.</td>
</tr>
<tr>
<td>RC_FIL_RENAME_FILE_FAILED</td>
<td>File sOldname not found or sNewName exists already.</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>Error in internal memory allocation. Maybe because of an access of a non existing directory or drive.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_PATH</td>
<td>If path is not existent.</td>
</tr>
</tbody>
</table>
Example

RenameFile("A:\test.gsi", "A:\test_1.gsi")

2.12.22 RenameDir

Description
Renames a directory.

Declaration

```
RenameDir( ByVal sOldName As FileName, 
            ByVal sNewName AS FileName )
```

Remarks
This function renames an existing directory to that given in sNewName. There should be no previous object (file or directory) with the new name. Absolute and relative paths can be used, but they must be already in existence. A directory cannot be moved from one root path to another.

Parameters

- `sOldname in` Existing directory name (with path).
- `sNewName in` New name for directory (with path).

See Also
-

Error Codes

- **RC_OK** Successfully completed.
- **RC_FIL_RENAME_DIR_FAILED** Directory sOldName not found or sNewName exists already.
- **RC_FIL_MEMORY_FAILED** Error in internal memory allocation. Maybe because of an access of a non-existing directory or drive.

Example

```
RenameDir("A:\GSI", "A:\GSI_1")
```

2.13 COMMUNICATION FUNCTIONS

2.13.1 Send

**Description**  Sends a string to the serial interface. The actual settings will be used to send data over the serial line.

**Declaration**  
```basic
Send(byVal sMessage AS String255)
```

**Remarks**  
The routine `Send` sends a message with a maximal length of 255 characters to the serial line. No formatting at all will be done but a TPS predefined terminator at the end will be added automatically to the message.

**Note**  The data-link must be active. The parameters for the transmission can be set in the GSI communications dialog.

**TPS_Sim**  Executing a GeoBASIC program on the TPS-Simulator redirects the communication stream to the debug window.

**Parameters**  
- `sMessage` in  The message string.

**See Also**  
- Receive
- COM_SetTimeOut

**Error Codes**  
- **RC_OK**  Send has been completed successfully.

**Example**  
The example uses the routine `Send` to send a message.

```basic
Send("This is a message for the routine " + "Send."
```

Send("This is a message for the routine " + "Send."
```
2.13.2 Receive

**Description**  Receives a string from the serial interface. The actual settings will be used to receive data from the serial line.

**Declaration**  
```
Receive( sMessage AS String255,  
nLength AS Integer )
```

**Remarks**  The routine `Receive` reads a message with a maximal length of 255 characters from the serial line. No formatting at all will be done. The routine will return from execution when either `nLength` characters or the pre set terminator has been received or the pre set time-out has been reached. An eventually received terminator will be excluded in the received message.

**Note**  The data-link must be active. The parameters for the transmission can be set in the GSI communications dialog.

- If time-out is reached, less characters than requested (even Zero) may be received.
- If `nLength > 255` then it will be limited to 255 automatically without notification of the caller.

**TPS_Sim**  Calling `Receive` on the TPS-Simulator has no effects.

**Parameters**

- `sMessage out`  The received message string.
- `nLength inout`  
  - **In:**  The maximum number of characters to be received.
  - **Out:**  The actual number of characters received.

**See Also**  
Send  
`COM_SetTimeOut`
## Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Receive has been completed successfully.</td>
</tr>
<tr>
<td>COM_OVERRUN</td>
<td>More characters than requested has been accounted in the internal buffer. Additional characters will be deleted and cannot be retrieved by a subsequent call.</td>
</tr>
<tr>
<td>COM_TIME_OUT</td>
<td>Time-out has been reached.</td>
</tr>
</tbody>
</table>

## Example

The example calls a procedure to process a successful received string. If the reception has not been completed successfully then nothing will be done. The time-out period will be set to 1 second.

```geobasic
DIM iSize AS Integer
DIM sIn AS String255
ON ERROR RESUME NEXT
COM_SetTimeOut (1)
iSize = 255
Receive (sIn, iSize)
IF Err = RC_OK THEN
    ProcessString (sIn)
END IF
```

### 2.13.3 COM_SetTimeOut

**Description**

Sets the current time-out value for `Receive` operations.

**Declaration**

```geobasic
COM_SetTimeOut ( ByVal nSec AS Integer )
```

**Remarks**

- `nSec` will be interpreted as seconds. The time-out value will be valid until it will be set anew. If set to Zero then `Receive` will not wait until it receives any character(s). Rather it will return immediately after calling. Then handling of input has to be done by the programmer.

**Note**

- The data-link must be active.
- The time-out from the TPS system will be saved and set back when the GeoBASIC program terminates.
This procedure has no effect if it is called on the TPS-Simulator.

### Parameters

<table>
<thead>
<tr>
<th>nSec</th>
<th>in</th>
<th>Negative: Unlimited wait (blocking behaviour).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Zero: Polling of data.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Positive: Wait time in seconds until the execution of Receive times out.</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

Send, Receive

### Error Codes

| RC_OK | Completed successfully. |

### Example

See the example for Receive statement.

---

### 2.13.4 COM_ExecCmd

**Description**

Executes a defined GeoCOM Remote Procedure.

**Declaration**

```plaintext
COM_ExecCmd ( ByVal szPacket AS String255, lStop AS Logical )
```

**Remarks**

The string `szPacket` will be parsed and executed. The format has to follow the text format of a GeoCOM Remote Procedure Call. See the dedicated documentation for further format information. `szPacket` can be a string which has been previously received via the data-link. `lStop` will be set to `TRUE` if and only if the GeoCOM RPC was either a ‘Go Local’ or ‘Stop’ command (RPC numbers 1 and 2). Once a GeoCOM has been recognised then the result will be sent back via the data link (conforming to the RPC format of GeoCOM).
This procedure has no effect if it is called on the TPS-Simulator.

Parameters

| szPacket in | The string that should be interpreted as a Remote procedure call. |
| lStop out   | Will be set to TRUE if and only if the command can be successfully parsed and if it is a ‘Go Local’ (1) command. |

See Also

Receive

Error Codes

| RC_OK          | Completed successfully. |
| RC_INVPARAM    | The string in szPacket does not contain a valid Remote procedure call. |

Example

This example polls the serial line and if it receives a Command then it executes it.

```
DIM iSize AS Integer
DIM sIn AS String255
DIM lStop AS Integer

ON ERROR RESUME NEXT
COM_SetTimeOut (0) ' do not wait
iSize = 255 ' try to get whole string
Receive (sIn, iSize)
IF Err = RC_OK AND iSize > 0 THEN
    COM_ExecCmd( sIn, lStop )
END IF
```
3 TPS 1100 SYSTEM AND GeoBASIC
This chapter describes the relationship of the GeoBASIC interpreter and the TPS system itself.

3.1 Applications on the TPS system.................................................................3-1
3.2 ‘Coding’-Applications on the TPS system .................................................3-2
3.3 Import of the application in a user configuration .......................................3-2
3.4 Events........................................................................................................3-3
3.5 A framework for an application ..............................................................3-3
3.6 Global Return Codes................................................................................3-5

3.1 APPLICATIONS ON THE TPS SYSTEM
The TPS1100 series have the possibility to store and execute external programs. Loading such a program stores it in the internal memory of the theodolite. After loading the program it has to be made accessible for the user. This has to be done by creating a menu item and associate it with a global subroutine. In general this will be done during the finalisation process of loading the program by executing the Install routine of a program. The Install routine is reserved for such purposes and will be called automatically by the loader. After connecting a program to a menu item the program itself can be executed by choosing just this item from the menu.

Additional to this static link of a program to a menu item there are two other possibilities to install a GeoBASIC application on the TPS system:
- Install as ‘Coding’-application
- Import of the application in a user configuration
3.2 ‘CODING’-APPLICATIONS ON THE TPS SYSTEM

With the Coding functionality an application does not need to be connected to a menu item. A Coding program will be invoked when the CODE button has been pressed, hence has not be connected to a menu item. Although the global subroutine Install has to exist because it is called anyway, but, of course, it may be empty.

A GeoBASIC program for the Coding functionality must have the name BasicCodeProgram and the subroutine which is called then must have the name BasicCodeSub.

The TPS system allows to handle not only a GeoBASIC program for the coding functionality. Since there exist three possible locations, the TPS system follows a default ordering rule to invoke one of the programs. First it checks if there is an appropriate set up GeoBASIC program. If yes, it will be executed, otherwise it examines the codelist management if a codelist is selected. If yes, then the codelist will be opened, otherwise the standard coding will be activated.

Note
At any time only one GeoBASIC Coding program can be loaded on the TPS system.
It must have the predefined names, otherwise it will not be recognised.

3.3 IMPORT OF THE APPLICATION IN A USER CONFIGURATION

The TPS1100 series theodolites supports the loading of individual configurations, with it a user can define his own dialogs and menus. If the user imports a GeoBASIC application into the special configuration definition tool, called Customization Tool, he can insert a global subroutine into his own menus and dialogs. To use this possibility for the installation of a GeoBASIC application the menu or dialog item must be defined in a special section named APPINFO at the end of the source code. Refer to chapter 9.3.2 in the user manual for a detailed description of this functionality.
3.4 EVENTS

The configuration functionality of the TPS1100 series is event driven. If a user has defined his own dialogs or menus in a configuration, he can link it with a special event (i.e. the user has pressed the PROG key or the initialisation sequence is finished). If the event occurs, the connected action will be performed, for example the linked dialog or the menu will be displayed. With the routine CSV_SysCall all events defined in the theodolite system software can be generated by a GeoBasic program. For more information about event generation refer to Chapter 9.6 in the user manual.

3.5 A FRAMEWORK FOR AN APPLICATION

In the following chapters standard functions and system functions are described. Almost every such description contains a small example. However, most examples are not ready to compile and run on your LEICA theodolite or PC simulation without setting up a proper program environment.

To keep the examples small, but nevertheless demonstrate some functionality, we now give a general schema for running most of the examples. Just insert the example code at the indicated location, and the program is ready to compile, link, and run. See also the file test.gbs as it is provided as an example in the samples directory.

The necessary environment

- provides the global installation routine Install that links the program into a theodolite menu,
- creates and deletes a text dialog for textual input and output (in this example up to 5 lines can be used)
- provides a function Test that may contain the example program,
- calls the function Test to run the example program, and
- waits for a key press after the function Test has terminated.
PROGRAM TestExample 'program to test the examples
', 'GeoBASIC test frame
', '---------------------------
', 'The example shows a small program frame for the
', 'beginning of a project.
', '(c) Leica AG, CH - Heerbrugg 1999
', '-----------------------------------------------
', 'GLOBAL SUB Install
', '-------
', 'Description
', 'Install it in the program menu.
', 'MMI_CreateMenuItem ( "TestExample", "Main",
', 'MMI_MENU_PROGRAMS, "EXAMPLE" )
', 'END Install
', '-----------------------------------------------
', 'SUB Test
', '--------
', 'INSERT YOUR SAMPLE CODE HERE
', '-----------------------------------------------
', 'END Test
', '-----------------------------------------------
', 'GLOBAL SUB Main
', '------
', 'Description
', 'Small program frame with an empty text dialog.
', 'CONST iLines AS Integer = 5 'display: 5 lines
', 'can be used
', 'DIM iButton AS Integer 'for the button pressed
', 'MMI_CreateTextDialog( iLines, "BASIC",
', '"EXAMPLE", " No Help " )
', 'Test() 'call the test routine
', 'MMI_GetButton( iButton, TRUE ) 'wait for a key press
', 'MMI_DeleteTextDialog()
', 'END Main
', 'END TestExample
3.6 GLOBAL RETURN CODES

In this section the general return codes are briefly described. Note that function specific return codes are found in the function description, and that details on error handling are found in Chapter 2.8.

Global Return Codes.

1. After a standard function or system function is called, the GeoBASIC variable ERR contains its return code. If everything went smoothly, it is set to the predefined constant RC_OK, and normal program execution goes on. However, if there was an error, ERR is set to the corresponding error code. (Therefore, we will rather use the term ERROR CODES for values other than RC_OK.)

2. Every function may have a set of possible error codes defined. If the result of a function is not RC_OK, the variable ERR will contain one of those error codes, describing the function's termination condition.

3. If the error handling is active (ON ERROR GOTO, see Chapter Error Handling), any error code will start the error handler after return from the erroneous function.

4. Usually error codes are grouped by the subsystem to which they are meaningful to (for example TMC_... for measurement error codes like TMC_ANGLE_ERROR, TMC_DIST_ERROR, etc.), but some error codes are generally applicable, for example if there has been a fatal error, an abort, etc.

A summary of all return codes is listed in Appendix F.
Here these general return codes are listed. Note that they will not be mentioned in the description of the standard functions and system functions explicitly unless they have a non-standard or more refined meaning.

<table>
<thead>
<tr>
<th>Predefined Constant</th>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>0</td>
<td>successful termination</td>
</tr>
<tr>
<td>RC_UNDEFINED</td>
<td>1</td>
<td>undefined result, unknown error</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>2</td>
<td>invalid parameter</td>
</tr>
<tr>
<td>RC_IVRESULT</td>
<td>3</td>
<td>invalid result</td>
</tr>
<tr>
<td>RC_FATAL</td>
<td>4</td>
<td>fatal error</td>
</tr>
<tr>
<td>RC_NOT_IMPL</td>
<td>5</td>
<td>not implemented</td>
</tr>
<tr>
<td>RC_TIME_OUT</td>
<td>6</td>
<td>time out</td>
</tr>
<tr>
<td>RC_SET_INCOMPL</td>
<td>7</td>
<td>parameter setup for subsystem is incomplete</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>8</td>
<td>function aborted</td>
</tr>
<tr>
<td>RC_NOMEMORY</td>
<td>9</td>
<td>not enough memory</td>
</tr>
<tr>
<td>RC_NOTINIT</td>
<td>10</td>
<td>subsystem not initialized</td>
</tr>
<tr>
<td>RC_SHUT_DOWN</td>
<td>12</td>
<td>subsystem is down</td>
</tr>
<tr>
<td>RC_SYSBUSY</td>
<td>13</td>
<td>system busy</td>
</tr>
<tr>
<td>RC_HWFAILURE</td>
<td>14</td>
<td>hardware failure (fatal)</td>
</tr>
<tr>
<td>RC_ABORT_APPL</td>
<td>15</td>
<td>Abort Application (Shift-Esc)</td>
</tr>
<tr>
<td>RC_LOW_POWER</td>
<td>16</td>
<td>Insufficient power level</td>
</tr>
<tr>
<td>RC_INVVERSION</td>
<td>17</td>
<td>Invalid version of file, ...</td>
</tr>
<tr>
<td>RC_BATT_EMPTY</td>
<td>18</td>
<td>Battery empty</td>
</tr>
<tr>
<td>RC_NO_EVENT</td>
<td>20</td>
<td>no event pending</td>
</tr>
<tr>
<td>RC_OUT_OF_TEMP</td>
<td>21</td>
<td>out of temperature range</td>
</tr>
<tr>
<td>RC_INSTRUMENT_TILT</td>
<td>22</td>
<td>instrument tilting out of range</td>
</tr>
<tr>
<td>RC_COM_SETTING</td>
<td>23</td>
<td>communication error</td>
</tr>
<tr>
<td>RC_NO_ACTION</td>
<td>24</td>
<td>RC_TYPE Input 'do no action'</td>
</tr>
<tr>
<td>RC_SLEEP_MODE</td>
<td>25</td>
<td>Instrument run into sleep mode</td>
</tr>
</tbody>
</table>
4 REMARKS ON THE DESCRIPTION

In the following two chapters all functions known to GeoBASIC are described. In this chapter you will read how this description is organised.

4.1 Structure of the Description ................................................. 4-2
  4.1.1 The whole system ...................................................... 4-2
  4.1.2 The Sections ............................................................. 4-2
  4.1.3 The function/procedure descriptions .......................... 4-5

4.2 Example of a Description .................................................. 4-8
  4.2.1 TMC_GetAngle ......................................................... 4-8
4.1 STRUCTURE OF THE DESCRIPTION

We describe the structure of the system top-down:
1. first the system as a whole,
2. then we describe the common parts of all sections,
3. and at last a single function/procedure description.

4.1.1 The whole system

The description of the whole system is split up into several sections, each describing:
- GeoBASIC built-in functions (such as Section Standard functions),
- extensions to GeoBASIC (such as Section Geodesy Mathematics), or a
- theodolite subsystem (such as the whole Chapter System Functions, for example Section MMI Functions describing the man machine interface).

4.1.2 The Sections

A section description consists of (at most) four parts.
1. The name of the section.
2. Lists of types, functions, procedures, and constants defined in the section.
3. Definition of types.
4. Declaration of functions, procedures, and constants.

We now explain these four parts in more detail.

Note: The identifiers in the examples of this section are stylised. Section 4.2 shows a “real” example, annotated with some explanations given in this section.
4. Remarks on the Description

4.1.2.1 Name of a section

The name of a section describes the section as a whole. It can be considered the smallest class under which all the types, functions, procedures, and constants can be grouped. For example,

6.1 MMI FUNCTIONS

4.1.2.2 Lists of identifiers

Then, lists of all identifiers that are defined in the section are given. First for types, then for functions/procedures, and at last for the constants. All lists are sorted by name. The schema is as follows.

<table>
<thead>
<tr>
<th>Summarising Lists of Types, Procedures, and Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types</td>
</tr>
<tr>
<td>type name</td>
</tr>
<tr>
<td>Some_New_Type</td>
</tr>
<tr>
<td>Some_Other_New_Type</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>Brief description of the type.</td>
</tr>
<tr>
<td>Brief description of the type.</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

| Functions                                              |
| function name                                          |
| Some_New_Function                                     |
| ...                                                   |
| description                                           |
| Brief description of the function.                    |
| ...                                                   |

| Procedures                                             |
| procedure name                                         |
| Some_New_Procedure                                    |
| ...                                                   |
| description                                           |
| Brief description of the procedure.                   |
| ...                                                   |
4. Remarks on the Description

### Constants

<table>
<thead>
<tr>
<th>constant name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some_New_Constant</td>
<td>Brief description of the constant.</td>
</tr>
</tbody>
</table>

### 4.1.2.3 Type definitions

After the lists, the type definitions are given. In the example (below) it can be seen that first the new type name and its intended usage is mentioned. In the description part, the type will be described in words. Then its definition follows, giving every component its type and a more detailed description.

**New_Type** - Here stands what it is used for

**Description** Here the new type is described.

```plaintext
TYPE New_Type
    Component1 ItsType  description of Component1
    Component2 ItsType  description of Component2
    Component3 ItsType  description of Component3
END New_Type
```

### 4.1.2.4 Function/procedure description

Then the function/procedure descriptions follow. (See Section 4.1.3 below.)

**Note** Not every section has all these four components. Only those parts will be given that actually have entries. (Empty ones are omitted.)
4.1.3 The function/procedure descriptions

We treat functions and procedures together since they only differ in the return value (procedures do not return a value, whereas functions do).

A function/procedure description consists of (at most) eight parts.

1. The function/procedure name.
2. The description.
3. The declaration.
4. Remarks.
5. A detailed parameter description.
7. Cross reference (see also).
8. An example.

Details:

♦ Ad 1) First, the function/procedure name is given. For example,

```
EXAMPLE_SomeFunction
```

♦ Ad 2) Then a description follows, describing the function’s/procedure’s task. For example,

```
Description       Here the function/procedure is described.
```
Ad 3, 4) Afterwards the interface declaration and remarks are given. A note may supplement the presentation. Additional a remark for the simulator may be given which is valid only for the TPS simulator. For example,

**Declaration**

```basic
EXAMPLE_Some_FUNCTION(
  ByVal dParameter AS double,
  sParameter AS String255,
  iVarParameter AS Integer )
```

**Remarks**

Remarks concerning `EXAMPLE_Some_FUNCTION`.

**Note**

Here come some important notes.

**TPS_Sim**

Has no effect.

Ad 5, 6) Now more details of the interface are described: the parameters and the error codes (see also Section Global Return Codes). While doing so, also predefined constants (for parameter values or error codes) are mentioned. For example,

**Parameters**

- `dParameter` in: description of `dParameter`
- `sParameter` in: description of `sParameter`
- `iVarParameter` out: description of `iVarParameter`; possible values for `iVarParameter`:
  - value 1: meaning 1
  - value 2: meaning 2
  - ...

**Error Codes**

- `ErrorCode1`: description of `ErrorCode1`
- `ErrorCode2`: description of `ErrorCode2`
- ...

...
Ad 7, 8) In the end a cross reference and an example of the use of the defined function is given (see also Section Putting the examples to work). For example,

**See Also**
- SomeOtherFunction1
- SomeOtherFunction2
- Some other chapter in the reference

**Example**
Description of the example.
Example source code.

**Note**
Not every description has *all* these components. Only those parts will be given that actually have entries. (Empty ones are omitted.)
The following picture in Section 4.2 shows an annotated example of a procedure description.

### 4.2 EXAMPLE OF A DESCRIPTION

#### 4.2.1 TMC_GetAngle

**Description**
Measure angles.

**Declaration**
```plaintext
TMC_GetAngle( Angle AS TMC_ANGLE, iReturnCode AS Integer )
```

**Remarks**
The function measures the horizontal and vertical angle.

**Note**
The measure program must have been started.

**Parameters**
- **Angle**
  - Description: out result of measuring the angle
  - Explanation: return code, see Error Codes
- **iReturnCode**
  - Description: out
  - Explanation: return code, see Error Codes

**See Also**
TMC_DoMeasure

**Return Codes**
- **RC_OK**
  - Description: angle OK
  - Explanation: no inclination measuring possible (no results)
- **TMC_ANGLE_ERROR**

**Example**
Read the currently valid angle.
```plaintext
DIM Angle AS Double
DIM RetCode AS Integer
TMC_GetAngle( Angle, RetCode )
```
5. **STANDARD FUNCTIONS**

5.1 Numeric to numeric

5.1.1 Abs - Absolute value
5.1.2 Int - Integer part
5.1.3 Round - Round
5.1.4 Sgn - Sign

5.2 String to numeric

5.2.1 Asc - ASCII code of a character
5.2.2 InStr - Index of a substring inside a string
5.2.3 Len - Length of a string
5.2.4 Val - Numerical value of a string

5.3 Numeric to string

5.3.1 Chr$ - Character from ASCII code
5.3.2 String$ - String from fill character
5.3.3 Str$ - String from a numerical value
5.3.4 SFormat Function

5.4 String to string

5.4.1 UCase$ - Change to upper case
5.4.2 LCase$ - Change to lower case
5.4.3 LTrim$ - Trim blanks from the left
5.4.4 RTrim$ - Trim blanks from the right
5.4.5 Left$ - Left substring
5.4.6 Right$ - Right substring
5.4.7 Mid$ - Substring anywhere

5.5 Standard Mathematics Functions

5.5.1 Summarising List of Mathematics Functions
5.5.2 Remark on the Conversion of Angles
5.5.3 Atn Function
5.5.4 Cos Function
5.5.5 Exp Function
5.5.6 Log Function............................................................................... 5-19
5.5.7 Sin Function............................................................................... 5-20
5.5.8 Sqr Function............................................................................... 5-21
5.5.9 Tan Function............................................................................... 5-22
5.5.10 Rnd Function............................................................................ 5-23
5.5.11 SRnd Function........................................................................... 5-24

5.6 Geodesy Mathematics........................................................................... 5-25
5.6.1 Summarising Lists of GM Types and Procedures....................... 5-25
5.6.2 GeoMath Structures .................................................................... 5-28
5.6.3 GM_CalcAreaOfCoord ............................................................... 5-32
5.6.4 GM_CalcAreaOfMeas ................................................................. 5-35
5.6.5 GM_CalcAziAndDist.................................................................. 5-37
5.6.6 GM_CalcCenterAndRadius .......................................................... 5-39
5.6.7 GM_CalcClothCoord .................................................................. 5-40
5.6.8 GM_CalcCoord .......................................................................... 5-41
5.6.9 GM_CalcDistPointCircle ............................................................ 5-42
5.6.10 GM_CalcDistPointCloth ............................................................. 5-43
5.6.11 GM_CalcDistPointLine............................................................... 5-44
5.6.12 GM_CalcHiddenPointObservation ........................................... 5-46
5.6.13 GM_CalcIntersectionCircleCircle ........................................... 5-47
5.6.14 GM_CalcIntersectionLineCircle ................................................. 5-49
5.6.15 GM_CalcIntersectionLineLine ................................................... 5-50
5.6.16 GM_CalcMean ........................................................................... 5-52
5.6.17 GM_CalcMeanOfHz ................................................................. 5-54
5.6.18 GM_CalcMedianOfHz ............................................................... 5-55
5.6.19 GM_CalcOrientationOfHz .......................................................... 5-57
5.6.20 GM_CalcPointInLine ................................................................. 5-59
5.6.21 GM_CalcPointInCircle ............................................................... 5-60
5.6.22 GM_CalcTriangle ...................................................................... 5-61
5.6.23 GM_CalcVAndSlope ................................................................. 5-63
5.6.24 GM_ConvertAngle .................................................................... 5-64
5.6.25 GM_ConvertDecSexa ................................................................. 5-65
All but one of the standard functions available in GeoBASIC belong to one of four groups: numeric to numeric, string to numeric, numeric to string, and string to string.

Note: Where string subscripts are used, indexing always starts at 1, as for arrays in GeoBASIC.

### 5.1 NUMERIC TO NUMERIC

#### 5.1.1 Abs - Absolute value

\[ \text{Abs} (X) \] yields the absolute value of the expression \( X \). The expression must be of a numeric type (\text{Integer, Double} or its variations). The result is of the same type as \( X \).
Examples:
Abs (-4.6) -> 4.6
Abs (5) -> 5

5.1.2 Int - Integer part

Int (X) yields the integer part of the expression X. The expression must be of a numeric type (Integer, Double or its variations). The result is of type Integer.

Examples:
Int (5.2) -> 5
Int (5.8) -> 5
Int (-5.5) -> -5

5.1.3 Round - Round

Round (X) yields the value of the expression X rounded to the nearest integer. Values halfway between two integers are always rounded away from zero. The expression must be of a numeric type (Integer, Double or its variations). The result is of type Integer.

Examples:
Round (5.2) -> 5
Round (5.8) -> 6
Round (5.5) -> 6
Round (6.5) -> 7
Round (-5.2) -> -5
Round (-5.8) -> -6
Round (-5.5) -> -6
Round (-6.5) -> -7

5.1.4 Sgn - Sign

Sgn (X) yields the sign of the value of the expression X. Positive values yield +1, negative values -1, and a zero value yields 0. The expression must be of a numeric type (Integer, Double or its variations). The result is of type Integer.
Examples:

Sgn (5.2) -> 1
Sgn (-4) -> -1
Sgn (0) -> 0

5.2 STRING TO NUMERIC

5.2.1 Asc - ASCII code of a character

Asc(S) yields the value of the first (or only) character of the string expression S. The result is of type Integer.

Examples:

Asc ("*") -> 42
Asc ("Alpha") -> 65

5.2.2 InStr - Index of a substring inside a string

InStr(S1,S2) looks for the substring S2 inside the string S1 and yields either the index of the first character where S2 starts in S1, or 0 if S2 cannot be found. Upper and lower case characters are considered distinct. Both parameters must be string expressions. The result is of type Integer.

Examples:

InStr ("Bananas", "na") -> 3
InStr ("Bananas", "nas") -> 5
InStr ("Bananas", "Na") -> 0

InStr(K,S1,S2) works like InStr(S1,S2) but looks for S2 only at the K-th character and beyond. S1 and S2 must be string expressions, K must be an expression of type Integer. The result is of type Integer.
Examples:

InStr (3, "Bananas", "na") -> 3
InStr (4, "Bananas", "na") -> 5
InStr (6, "Bananas", "na") -> 0

5.2.3 Len - Length of a string

Len(S) yields the length of the string expression S, i.e. the number of characters in S (not counting the terminating zero). The result is of type Integer.

Examples:

Len ("Bananas") -> 7
Len ("A + B = ") -> 8
Len (**) -> 0

5.2.4 Val - Numerical value of a string

Val(S) yields the value of the string expression S interpreted as a numeric constant. S may contain leading blanks, one sign, a decimal point, and a power of ten part with or without sign. Blanks within the number are not allowed. Interpretation ends with the first character that cannot be part of a legal GeoBASIC numeric constant representation. If S does not represent a number, the result of Val(S) is 0. The result is of always of type Double.

Examples:

Val ("1.5") -> 1.5
Val (" +7.3e-4") -> 0.00073
Val ("-2E5xyz") -> -200000.0
Val ("X") -> 0.0
Val (" -3") -> -3.0
5.3 NUMERIC TO STRING

5.3.1 Chr$ - Character from ASCII code

Chr$(N) yields a string of length one, consisting of the character whose ASCII code is the value of the expression N. The result is of type string * 1.

Example:

Chr$ (42) -> "*"

5.3.2 String$ - String from fill character

String$(N,X) yields a string consisting of N identical characters. This character is either the first character of the string expression X, or the character whose ASCII code is the value of the integer expression X. The result is of type String.

Examples:

String$ (6, 42) -> "******"
String$ (5, "/") -> "/////
String$ (4, "abc") -> "aaaa"

5.3.3 Str$ - String from a numerical value

Str$(X) yields the string representing (in a fixed format) the value of the expression X. The expression must be of a numeric type (Integer, Double or its variations). The result is of type string * n, where n is the length of the resulting string.

Examples:

Str$ (6) -> "6"
Str$ (-5.88) -> "-5.88"
Str$ (0.00000042) -> "4.2e-07"
5.3.4 SFormat Function

**Description**
Generate a string using a value according to a C-format specification.

**Syntax**

```plaintext
SFormat( ByVal sFormatStr AS String,
          ByVal iValue AS Integer ) AS String

SFormat( ByVal sFormatStr AS String,
          ByVal dValue AS Double ) AS String

SFormat( ByVal sFormatStr AS String,
          ByVal lValue AS Logical ) AS String
```

**Remarks**
The first argument is an input parameter and must contain a valid format specification for value. It has to follow the general rules of GeoBASIC strings and may be of any string type.

The second argument value can be any valid numeric (integer, double) or logical expression.

A double value larger than 10\(^{256}\) with ",%f" formatting will result in the string "xxxxxxxxxxxx", since the value can be transformed to a maximum of 250 characters only.

**Note**
The format string and the value argument must match. sFormatStr255 may contain only one ",%". More than one ",%" are not allowed and may lead to unpredictable behaviour.

Other than the here explained formatting sequences are not allowed and may lead to unpredictable behaviour.

The computed result cannot be larger than 255 characters long in any case.
General format specification:

"%[flags][width][.precision][l]type"

flags - left justify (default: right justify)

+ prefix the output value with a sign (+/-) (default: sign only for neg. numbers)

0 if width is prefixed with "0", zeros are added until the minimum width is reached. If specified with integer type, it is ignored (default: no padding)

blank if positive, instead of sign (default: no padding blank for sign)

# when used with e, E or f format type, the flag forces the output value to contain a decimal point in all cases; for g, G format type, it prevents in addition the truncation of trailing zeros (default: decimal point appears only if digits follow, for g, G trailing zeros are truncated).

width Optional number that specifies the minimum number of characters printed. If the generated string is bigger then all characters are printed.

precision Optional number that specifies the minimum number of digits printed for integer values. Can cause truncation of output.
<table>
<thead>
<tr>
<th>Type</th>
<th>Integer types</th>
<th>output format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>ld, li</td>
<td>signed decimal long integer</td>
</tr>
<tr>
<td></td>
<td>lu</td>
<td>unsigned decimal long integer</td>
</tr>
<tr>
<td></td>
<td>lo</td>
<td>unsigned octal long integer</td>
</tr>
<tr>
<td></td>
<td>lx</td>
<td>unsigned hexadecimal long integer, using &quot;abcdef&quot;</td>
</tr>
<tr>
<td></td>
<td>lX</td>
<td>unsigned hexadecimal long integer, using &quot;ABCDEF&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Double types</th>
<th>output format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>lf</td>
<td>signed value having the form [-]dddd.dddd, where dddd is one or more digits. Only values in between ± 10^{250} can be formatted.</td>
</tr>
<tr>
<td></td>
<td>le</td>
<td>signed value having the form [-]d.dddd e [sign]ddd, where d is a single digit, ddd are exactly 3 digits.</td>
</tr>
<tr>
<td></td>
<td>lE</td>
<td>identical to le, exponent character E instead of e</td>
</tr>
<tr>
<td></td>
<td>lg</td>
<td>signed value printed in f or e format, whichever is more compact for the given value and precision</td>
</tr>
<tr>
<td></td>
<td>lG</td>
<td>identical to &quot;lg&quot;, except that lG, rather than lg, introduces the exponent (where appropriate)</td>
</tr>
<tr>
<td><strong>Data Type</strong> (value)</td>
<td><strong>Format Specification</strong></td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| Integer             | any format specification that can be used for a 4-byte value (type long in ANSI-C), see description above For more detailed descriptions, please refer to the format spec. in the description of the ANSI-C-function "%ld"
| Double              | 8-byte value (double in ANSI-C), see description above "%lf" is recommended. |
| Logical             | the following two formats are implemented:  
  - "%s": Generate a string ("T" / "F")  
  - "%d": Generate a number (1 / 0) |

**See Also** ANSI-C function `sprintf` format specifications.
Example

The example uses the SFormat function to generate strings.

```plaintext
sFormatVal = SFormat( "Double = %lf", 3.5e-4 )
  ' sFormatVal -> "Double = 0.000350"

sFormatVal = SFormat( "Integer = %ld", -10 )
  ' sFormatVal -> "Integer = -10"

sFormatVal = SFormat( "Logical = %s", TRUE )
  ' sFormatVal -> "Logical = T"

sFormatVal = SFormat( "Hex = %lX", 15 )
  ' sFormatVal -> "Hex = F"

sFormatVal = SFormat( "Octal = %lo", 15 )
  ' sFormatVal -> "Octal = 17"

sFormatVal = SFormat( "Double=%.6lf",1111.12345)
  ' sFormatVal -> "Double = 1111.123450"

sFormatVal = SFormat("Double=%+.6lf",1111.12345)
  ' sFormatVal -> "Double=+1111.123450"
```

5.4 STRING TO STRING

5.4.1 UCase$ - Change to upper case

UCase$(S) yields the string expression $S$ with all lower case letters "a" to "z" replaced by their upper case. Any other character is unchanged. The result is of type string * n, where n is the length of $S$.

Examples:

```plaintext
UCase$ ("Start") -> "START"
UCase$ ("kürzer/länger?") -> "KÜRZER/LÄNGER?"
(umlaut unchanged!)
```
5.4.2  LCase$ - Change to lower case

LCase$(S) yields the string expression  $ with all upper case letters "A" to "Z" replaced by their lower case. Any other character is unchanged. The result is of type string * n, where n is the length of S.

Examples:

LCase$ ("START") -> "start"
LCase$ ("GRÖSSER?") -> "grösse?"  (umlaut unchanged!)

5.4.3  LTrim$ - Trim blanks from the left

LTrim$(S) yields the value of the string expression  $ with all leading blanks removed. The result is of type string * n, where n = (length of S) - (number of blanks).

Example:

LTrim$ (" Stop ") -> "Stop  ">

5.4.4  RTrim$ - Trim blanks from the right

RTrim$(S) yields the value of the string expression  $ with all trailing blanks removed. The result is of type string * n, where n = (length of S) - (number of blanks).

Example:

RTrim$ (" Stop ") -> " Stop"
5.4.5 Left$ - Left substring

Left$(S,N) yields the substring consisting of the first N characters of the string expression S. N must be an expression of type Integer. The result is of type string * N.

Example:

Left$ ("Railwaytrack", 4) -> "Rail"

5.4.6 Right$ - Right substring

Right$(S,N) yields the substring consisting of the last N characters of the string expression S. N must be an expression of type Integer. The result is of type string * N.

Example:

Right$ ("Railwaytrack", 5) -> "track"

5.4.7 Mid$ - Substring anywhere

Mid$(S,K,N) yields the substring consisting of N characters of the string expression S, starting at the K-th character. K and N must be expressions of type Integer. The length of the resulting string is N. If parameter N is omitted, the substring runs to the end of S.

Examples:

Mid$ ("Railwaytrack", 5, 3) -> "way"
Mid$ ("Railwaytrack", 9) -> "rack"

Mid$ can also be used to assign a character or a substring to another string at a certain place. With Mid$(S,K,N) = T single characters of a string can be set or replaced. If the length of T is higher than N, only the first N characters of T are set in S. Is parameter N omitted, the whole substring T will be inserted in S (if the length of S this allows).
Examples:

s = "123456789"
Mid$ (s, 2, 3) = "abcde"
' 3 characters (2..4) are replaced
' s -> "1abc56789"

s = "123456789"
Mid$ (s, 2, 7) = "abcde"
' 5 characters (2..6) are replaced
' s -> "1abcde789"

s = "123456789"
Mid$ (s, 2) = "abcde"
' 5 characters (2..6) are replaced
' s -> "1abcde789"
5.5 STANDARD MATHEMATICS FUNCTIONS

5.5.1 Summarising List of Mathematics Functions

<table>
<thead>
<tr>
<th>function name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atn</td>
<td>Returns the arcs tangent of a number.</td>
</tr>
<tr>
<td>Cos</td>
<td>Returns the cosine of an angle.</td>
</tr>
<tr>
<td>Exp</td>
<td>Returns e (the base of natural logarithms) raised to a power.</td>
</tr>
<tr>
<td>Log</td>
<td>Returns the natural logarithm of a number.</td>
</tr>
<tr>
<td>Rnd</td>
<td>Returns a random number in a user-defined value-range.</td>
</tr>
<tr>
<td>Sin</td>
<td>Returns the sine of an angle.</td>
</tr>
<tr>
<td>Sqr</td>
<td>Returns the square root of a number.</td>
</tr>
<tr>
<td>SRnd</td>
<td>Initialises the random-number generator.</td>
</tr>
<tr>
<td>Tan</td>
<td>Returns the tangent of an angle.</td>
</tr>
</tbody>
</table>

5.5.2 Remark on the Conversion of Angles

GeoBASIC computes in SI units, for angles this means in radians. The conversion from grad to radians and vice versa is described next.

Let the variable \( \text{halfCircle} \) be 200 gon. (For decimal degrees, \( \text{halfCircle} \) is 180 degrees. The value in the variable \( \text{grad} \) must be in the corresponding degree units.)

\[
\text{radians} = \frac{\text{grad} \times \pi}{\text{halfCircle}} \quad \text{grad} = \frac{\text{radians} \times \text{halfCircle}}{\pi}
\]

Another way to convert angles is to use the geodesy mathematics conversion function. For example to convert \( \text{dDegree} \) decimal degrees to radians (the result will be in \( \text{dRadian} \)), use the following function call. (See section 5.6.24 for a detailed description.)

\[
\text{GM_ConvertAngle( GM_DEGREE_DEZ, dDegree, GM_RADIANS, dRadian, iReturnCode )}
\]

See Also Geodesy Mathematical Formulas: Section on "Conversion of Angles".
5.5.3 Atn Function

Description
Returns the arcs tangent of a number.

Declaration
Atn( dAngle AS Double ) AS Double

Remarks
The argument dAngle can be any valid numeric expression. The return type of Atn is Double.

The Atn function takes the ratio (a floating point number) of two sides of a right triangle and returns the corresponding angle. The ratio is the length of the side opposite to the angle divided by the length of the side adjacent to the angle. (The hypotenuse is not involved.)

The result's unit is radians. It is in the floating point range $-\frac{\pi}{2}$ to $\frac{\pi}{2}$.

Note
Atn is the inverse trigonometric function of Tan. Do not confuse arcus tangent with the cotangent, which is simply the multiplicative inverse of a tangent (i.e. $\frac{1}{\tan}$).

See Also
Cos, Sin, Tan
Remark on the Conversion of Angles (5.5.2)

Example
The example uses Atn to compute Pi. By definition, Atn(1) is $\frac{\pi}{4}$ radians (that equals 50 grad or 45 degrees).

DIM dMyPi AS Double ' Declare variables.
dMyPi = 4 * Atn(1) ' Calculate Pi.
WRITE "Pi is equal to " + str$(dMyPi)
5.5.4 Cos Function

Description
Returns the cosine of an angle.

Declaration
Cos( dAngle AS Double ) AS Double

Remarks
The argument dAngle can be any valid numeric expression measured in radians. The return type of Cos is Double.

The Cos function takes an angle and returns the ratio of two sides of a right triangle: of the length of the side adjacent to the angle to the length of the hypotenuse.

The result is in the floating point range -1.0 to 1.0.

See Also
Atn
Sin
Tan
Remark on the Conversion of Angles (5.5.2)

Example
The example uses Cos to calculate the cosine of an angle with a user-specified number of degrees.

```
DIM dDegrees AS Double 'Declare variables
DIM dRadians AS Double

dDegrees = 45.0
dRadians = dDegrees * (Pi / 180.0) 'Convert to radians.
WRITE "The cosine of a " + Str$(dDegrees) + " degree angle is " + Str$(Cos(dRadians))
```

5.5.5 Exp Function

Description
Returns e (the base of natural logarithms) raised to a power.

Declaration
Exp( dPower AS Double ) AS Double

Remarks
The argument dPower can be any valid numeric expression. The return type of Exp is Double.
e is the exponential constant (base of natural logarithms), with numerical value \( e = e^1 = \text{Exp}(1) = 2.71828\ldots \)

| Note | Exp is the inverse function of the Log function and is sometimes referred to as the antilogarithm. |

**See Also**

Log.

**Example**
The example uses Exp to compute the value of e. \( \text{Exp}(1) \) is \( e \) raised to the power of 1.

```basics
' \text{Exp}(x) \text{ is } e^x \text{ so } \text{Exp}(1) \text{ is } e^1 \text{ or } e.
DIM dValueOfE AS Double  ' Declare variables.
dValueOfE = \text{Exp}(1)  ' Calculate value of e.
WRITE "The value of e is " + Str$(dValueOfE)
```

**5.5.6 Log Function**

**Description**
Returns the natural logarithm of a number.

**Declaration**

```
Log( dNumber AS Double ) AS Double
```

**Remarks**
The argument \( dNumber \) can be any valid numeric expression that denotes a value greater than zero. The return type of \( \text{Log} \) function is Double.

The natural logarithm is the logarithm to the base \( e \). \( e \) is the exponential constant (base of natural logarithms), with numerical value \( e = 2.71828\ldots \)

You can calculate base-n logarithms (logarithms to the base \( n \)) for any number \( x \) by dividing the natural logarithm of \( x \) by the natural logarithm of \( n \) as follows:

\[
\text{Log}_n(x) = \frac{\text{Log}(x)}{\text{Log}(n)}
\]

It holds that \( n^{\text{Log}_n(x)} = x \).

The following example illustrates a function that calculates base-10 logarithms:
Function Log10( dX AS Double ) As Double
    Log10 = Log(dX) / Log(10)
End Log10

The more general function LogN takes the base as an additional argument:
Function LogN( iBase AS Integer, dX AS Double ) As Double
    LogN = Log(dX) / Log(iBase)
End LogN

See Also
Exp

Example
The example calculates the value of e, then uses the Log function to calculate the natural logarithm of e to the third power.
DIM dValueOfE AS Double  ' Declare variables.
dValueOfE = Exp(1)  ' Declare variables.
WRITE Str$(Log(dValueOfE ^ 3))

5.5.7 Sin Function

Description
Returns the sine of an angle.

Declaration
Sin( dAngle AS Double ) AS Double

Remarks
The argument dAngle can be any valid numeric expression measured in radians. The return type of Sin is Double.

The Sin function takes an angle and returns the ratio of two sides of a right triangle: of the length of the side opposite to the angle to the length of the hypotenuse.

The result is in the floating point range -1.0 to 1.0.

See Also
Atn
Cos
Tan
Remark on the Conversion of Angles (5.5.2)
Example  In the example the user can enter a slope distance and a zenith angle. Out of this the horizontal length is computed and displayed.

```
DIM dSlopeDist AS Distance 'slop distance
DIM dZenith AS Angle 'zenith angle
DIM dHorizDist AS Distance 'computed horizontal distance
DIM iButton AS Integer 'button id

PrintStr( 0, 0, "Slope dist.:" )
InputVal( 19, 0, MMI_FFORMAT_DISTANCE, 8, 2, dSlopeDist, TRUE, 0.0, 10000.0, iButton )
PrintStr( 0, 1, "Zenith angle:" )
InputVal( 19, 1, MMI_FFORMAT_ANGLE, 8, 3, dZenith, TRUE, 0.0, 2*Pi, iButton )

dHorizDist = dSlopeDist * Sin( dZenith )
PrintStr( 0, 2, "Horiz. Dist:" )
PrintVal( 19, 2, 8, 2, dHorizDist, TRUE, MMI_DIM_ON )
```

5.5.8 Sqr Function

**Description**  Returns the square root of a number.

**Declaration**  
```
Sqr( dNumber AS Double ) AS Double
```

**Remarks**  The argument dNumber can be any valid numeric expression that denotes a value greater than or equal to zero. The return type of Sqr is Double.

**Example**  The example uses Sqr to calculate the square root of a user-supplied number.
DIM dNumber AS Double  ' Declare variables.

dNumber = 2.0
IF dNumber < 0.0 THEN
    WRITE "Cannot determine the square root " +
    "of a negative number!"
ELSE
    WRITE "The square root of " + Str$(dNumber) +
    " is " + Str$(Sqr(dNumber)) + "."
END IF

5.5.9 Tan Function

Description    Returns the tangent of an angle.

Declaration    Tan( dAngle AS Double ) AS Double

Remarks        The argument dAngle can be any valid numeric expression measured in radians. The return type of Tan is Double.

The Tan function takes an angle and returns the ratio of two sides of a right triangle: of the length of the side opposite the angle to the length of the side adjacent to the angle.

Mind that Tan is not defined for dAngle = $\pi \over 2$ and 

\[
dAngle = -\pi \over 2.
\]

See Also    Atn
            Cos
            Sin

Remark on the Conversion of Angles (5.5.2).

Example     The example uses Tan to calculate the tangent of an angle with a user-specified number of degrees.
DIM dDegrees AS Double ' Declare variables.
DIM dRadians AS Double

dDegrees = 45.0
dRadians = dDegrees * (Pi / 180) ' Convert to radians.
Write("The tangent of a " + Str$(dDegrees) + 
" degree angle is " + Str$( Tan(dRadians)) )

5.5.10 Rnd Function

Description       Returns a random number in a user-defined value-range.

Declaration       Rnd( dNumber AS Double ) AS Double
                   Rnd( iNumber AS Integer ) AS Integer

Remarks           The argument dNumber can be any valid numeric expression.
The Rnd function returns a pseudo random value in the range 0 to dNumber. The SRnd function can be used to seed the pseudo random number generator before calling Rnd.

Note              The same random-number sequence is generated each time the program runs. To have the program generate a different random-number sequence each time it is run, use the SRnd function to initialise the random-number generator before Rnd is called.

See Also          SRnd

Example           The example uses the Rnd function to generate 20 random values in the range from 0 to 10. Each time this program runs, the user can initialise the random-number generator by using SRnd to give a new seed value.
Sub Rnd_Example()
  DIM iStart AS Integer
  DIM iCnt AS Integer
  DIM DateTime AS Date_Time_Type

  CSV_GetDateTime( DateTime )
  iStart = DateTime.Time.Second
  iStart = SRnd( iStart ) 'seed random number
                      ' generator

  FOR iCnt = 1 TO 20
    Write( Str$(Rnd(10)) ) 'generate 20
            ' random values
  NEXT
END Rnd_Example

### 5.5.11 SRnd Function

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>Initialises the random-number generator.</th>
</tr>
</thead>
</table>
| **Declaration** | $SRnd(\text{dNumber AS Double})$ AS Double  
                      $SRnd(\text{iNumber AS Integer})$ AS Integer |
| **Remarks**     | The argument number can be any valid numeric expression, both $\text{Integer}$ and $\text{Double}$ works. $\text{iNumber}$ (or $\text{dNumber}$) is used to initialise the pseudo random-number generator by giving it a new seed value. If $SRnd$ is not used, the $\text{Rnd}$ function returns the same sequence of random numbers every time the program runs. To have the sequence of random numbers change each time the program is run, place the $\text{SRnd}$ function at the beginning of the program. The $\text{SRnd}$-function returns the value of its argument unchanged. |
| **See Also**    | $\text{Rnd}$ |
| **Example**     | See $\text{Rnd}$ function. |
5.6 GEODESY MATHEMATICS

5.6.1 Summarising Lists of GM Types and Procedures

5.6.1.5 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM_Transform_Param_Type</td>
<td>Transformation parameters.</td>
</tr>
<tr>
<td>GM_Circle_Type</td>
<td>Definition of a circle.</td>
</tr>
<tr>
<td>GM_Excenter_Elems_Type</td>
<td>Elements of the eccentric observation.</td>
</tr>
<tr>
<td>GM_Line_Type</td>
<td>Definition of a line.</td>
</tr>
<tr>
<td>GM_Mean_StdDev_Type</td>
<td>Average, middle error of average, and middle error of any observation.</td>
</tr>
<tr>
<td>GM_Measurements_Type</td>
<td>Structure used for measurement (polar coordinates).</td>
</tr>
<tr>
<td>GM_Point_Type</td>
<td>Definition of a point.</td>
</tr>
<tr>
<td>GM_QXX_Matrix_Type</td>
<td>Coefficients of the cofactor matrix of the unknown.</td>
</tr>
<tr>
<td>GM_Triangle_Accuracy_Type</td>
<td>Accuracy of angle and side of the triangle.</td>
</tr>
<tr>
<td>GM_Triangle_Values_Type</td>
<td>Sides and angles of a triangle.</td>
</tr>
</tbody>
</table>

5.6.1.6 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM_AdjustAngleFromZeroToTwoPi</td>
<td>Normalise angle to [0, 2*Pi].</td>
</tr>
<tr>
<td>GM_AngleFromThreePoints</td>
<td>Calculate enclosed angle from three points.</td>
</tr>
<tr>
<td>GM_CalcAreaOfCoord</td>
<td>Calculation of area result from measurement.</td>
</tr>
<tr>
<td>GM_CalcAreaOfMeas</td>
<td>Calculation of area result from measurement.</td>
</tr>
<tr>
<td>GM_CalcAziZenAndDist</td>
<td>Convert a point given in Cartesian coordinates to polar coordinates.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GM_CalcCenterAndRadius</td>
<td>Calculation of centre coordinate and radius result from 3 points.</td>
</tr>
<tr>
<td>GM_CalcClothCoord</td>
<td>Calculation of coordinate on the unitary clothoids (A=1).</td>
</tr>
<tr>
<td>GM_CalcAziAndDist</td>
<td>Calculation of azimuth and distance result from coordinate.</td>
</tr>
<tr>
<td>GM_CalcCoord</td>
<td>Calculation of coordinate result from azimuth and distance.</td>
</tr>
<tr>
<td>GM_CalcDistPointCircle</td>
<td>Calculation of the distance point to circle and the base point of plumb line.</td>
</tr>
<tr>
<td>GM_CalcDistPointCloth</td>
<td>Calculation of the distance point - clothoide and the base point of plumb line.</td>
</tr>
<tr>
<td>GM_CalcDistPointLine</td>
<td>Calculation of the distance point - line and the base point of plumb line.</td>
</tr>
<tr>
<td>GM_CalcHiddenPointObservation</td>
<td>Calculated measurement to the hidden point.</td>
</tr>
<tr>
<td>GM_CalcIntersectionCircleCircle</td>
<td>Calculation of intersection-point circle - circle.</td>
</tr>
<tr>
<td>GM_CalcIntersectionLineCircle</td>
<td>Calculation of intersection-point line - circle.</td>
</tr>
<tr>
<td>GM_CalcIntersectionLineLine</td>
<td>Calculation of intersection-point line - line.</td>
</tr>
<tr>
<td>GM_CalcMean</td>
<td>Calculation of the average result from several observations.</td>
</tr>
<tr>
<td>GM_CalcMean_Add</td>
<td>Calculation of the average result from several observations.</td>
</tr>
<tr>
<td>GM_CalcMeanOfHz</td>
<td>Calculation of the average from several Hz-directions.</td>
</tr>
<tr>
<td>GM_CalcMedianOfHz</td>
<td>Calculation of Hz-directions and the average as median.</td>
</tr>
<tr>
<td>GM_CalcOrientationOfHz</td>
<td>Calculation of the circle-section orientation.</td>
</tr>
<tr>
<td>GM_CalcPointInCircle</td>
<td>Calculation of a point on a circle.</td>
</tr>
<tr>
<td>GM_CalcPointInLine</td>
<td>Calculation of a point on a line.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GM_CalcTriangle</td>
<td>Calculation of the missing values of a triangle.</td>
</tr>
<tr>
<td>GM_CalcVAndSlope</td>
<td>Calculation of zenith- and slope-distance from given points (Cartesian coordinates).</td>
</tr>
<tr>
<td>GM_ConvertAngle</td>
<td>Conversion of angle from one system into the other.</td>
</tr>
<tr>
<td>GM_ConvertDecSexa</td>
<td>Conversion of value from the decimal into the sexagesimal system.</td>
</tr>
<tr>
<td>GM_ConvertDist</td>
<td>Conversion of distances from one system into the other.</td>
</tr>
<tr>
<td>GM_ConvertExcentricHzV</td>
<td>Re-centration of hz- and v-direction.</td>
</tr>
<tr>
<td>GM_ConvertExcentricHzVDist</td>
<td>Re-centration of hz- and v-direction and distance.</td>
</tr>
<tr>
<td>GM_ConvertPressure</td>
<td>Conversion of pressure from one system into the other.</td>
</tr>
<tr>
<td>GM_ConvertSexaDec</td>
<td>Conversion of value from the sexagesimal into the decimal system.</td>
</tr>
<tr>
<td>GM_ConvertTemp</td>
<td>Conversion of temperature from one system into the other.</td>
</tr>
<tr>
<td>GM_ConvertVDirection</td>
<td>Conversion of v-directions from one system into the other.</td>
</tr>
<tr>
<td>GM_CopyPoint</td>
<td>Copy the contents of a point.</td>
</tr>
<tr>
<td>GM_InitQXXMatrix</td>
<td>Initialise the QXX-Matrix for a point structure.</td>
</tr>
<tr>
<td>GM_LineAzi</td>
<td>Calculate azimuth of a line.</td>
</tr>
<tr>
<td>GM_MathOrSurveyorsAngleConv</td>
<td>Adjusts a math angle in radians to a surveyor's angle in radians or vice versa.</td>
</tr>
<tr>
<td>GM_SamePoint</td>
<td>Test if two points are equal.</td>
</tr>
<tr>
<td>GM_TransformPoints</td>
<td>Transformation of point.</td>
</tr>
<tr>
<td>GM_Traverse3D</td>
<td>Convert a point in polar coordinates to Cartesian coordinates.</td>
</tr>
</tbody>
</table>
5.6.2 GeoMath Structures

**GM_Mean_StdDev - Exactness**

**Description**  
With this structure, average, middle error of average, and middle error of any observation are defined.

```plaintext
TYPE GM_Mean_StdDev_Type  
dMeanValue AS Double   average [m]  
dStdvOfMean AS Double   middle Error of average [m]  
dStdvOfAnyValue AS Double   middle Error of any observation [m]  
END GM_Mean_StdDev_Type
```

**GM_Excenter_Elems - Eccentric Elements**

**Description**  
Elements of the eccentric observation.

```plaintext
TYPE GM_Excenter_Elems_Type  
dHzCent AS Double   horizontal angle to centre [rad]  
dExDist AS Double   horizontal distance to centre [m]  
dDHeight AS Double   height difference excenter-centre  
END GM_Excenter_Elems_Type
```

**GM_4Transform_Param - Transformation parameters**

**Description**  
In this structure the transformation parameters are defined.

```plaintext
TYPE GM_4Transform_Param_Type  
dPhi AS Double   rotation angle measure  
dScal AS Double  
END GM_4Transform_Param_Type
```
dX0 AS Double  translation in X-direction

dY0 AS Double  translation in Y-direction

END GM_4Transform_Param_Type

GM_Measurements - Measurement

Description  Structure used for measurement (polar coordinates).

TYPE GM_Measurements_Type
  dHz AS Double  horizontal reading [rad]
  dV AS Double  vertical reading [rad]
  dSlopeDist AS Double  slope distance [m]
END GM_Measurements_Type

GM_QXX_Matrix - Co-Factor Matrix of the Unknown

Description  With this structure the coefficients of the cofactor matrix of the unknown are defined.

TYPE GM_QXX_Matrix_Type
  dM0 AS Double  middle weight unit error
  dA11 AS Double  dA11 to dA33 are the coefficient of the cofactor matrix of the unknown
  dA12 AS Double
  dA13 AS Double
  dA22 AS Double
  dA23 AS Double
  dA33 AS Double
END GM_QXX_Matrix_Type

GM_Point - Definition of a point

Description  With this structure the point is defined.

TYPE GM_Point_Type
GeoBASIC Reference Manual

5. Standard Functions

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dE</td>
<td>AS Double</td>
<td>e-coordinate [m]</td>
</tr>
<tr>
<td>dN</td>
<td>AS Double</td>
<td>n-coordinate [m]</td>
</tr>
<tr>
<td>dHeight</td>
<td>AS Double</td>
<td>height [m]</td>
</tr>
<tr>
<td>bHeightValid</td>
<td>AS Logical</td>
<td>indicates whether the height is valid</td>
</tr>
<tr>
<td>Koeff</td>
<td>AS GM_QXX_Matrix_Type</td>
<td>coefficient of the cofactor matrix of the unknown</td>
</tr>
</tbody>
</table>

END GM_Point_Type

**GM_Line - Definition of a line**

**Description**
With this structure a line is defined.

```
TYPE GM_Line_Type
    iType AS Integer
        defines the line type
        Valid values: Meaning:
        GM_POINT_AND_POINT Line defined with two points
        GM_POINT_AND_AZI Line defined with point and azimuth
    FirstPt AS GM_Point_Type
        first point on the line
    SecondPt AS GM_Point_Type
        second point on the line
    dAzi AS Double
        azimuth [rad]
    dParShift AS Double
        parallel displacement
END GM_Line_Type
```
GM_Circle - Definition of a circle
Description
With this structure a circle is defined.

```plaintext
TYPE GM_Circle_Type
    Center AS GM_Point_Type  ; centre of the circle
    Radius  AS Double        ; radius
END GM_Circle_Type
```

GM_Triangle_Values - Sides and angles of a triangle
Description
With this structure the sides and angles of a triangle are defined.

```plaintext
TYPE GM_Triangle_Values_Type
    dSide1 AS Double  ; 1st triangle side [m]
    dSide2 AS Double  ; 2nd triangle side [m]
    dSide3 AS Double  ; 3rd triangle side [m]
    dAngle1 AS Double ; angle opposite side 1 [rad]
    dAngle2 AS Double ; angle opposite side 2 [rad]
    dAngle3 AS Double ; angle opposite side 3 [rad]
END GM_Triangle_Values_Type
```

GM_Triangle_Accuracy - Accuracy of angle and side of the triangle
Description
With this structure the exactness of the sides and angles are defined.

```plaintext
TYPE GM_Triangle_Accuracy_Type
    dMeS1 AS Double ; mean error of the 1st triangle side [m]
    dMeS2 AS Double ; mean error of the 2nd triangle side [m]
    dMeS3 AS Double ; mean error of the 3rd triangle side [m]
    dMeA1 AS Double ; mean error of the angle opposite side 1 [rad]
    dMeA2 AS Double ; mean error of the angle opposite
```

TPS1100-Version 1.30 5-31
5.6.3 GM_CalcAreaOfCoord

Description  Calculation of area result from measurement.

Declaration  

\[
\begin{align*}
\text{GM\_CalcAreaOfCoord\_Start(} & \text{StartPt AS GM\_Point\_Type) } \\
\text{GM\_CalcAreaOfCoord\_Add(} & \text{CurrPt AS GM\_Point\_Type, } \\
& \text{byVal dRadius AS Double, } \\
& \text{dArea AS Double, } \\
& \text{iReturnCode AS Integer) }
\end{align*}
\]

Remarks  With the first function the calculation of the area of an arbitrary polygon can be started by defining the start-point \((\text{StartPt, \text{cartesian coordinates}})\). The second function allows to extend the polygon by adding new points. When \((\text{CurrPt})\) equates to the start-point, the area of the now closed polygon will be calculated.

Note  The computation is done the plane, i.e. the height is ignored.

Note  For the used formula see Appendix, Geodesy Math. Formulas.

Parameters  

- \text{StartPt in start point of the polygon in Cartesian coordinates}
- \text{CurrPt in current point to be added to the polygon in cart. coordinates}
if dRadius > 0, the connection between the last point added and the current point (current edge) is assumed to be an arc. The area for the arc segment will be calculated as follows:

\[ F = \frac{1}{2} \times d\text{Radius}^2 \times (d - \sin(d)) \],

where \( d \) is the angle change of the arc.

- **dArea out**: superfiaces of the closed polygon [m²]
- **iReturnCode out**: return code
  - **GM_NO_SOLUTION**: current and start-point are not yet identical, point has been added to polygon

**Return Codes**

- **GM_RADIUS_NOT_POSSIBLE**: invalid value for dRadius; this is the case if
  1) \( d\text{Radius} \neq 0.0 \) and
  2) \( \text{Abs}(d\text{Radius}) < \frac{\text{length of current edge}}{2} \).
Example

Calculate the area defined by 3 given edges.

DIM iRetCode AS Integer
DIM CurrPt AS GM_Point_Type
DIM dRadius AS Double
DIM dArea AS Double

'init CurrPt and dRadius with the first point
Init_GM_Point_Type( CurrPt )
CurrPt.dE = 1.0
CurrPt.dN = 1.0
GM_CalcAreaOfCoord_Start( CurrPt )

'add the second point
CurrPt.dE = 3.0
CurrPt.dN = 1.0
GM_CalcAreaOfCoord_Add( CurrPt, dRadius,
                        dArea, iRetCode )

'add the third point
CurrPt.dE = 2.0
CurrPt.dN = 2.0
GM_CalcAreaOfCoord_Add( CurrPt, dRadius,
                        dArea, iRetCode )

'close the polygon: back to the first point
CurrPt.dE = 1.0
CurrPt.dN = 1.0
GM_CalcAreaOfCoord_Add( CurrPt, dRadius,
                        dArea, iRetCode )
5.6.4 GM_CalcAreaOfMeas

Description
Calculation of area result from measurement.

Declaration
GM_CalcAreaOfMeas_Start( StartPt AS GM_Measurements_Type )

GM_CalcAreaOfMeas_Add(
    CurrPt AS GM_Measurements_Type,
    ByVal dRadius AS Double,
    dArea AS Double,
    iReturnCode AS Integer )

Remarks
With the first function the calculation of the area of an arbitrary polygon can be started by defining the start-point (startPt, polar coordinates). The second function allows to extend the polygon by adding new points. When currPt equates the start-point, the area of the now closed polygon will be calculated.

Note
The computation is done the plane, i.e. the horizontal distance is computed and the height is ignored. For the used formula see Appendix, Geodesy Math. Formulas.

Parameters

StartPt  in  start - point of the polygon in polar coordinates
CurrPt  in  current point to be added to the polygon in polar coordinates
dRadius  in  if dRadius>0, the connection between the last point added and the current point (current edge) is assumed to be an arc. The area for the arc segment will be calculated as follows:

\[ F = \frac{1}{2} \cdot dRadius^2 \cdot (d - \sin(d)), \]

where \( d \) is the angle change of the arc.

dArea  out  Superficies of the closed polygon [m²]
Return code; possible values:

- **RC_OK**: successful calculation of area
- **GM_NO_SOLUTION**: current and start-point are not yet identical, point has been added to polygon

### Return Codes

- **RC_OK**: successful calculation of area
- **GM_RADIUS_NOT_POSSIBLE**: invalid value for dRadius; this is the case if
  1) \( d\text{Radius} \neq 0.0 \) and
  2) \( \text{Abs}(d\text{Radius}) < \frac{\text{length of current edge}}{2} \)

### Example

Calculate the area from 3 given edges.

```basic
DIM iRetCode AS Integer
DIM CurrPt AS GM_Measurements_Type
DIM dRadius AS Double
DIM dArea AS Double

' init CurrPt and dRadius with the first point
Init_GM_Point_Type( CurrPt )
CurrPt.dHz = 0.0
CurrPt.dV = 1.5707963
CurrPt.dSlopeDist = 10.0
GM_CalcAreaOfMeas_Start( CurrPt )

' add the second point
CurrPt.dHz = 1.5707863
CurrPt.dV = 1.5707963
CurrPt.dSlopeDist = 5.0
GM_CalcAreaOfMeas_Add( CurrPt, dRadius, dArea, iRetCode )
```
5.6.5 GM_CalcAziAndDist

**Description** Calculation of azimuth and distance result from coordinates.

**Declaration**

```plaintext
GM_CalcAziAndDist(
  StationPt AS GM_Point_Type,
  TargetPt AS GM_Point_Type,
  dAzi AS Double,
  dDist AS Double,
  dStdvAzi AS Double,
  dStdvDist AS Double )
```

**Remarks**

This function is calculating azimuth and distance result from coordinates.

**Note** Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

- **StationPt** in coordinates and exactness of the station-point
- **TargetPt** in coordinates and exactness of the target-point
- **dAzi** out calculated azimuth [rad]
- **dDist** out calculated distance [m]
- **dStdvAzi** out set to 0 (reserved for future use)
- **dStdvDist** out set to 0 (reserved for future use)
Return Codes

RC_OK
successful calculation of azimuth
and distance

GM_IDENTICAL_POINTS
Station- and target-point are
identical, calculation not
possible. The recovered values
are not defined.

Example
Calculate the distance of a target from a station according to given
StationPt and TargetPt.

`DIM StationPt AS GM_Point_Type
DIM TargetPt AS GM_Point_Type
DIM dAzi AS Double
DIM dDist AS Double
DIM dStdvAzi AS Double
DIM dStdvDist AS Double`

`'initialize StationPt and TargetPt
StationPt.dN   = 3.0
StationPt.dE  = 0.0
StationPt.dHeight = 0.0
TargetPt.dN    = 0.0
TargetPt.dE   = 5.0
TargetPt.dHeight = 0.0
'in GM_QXX_MATRIX set all values to 0.0 (for
' StationPt and TargetPt)

GM_CalcAziAndDist( StationPt, TargetPt,
                  dAzi, dDist,  
                  dStdvAzi, dStdvDist)"
5.6.6 GM_CalcCenterAndRadius

Description  Calculation of centre coordinate and radius result from 3 points.

Declaration  GM_CalcCenterAndRadius(
                     Pt0    AS GM_Point_Type,
                     Pt1    AS GM_Point_Type,
                     Pt2    AS GM_Point_Type,
                     dRadius AS Double,
                     Center AS GM_Point_Type,
                     dMRadius AS Double )

Remarks  This function is calculating the coordinate of the centre and the radius result from 3 given points.

Note  Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pt0</td>
<td>in contains the coordinate and the exactness</td>
</tr>
<tr>
<td></td>
<td>of the 1. point</td>
</tr>
<tr>
<td>Pt1</td>
<td>in contains the coordinate and the exactness</td>
</tr>
<tr>
<td></td>
<td>of the 2. point</td>
</tr>
<tr>
<td>Pt2</td>
<td>in contains the coordinate and the exactness</td>
</tr>
<tr>
<td></td>
<td>of the 3. point</td>
</tr>
<tr>
<td>dRadius</td>
<td>out calculated radius [m]</td>
</tr>
<tr>
<td>Center</td>
<td>out calculated coordinates and exactness of</td>
</tr>
<tr>
<td></td>
<td>the centre</td>
</tr>
<tr>
<td>dMRadius</td>
<td>out middle error of the radius [m]</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM_PTS_IN_LINE</td>
<td>The 3 points are located on one line, the</td>
</tr>
<tr>
<td></td>
<td>calculation not possible. All output values</td>
</tr>
<tr>
<td></td>
<td>are undefined.</td>
</tr>
</tbody>
</table>
**Example**  
Calculate the centre from the 3 given points.

```plaintext
DIM Pt0 AS GM_Point_Type
DIM Pt1 AS GM_Point_Type
DIM Pt2 AS GM_Point_Type
DIM dRadius AS Double
DIM dMRadius AS Double
DIM Center AS GM_Point_Type

GM_CalcCenterAndRadius( Pt0, Pt1, Pt2, dRadius,
                        Center, dMRadius )
```

### 5.6.7 GM_CalcClothCoord

**Description**  
Calculation of coordinate on the unitary clothoid (A=1).

**Declaration**  
```plaintext
GM_CalcClothCoord( ByVal dTau AS Double,
                    dX AS Double,
                    dY AS Double )
```

**Remarks**  
This function is calculating the coordinate, dependent from the tangent angle, of one point on the unitary clothoid.

**Note**  
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**
- `dTau` in tangent angle [rad]
- `dX` out x-coordinate of the Clothoid point
- `dY` out y-coordinate of the Clothoid point

**Return Codes**
- `RC_OK` always OK

**Example**  
Calculate the centre from the 3 given points.

```plaintext
DIM dX AS Double
DIM dY AS Double

GM_CalcClothCoord( 3.1415, dX, dY )
```
5.6.8 GM_CalcCoord

**Description**
Calculation of coordinate result from azimuth and distance.

**Declaration**

```geo
GM_CalcCoord( StationPt AS GM_Point_Type,
             byVal dAzi AS Double,
             byVal dHorizDist AS Double,
             TargetPt AS GM_Point_Type )
```

**Remarks**
This function is calculating the coordinate result from azimuth and distance.

**Note**
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**
- **StationPt** in coordinates and exactness of the station point
- **dAzi** in azimuth [rad]
- **dHorizDist** in horizontal distance [m]
- **TargetPt** out coordinates and exactness of the target point

**Return Codes**
- **RC_OK** always OK

**Example**
Calculate the distance of a target from a station according to given azimuth and horizontal distance.

```geo
DIM StationPt AS GM_Point_Type
DIM TargetPt AS GM_Point_Type

'initialize StationPt
GM_CalcCoord( StationPt, 0.5, 1000.0, TargetPt )
```
5.6.9 GM_CalcDistPointCircle

**Description**  
Calculation of the distance point to circle and the base point of plumb line.

**Declaration**  
```basic  
GM_CalcDistPointCircle(  
    Point AS GM_Point_Type,  
    Circle AS GM_Circle_Type,  
    dDist AS Double,  
    FootPoint AS GM_Point_Type  
)  
```

**Remarks**  
This function is calculating the distance of one point to a circle and his base-point of the foot of a perpendicular observation.

**Note**  
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>in</td>
<td>coordinates and exactness of the point to be plumbed</td>
</tr>
<tr>
<td>Circle</td>
<td>in</td>
<td>circle</td>
</tr>
<tr>
<td>dDist</td>
<td>out</td>
<td>distance point - circle [m]</td>
</tr>
<tr>
<td>FootPoint</td>
<td>out</td>
<td>coordinate of the base point of plumb line</td>
</tr>
</tbody>
</table>

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>always OK</td>
</tr>
</tbody>
</table>

**Example**  
Calculate the distance of a point to a circle.

```basic  
DIM Pt AS GM_Point_Type  
DIM Circle AS GM_Circle_Type  
DIM dDist AS Double  
DIM BasePt AS GM_Point_Type  

'initialize Pt and circle with any values  
GM_CalcDistPointCircle( Pt, Circle,  
    dDist, BasePt )  
```
5.6.10 GM_CalcDistPointCloth

Description
Calculation of the distance point - Clothoid and the base point of plumb line.

Declaration
GM_CalcDistPointCloth(
    BA AS GM_Point_Type,
    BE AS GM_Point_Type,
    Point AS GM_Point_Type,
    byVal dA AS Double,
    byVal dL AS Double,
    dDist AS Double,
    dDistAlongSpiral AS Double,
    FootPoint AS GM_Point_Type )

Remarks
This function is calculating the distance of one point to the clothoid and his base point of plumb line in the area of $0 < \tau < \pi/2$. Prerequisite that, the Clothoid is placed in the country-coordinate system.

Note
Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
<td>in</td>
<td>beginning of the arc in the country coordinate system</td>
</tr>
<tr>
<td>BE</td>
<td>in</td>
<td>end of the arc in the country coordinate system</td>
</tr>
<tr>
<td>Point</td>
<td>in</td>
<td>point to be plumbed out in the country coordinate system</td>
</tr>
<tr>
<td>dA</td>
<td>in</td>
<td>clothoid - parameter</td>
</tr>
<tr>
<td>dL</td>
<td>in</td>
<td>arc length [m]</td>
</tr>
<tr>
<td>dDist</td>
<td>out</td>
<td>distance point - Clothoid [m]</td>
</tr>
<tr>
<td>dDistAlongSpiral</td>
<td>out</td>
<td>distance along arc</td>
</tr>
<tr>
<td>FootPoint</td>
<td>out</td>
<td>coordinate of the base point of foot of a perpendicular observation</td>
</tr>
</tbody>
</table>

Return Codes

TPS1100-Version 1.30 5-43
The foot of a perpendicular observation is placed outside the area $0 < \tau < \pi / 2$, not perpendicular.

**Example**

Calculate the distance of a point to a clothoid.

```geoBasic
DIM BA AS GM_Point_Type
DIM BE AS GM_Point_Type
DIM Point AS GM_Point_Type
DIM dL AS Double
DIM dA AS Double
DIM dDist AS Double
DIM dDist2 AS Double
DIM BasePt AS GM_Point_Type

'initialize BA, BE, Point, dA, dL adequately
GM_CalcDistCloth( BA, BE, Point, dA, dL,
    dDist, dDist2, BasePt )
```

### 5.6.11 GM_CalcDistPointLine

**Description**
Calculation of the distance point - line and the base point of foot of a perpendicular observation.

**Declaration**

```geoBasic
GM_CalcDistPointLine( 
    Line AS GM_Line_Type,
    Point AS GM_Point_Type,
    dDistX AS Double,
    dDistY AS Double,
    FootPoint AS GM_Point_Type )
```

**Remarks**
This function is calculating the distance of one point to the line and his base point of the foot of a perpendicular observation. One effective definition of line is also possible result from one parallel (see predefined type GM_Line_Type).

**Note**
Used formula: see Appendix, Geodesy Math. Formulas.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>in</td>
<td>line</td>
</tr>
<tr>
<td>Point</td>
<td>in</td>
<td>point to be plumbed out</td>
</tr>
<tr>
<td>dDistX</td>
<td>out</td>
<td>distance point - line [m]</td>
</tr>
<tr>
<td>dDistY</td>
<td>out</td>
<td>distance point in the direction of the line [m]</td>
</tr>
<tr>
<td>FootPoint</td>
<td>out</td>
<td>coordinate of the base point of plumb line</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>successful calculation</td>
</tr>
<tr>
<td>GM_IDENTICAL_PTS</td>
<td>Start - and endpoint of the line are identical. Calculation is not possible. The recovered values are not defined.</td>
</tr>
</tbody>
</table>

Example

Calculate the distance of a point to a line.

```geo
DIM Line AS GM_Line_Type
DIM Point AS GM_Point_Type
DIM dDistX AS Double
DIM dDistY AS Double
DIM BasePt AS GM_Point_Type

'initialize Line and Point adequatley

GM_CalcDistPointLine( Line, Point, dDistX, dDistY, BasePt )
```
5.6.12 GM_CalcHiddenPointObservation

**Description**  
Calculated measurement to the hidden point.

**Declaration**  
```plaintext
GM_CalcHiddenPointObservation(
    Point1 AS GM_Measurements_Type,
    Point2 AS GM_Measurements_Type,
    ByVal dDistP1P2 AS Double,
    ByVal dDistP1HP AS Double,
    HiddenPt AS GM_Measurements_Type
)
```

**Remarks**  
This function is calculating the measurement to the hidden point, result from the measurements onto both reflectors of the hidden point staff.

**Note**  
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

- **Point1** in  
  contains the measurement of the reflector 1 of hidden point staff
- **Point2** in  
  contains the measurement of the reflector 2 of hidden point staff
- **dDistP1P2** in  
  Distance of both reflectors [m].
- **dDistP1HP** in  
  Distance of reflectors 1 and the hidden point's [m].
- **HiddenPt** out  
  calculated measurement to the hidden point

**Return Codes**

- **GM_IDENTICAL_PTS**  
  Both measurement onto the same point. Calculation is not possible. The recovered values are not defined.
- **GM_PLAUSIBILITY_ERR**  
  The distance to the reflectors does not correspond to the measurement. The recovered values are not defined.
Example: Calculate the hidden point.
DIM Point1 AS GM_Point_Type
DIM Point2 AS GM_Point_Type
DIM dDistP1P2 AS Double
DIM dDistP1Hd AS Double
DIM HiddenPt AS GM_Point_Type

'initialize Point1, Point2, dDistP1P2, dDistP1Hd adequately

GM_CalcHiddenPointObservation( Point1, Point2, dDistP1P2, dDistP1Hd, HiddenPt )

5.6.13 GM_CalcIntersectionCircleCircle

Description: Calculation of intersection-point circle - circle.

Declaration: GM_CalcIntersectionCircleCircle(
  FirstCircle AS GM_Circle_Type,
  SecondCircle AS GM_Circle_Type,
  FirstInters AS GM_Point_Type,
  SecondInters AS GM_Point_Type,
  iReturnCode AS Integer )

Remarks: This function is calculating the intersection point(s) between two circles.

Note: Used formula: see Appendix, Geodesy Math. Formulas.
Parameters

FirstCircle in  Definition of the 1. circle
SecondCircle in  Definition of the 2. circle
FirstInters out  Coordinate. and exactness of the 1. intersect. point
SecondInters out  Coordinate. and exactness of the 2. intersect. point
iReturnCode out  indicates the number of solutions
   GM_NO_SOLUTION  no intersection point
   GM_ONE_SOLUTION  exactly one solution. The values for
                    Second-Inters are nor defined.
   GM_TWO_SOLUTION  two intersection points

Return Codes

RC_OK  successful calculation

Example

Calculate the intersection points between the circles.

DIM Circle1 AS GM_Circle_Type
DIM Circle2 AS GM_Circle_Type
DIM Interspt1 AS GM_Point_Type
DIM Interspt2 AS GM_Point_Type
DIM iRetCode AS Integer

'initialize circle1 and circle2 adequately

GM_CalcIntersectionCircleCircle( Circle1, Circle2, Interspt1, Interspt2, iRetCode )
5.6.14 GM_CalcIntersectionLineCircle

**Description**  Calculation of intersection-point line - circle.

**Declaration**

```basic
GM_CalcIntersectionLineCircle(
    Line AS GM_Line_Type,
    Circle AS GM_Circle_Type,
    FirstInters AS GM_Point_Type,
    SecondInters AS GM_Point_Type,
    iReturnCode AS Integer )
```

**Remarks**

This function is calculating the intersection-point(s) between one line and one circle. The line could show a transverse displacement and can be defined as a result from 2 points, or as result from one point and azimuth (see predefined type GM_Line).

**Note**  Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>in</td>
<td>Definition of the line.</td>
</tr>
<tr>
<td>Circle</td>
<td>in</td>
<td>Definition of the circle.</td>
</tr>
<tr>
<td>FirstInters</td>
<td>out</td>
<td>Coordinate and exactness of the 1. intersect. point.</td>
</tr>
<tr>
<td>SecondInters</td>
<td>out</td>
<td>Coordinate and exactness of the 2. intersect. point.</td>
</tr>
<tr>
<td>iReturnCode</td>
<td>out</td>
<td>indicates the number of solutions</td>
</tr>
</tbody>
</table>

- **GM_NO_SOLUTION**  no intersection point
- **GM_ONE_SOLUTION**  exactly one solution; the values for Second-Inters are not defined
- **GM_TWO_SOLUTIONS**  two intersection points
Return Codes

GM_IDENTICAL_PTS  Start- and endpoint of the line are identical. Calculation is not possible.

Example  Calculate the intersection points between the line and the circle.

```plaintext
DIM Line AS GM_Line_Type
DIM Circle AS GM_Circle_Type
DIM Interspt1 AS GM_Point_Type
DIM Interspt2 AS GM_Point_Type
DIM iRetCode AS Integer

'initialize Line and Circle adequately
GM_CalcIntersectionLineCircle( Line, Circle,
                              Interspt1,
                              Interspt2,
                              iRetCode )
```

5.6.15  GM_CalcIntersectionLineLine

**Description**  Calculation of intersection-point line - line.

**Declaration**  
```
GM_CalcIntersectionLineLine( 
    FirstLine AS GM_Line_Type,
    SecondLine AS GM_Line_Type,
    Intersection AS GM_Point_Type,
    iReturnCode AS Integer )
```

**Remarks**  This function is calculating the intersection-point between two Lines. The lines could show a transverse displacement and can be defined as a result from 2 points, or as result from one point and azimuth (see predefined type GM_Line).

**Note**  Used formula: see Appendix, Geodesy Math. Formulas.
Parameters

FirstLine in Definition of the 1. line.
SecondLine in Definition of the 2. line.
Intersection out Coordinate and exactness of the intersect. point.
iReturnCode out indicates the number of solutions
GM_NO_SOLUTION no intersection point, i.e. the lines are parallel
GM_ANGLE_SMALLER_15GON Warning: the intersect. Angle of the line is smaller than 15 gon. The intersect. point was still calculated.

Return Codes

GM_IDENTICAL_PTS Start- and endpoint of a line are identical. Calculation is not possible.

Example

Calculate the intersection points between the 2 lines.

DIM Linel AS GM_Line_Type
DIM Line2 AS GM_Line_Type
DIM IntersPt AS GM_Point_Type
DIM iRetCode AS Integer

' initialize Linel and Line2 adequately

GM_CalcIntersectionLineLine( Linel, Line2, IntersPt, iRetCode )
5.6.16 GM_CalcMean

**Description**  
Calculation of the average result from several observations.

**Declaration**  
```
GM_CalcMean_Add(
    ByVal dObservation AS Double,
    ByVal dWeight AS Double,
    ByVal lStartNew AS Logical )
```

```
GM_CalcMean( Mean AS GM_Mean_StdDev_Type )
```

**Remarks**  
The first function creates an internal data list and adds the values \((dObservation, dWeight)\) to it. The second is calculating the average, the middle error of the average, the middle error of the observations stored in the data list.

**Note**  
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

- **dObservation**  
in  
observation to be averaged

- **dWeight**  
in  
weight for averaging

- **lStartNew**  
in  
TRUE: the given values \((dObservation, dWeight)\) are the first in a new series (initialisation). The old series (belonging to this function) will be lost.
FALSE: add the values to an existing data series.

- **Mean**  
out  
calculated results from the current data series
Return Codes

RC_OK
successful creation, adding, and calculation

GM_OUT_OF_RANGE
This may occur when calling GM_CalcMean_Add( ..., ..., FALSE ).
Two reasons:
1. no data series exists,
2. too many data items.

RC_IV_RESULT
When calling GM_CalcMean with no successful previous call of GM_CalcMean_Add.

GM_TOO_FEW_OBSERVATIONS
Too few observations to be able to calculate the average. The recovered values are not defined.

GM_PLAUSIBILITY_ERR
The sum of the weights is 0.

Example
Calculate the weighted average and standard deviation.
DIM Mean AS GM_Mean_StdDev_Type
GM_CalcMean_Add( 1.0, 0.5, TRUE )
GM_CalcMean_Add( 2.0, 1.0, FALSE )
GM_CalcMean_Add( 3.0, 1.5, FALSE )
GM_CalcMean( Mean )
5.6.17 GM_CalcMeanOfHz

**Description**  
Calculation of the average from several Hz-directions.

**Declaration**  
```
GM_CalcMeanOfHz_Add(
    ByVal dHzDirection AS Double,
    ByVal lStartNew AS Logical )
```

```
GM_CalcMeanOfHz(
    Mean AS GM_Mean_StdDev_Type )
```

**Remarks**  
The first function creates an internal data list and adds Hz-directions to it. The second is calculating the average, the middle error of the average, the middle error of any direction evaluating the added Hz-directions in the list.

**Note**  
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

- **dHzDirection**  
in  
Hz - direction

- **lStartNew**  
in  
TRUE: the given value (dHzDirection) is the first in a new series (initialisation). The old series (belonging to this function) will be lost.  
FALSE: add the values to an existing data series.

- **Mean**  
out  
calculated results from the current data series.
Return Codes

RC_OK successful creation, adding, and evaluation

RC_IV_RESULT When calling GM_CalcMeanOfHz with no successful previous call of GM_CalcMeanOfHz_Add.

GM_OUT_OF_RANGE This may occur when calling GM_CalcMeanOfHz_Add(..., ..., FALSE).
Two reasons:
1. no data series exists,
2. too many data items.

GM_TOO_FEW_OBSERVATIONS Too few observations to be able to calculate the average. The recovered values are not defined.

Example Calculate the weighted average etc.

```
DIM Mean AS GM_Mean_StdDev_Type

GM_CalcMeanOfHz_Add( 1.0, TRUE )
GM_CalcMeanOfHz_Add( 2.0, FALSE )
GM_CalcMeanOfHz_Add( 3.0, FALSE )
GM_CalcMean( Mean )
```

5.6.18 GM_CalcMedianOfHz

Description Calculation of Hz-directions and the average as median.

Declaration

```
GM_CalcMedianOfHz_Add(
    ByVal dHzDirection AS Double,
    ByVal lStartNew AS Logical)
```

```
GM_CalcMedianOfHz( dMedian AS Double )
```

Remarks The first function creates an internal data list and adds Hz-directions to it. The second is calculating the average as median evaluating the added Hz-directions in the list.
5. Standard Functions

### Parameters

- **dHzDirection** in Hz - direction
- **lStartNew** in TRUE: the given value (dHzDirection) is the first in a new series (initialisation). The old series (belonging to this function) will be lost. FALSE: add the values to an existing data series.

- **DMedian** out Median [rad]

### Return Codes

- **RC_OK** successful creation, adding, and evaluation
- **RC_IV_RESULT** When calling GM_CalcMedianOfHz with no successful previous call of GM_CalcMedianOfHz_Add.
- **GM_OUT_OF_RANGE** This may occur when calling GM_CalcMedianOfHz_Add(..., ..., FALSE )
  Two reasons:
  1. no data series exists,
  2. too many data items.
- **GM_TOO_FEW_OBSERVATIONS** Too few observations to be able to calculate the average. The recovered values are not defined.

### Example

Calculate the median.

```geo
DIM dMedian AS Double
GM_CalcMedianOfHz_Add( 1.0, TRUE )
GM_CalcMedianOfHz_Add( 2.0, FALSE )
GM_CalcMedianOfHz_Add( 3.0, FALSE )
GM_CalcMedian( dMedian )
```
5.6.19 GM_CalcOrientationOfHz

Description  Calculation of the circle-section orientation of graduated circle.

Declaration  

\[
\begin{align*}
\text{GM\_CalcOrientationOfHz\_Add}( & \text{Station AS GM\_Point\_Type, } \\
& \text{Target AS GM\_Point\_Type, } \\
& \text{byVal dHz AS Double, } \\
& \text{byVal lStartNew AS Logical }) \\
\end{align*}
\]

\[
\begin{align*}
\text{GM\_CalcOrientationOfHz}( & \text{Ori AS GM\_Mean\_StdDev\_Type, } \\
& \text{dOriMedian AS Double }) \\
\end{align*}
\]

Remarks  The first function creates an internal data list and adds the data to it. The second is calculating the orientation of graduated circle evaluating the added data in the list.

Note  Used formula: see Appendix, Geodesy Math. Formulas.

Parameters  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station</td>
<td>in GM_Point_Type</td>
<td>Coordinate of the station-point.</td>
</tr>
<tr>
<td>Target</td>
<td>in</td>
<td>measured point</td>
</tr>
<tr>
<td>dHz</td>
<td>in Double</td>
<td>observed Hz-direction</td>
</tr>
<tr>
<td>lStartNew</td>
<td>in Logical</td>
<td>TRUE: the given value (dHzDirection) is the first in a new series (initialisation). The old series (belonging to this function) will be lost. FALSE: add the values to an existing data series.</td>
</tr>
<tr>
<td>Ori</td>
<td>out</td>
<td>unknown -orientation -variable and the exactness</td>
</tr>
<tr>
<td>dOriMedian</td>
<td>out Double</td>
<td>as median middle unknown - orientation - variable</td>
</tr>
</tbody>
</table>
Return Codes

- **RC_OK**: successful creation, adding, and evaluation
  When calling `GM_CalcOrientationOfHz` with no successful previous call of `GM_CalcOrientationHz_Add`.

- **RC_IV_RESULT**: This may occur when calling `GM_CalcOrientationOfHz_Add( .., .., FALSE )`.
  Two reasons:
  1. no data series exists,
  2. too many data items.

- **GM_OUT_OF_RANGE**: Too few observations to be able to calculate the average. The recovered values are not defined.

Example

```
DIM Station AS GM_Point_Type
DIM Target AS GM_Point_Type
DIM Ori AS GM_Mean_StdDev_Type
DIM dOriMedian AS Double

'initialize Station and Target
GM_CalcOrientationOfHz_Add( Station, Target, 1.571, TRUE )
GM_CalcOrientationOfHz_Add( Station, Target, 3.109, FALSE )
GM_CalcOrientationOfHz_Add( Station, Target, 2.395, FALSE )
GM_CalcOrientationOfHz( Ori, dOriMedian )
```
5.6.20 GM_CalcPointInLine

Description  Calculation of a point on a line.

Declaration  

```
GM_CalcPointInLine(
    Line AS GM_Line_Type,
    ByVal dDist AS Double,
    Point AS GM_Point_Type )
```

Remarks  This function is calculating the point with the distance dDist from a given point on a line (the first point of the line definition - see predefined structure GM_Line_Type) on the line.

Note  Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>in</td>
<td>Definition of the line.</td>
</tr>
<tr>
<td>dDist</td>
<td>in</td>
<td>Distance of the point on the line to be calculated, from the 1. point of the line [m].</td>
</tr>
<tr>
<td>Point</td>
<td>out</td>
<td>Calculated point on the line.</td>
</tr>
</tbody>
</table>

Return Codes

- **GM_IDENTICAL_PTS**: Start- and endpoint of a line are identical. Calculation is not possible.

Example  Calculate the point in the line.

```basic
DIM Line AS GM_Line_Type
DIM Point AS GM_Point_Type

'initialize line
GM_CalcPointInLine( Line, 1.0, Point )
```
5.6.21 GM_CalcPointInCircle

**Description**
Calculation of a point on a circle.

**Declaration**

```vbc
GM_CalcPointInCircle( 
    StartOfArc AS GM_Point_Type, 
    EndOfArc AS GM_Point_Type, 
    ByVal dRadius AS Double, 
    ByVal dLengthOfArc AS Double, 
    Point AS GM_Point_Type )
```

**Remarks**
This function is calculating the point with the distance dDist from a given point on a circle (the first point of the circle definition - see predefined structure GM_Circle_Type) on the circle.

**Note**
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**
- **StartOfArc** in beginning of the arc
- **EndOfArc** in end of the arc
- **dRadius** in radius
- **dLengthOfArc** in arc length clockwise relative to StartOfArc are positive
- **Point** out Calculated point on the arc.

**Return Codes**
- **GM_IDENTICAL_PTS** Startpoint and endpoint of the arc are identical. Calculation is not possible.

**Example**
Calculate the point in the circle.

```vbc
DIM Arc1 AS GM_Point_Type
DIM Arc2 AS GM_Point_Type
DIM Point AS GM_Point_Type

'initialize Arc1 and Arc2
GM_CalcPointInLine( Arc1, Arc2, 1.0, Pi, Point )
```
5.6.22 GM_CalcTriangle

Description Calculation of the missing values of a triangle.

Declaration

```
GM_CalcTriangle(
    ByVal iProblemKind AS Integer,
    FirstSol AS GM_Triangle_Values_Type,
    MeanError AS GM_Triangle_Accuracy_Type,
    SecondSol AS GM_Triangle_Values_Type,
    iRetCode AS Integer )
```

Remarks With this function (depending on which triangle is chosen) the missing sides and angles are calculated. If there is a second solution, it also will be calculated and the recovered code will be returned. Subsequently following the calculation of the exactness.

Note Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

iProblemKind in Shows the function which triangle-type has to be used; possible values:

```
GM_SIDE_ANGLE_SIDE   Case:
    Side-Angle-Side
GM_SIDE_SIDE_SIDE
GM_SIDE_SIDE_ANGLE
GM_ANGLE_SIDE_SIDE
GM_ANGLE_ANGLE_SIDE
GM_SIDE_ANGLE_ANGLE
GM_ANGLE_SIDE_ANGLE
```

FirstSol in-out The given sides and angles have to be recorded in this structure.

MeanError in-out The exactness of the corresponding sides respective angles have to be recorded in this structure.

SecondSol out The calculated sides respective angles of the 2. solution (if existing) are recorded in this structure.
Return Codes

**GM_INVALID_TRIANGLE_TYPE**

Invalid triangle-type. There was no calculation. The recovered values are not defined.

Example

Calculate the distance of a target from a station according to given StationPt and TargetPt.

```basic
DIM FirstSol AS GM_Triangle_Values_Type
DIM SecondSol AS GM_Triangle_Values_Type
DIM MeanError AS GM_Triangle_Accuracy_Type
DIM iRetCode AS Integer

'initialize
FirstSol.dSide1 = 3.0
FirstSol.dSide3 = 5.0
FirstSol.dAngle2 = Atn( 4.0/3.0 )

GM_CalcTriangle( GM_SIDE_ANGLE_SIDE, FirstSol, MeanError, SecondSol, iRetCode)

' iRetCode will be GM_ONE_SOLUTION for GM_SIDE_ANGLE_SIDE problems
```
5.6.23 GM_CalcVAndSlope

**Description**  
Calculation of zenith- and slope-distance from given points (Cartesian coordinates).

**Declaration**  
```vbc
GM_CalcVAndSlope(  
    StationPt AS GM_Point_Type,  
    TargetPt AS GM_Point_Type,  
    byVal dInstrHeight AS Double,  
    byVal dRefHeight AS Double,  
    dVZenit AS Double,  
    dSlopeDist AS Double,  
    dStdvVZenit AS Double,  
    dStdvSlopeDist AS Double  
)
```

**Remarks**  
Calculation of zenith- and slope-distance from given points - cart. coordinates.

**Note**  
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**
- **StationPt** in coordinates and exactness of the station point
- **TargetPt** in coordinates and exactness of the target point
- **dInstrHeight** in instrument height [m]
- **dRefHeight** in reflector height [m]
- **dVZenit** out calculated V-direction (zenith - distance) [rad]
- **dSlopeDist** out calculated slope distance [m]
- **dStdvVZenit** out middle error of the V-direction [rad]
- **dStdvSlopeDist** out middle error of the slope-distance [m]

**Return Codes**
- **GM_IDENTICAL_PTS**  
  StationPt and TargetPt are identical. Calculation is not possible.

**Example**  
Calculate the values.
DIM StationPt AS GM_Point_Type
DIM TargetPt AS GM_Point_Type
DIM dVZenit AS Double
DIM dSlopeDist AS Double
DIM dStdvVZenit AS Double
DIM dStdvSlopeDist AS Double

'initialize StationPt, TargetPt
GM_CalcVAndSlope( StationPt, TargetPt, 
1.75, 1.0, dVZenit, 
dSlopeDist, dStdvVZenit, 
dStdvSlopeDist )

5.6.24 GM_ConvertAngle

Description  Conversion of angle from one system into the other.

Declaration  GM_ConvertAngle(
    ByVal iOldSys AS Integer, 
    ByVal dAngleOldSys AS Angle, 
    ByVal iNewSys AS Integer, 
    dAngleNewSys AS Angle )

Remarks  This function is converting angle-value from one standard system into the other.

Note  Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

iOldSys in  standard system of the given angle
GM_DEGREE_SEXA  sexagesimal degrees
GM_DEGREE_DEZ  decimal degrees
GM_GRAD  grads (gons)
GM_RADIANS  radians
GM_MIL  mils
dAngleOldSys in  angle to convert
iNewSys in  standard system of the wanted
5.6.2 GM_ConvertAngle

**Description**
Conversion of an angle from one angle-system to another angle-system.

**Declaration**
```
GM_ConvertAngle( iOldSys, dOldSys, iNewSys, dNewSys )
```

**Remarks**
This function returns the converted angle.

### Return Codes
- **GM_INVALID_ANGLE_SYSTEM**: One of the angle-systems was invalid. There was no conversion. The recovered value is not defined.

### Example
Convert `dAngleOldSys` from [g] to [rad].

The following variables have to be defined:
```
DIM dAngleOldSys AS Angle
DIM dAngleNewSys AS Angle
DIM iOldSys AS Integer
DIM iNewSys AS Integer
```

Initialize values
```
iOldSys = GM_GRAD 'the old angle is given in grad
dAngleOldSys = 200.0 'its value is 200.0 gon
iNewSys = GM_RADIANS 'the new angle should be in radians
```

Call the function
```
GM_ConvertAngle( iOldSys, dAngleOldSys, iNewSys, dAngleNewSys )
```

5.6.25 GM_ConvertDecSexa

**Description**
Conversion of value from decimal into the sexagesimal system.

**Declaration**
```
GM_ConvertDecSexa( ByVal dValueDec AS Double, dValueSexa AS Double )
```

**Remarks**
This function is converting the value from the decimal into the sexagesimal system.

**Note**
Used formula: see Appendix, Geodesy Math. Formulas.
Parameters

- **dValueDec** in decimal value
- **dValueSexa** out sexagesimal value

Return Codes

- **RC_OK** always OK

Example

Convert the angle.

```bas
DIM dAngleSexa AS Double
GM_ConvertDecSexa( dAngleSexa )
```

5.6.26 GM_ConvertDist

Description

Conversion of distances from one system into the other.

Declaration

```bas
GM_ConvertDist(  
    byVal iOldSys AS Integer,  
    byVal dDistOldSys AS Double,  
    byVal iNewSys AS Integer,  
    dDistNewSys AS Double )
```

Remarks

This function is converting distance-values from one standard system into the other.

**Note** Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

- **iOldSys** standard system of the given distance
  - GM_METER meter
  - GM_US_FOOT American feet
  - GM_SURVEY_FOOT surveyor feet
  - GM_INTER_FOOT international feet
- **dDistOldSys** distance to convert
- **iNewSys** standard system of the wanted distance
- **dDistNewSys** converted distance
Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM_INVALID_DIST_SYSTEM</td>
<td>One of the distance standard systems was invalid. There was no conversion. The recovered value was not defined.</td>
</tr>
</tbody>
</table>

Example

Convert dDistOldSys from [m] to [us-feet].

```basic
DIM dDistOldSys AS Double
DIM dDistNewSys AS Double
DIM iOldsys AS Integer
DIM iNewsys AS Integer

'initialize values
iOldsys = GM_METER
dDistOldSys = 1.8
iNewsys = GM_US_FOOT

GM_ConvertDist( iOldsys, dDistOldSys,
                 iNewsys, dDistNewSys )
```

5.6.27 GM_ConvertExcentricHzV

Description
Re-centration of hz- and v-direction.

Declaration

```basic
GM_ConvertExcentricHzV(
    ExCentMeas AS GM_Measurements_Type,
    ExCentElems AS GM_Excenter_Elems_Type,
    Center AS GM_Point_Type,
    Target AS GM_Point_Type,
    CentMeas AS GM_Measurements_Type )
```

Remarks
With this function, the measured values (which are measured to the excenter) could be re-centred to the Centre. The difference to the function GM_ConvertExcentricHzVDist is that only the directions hz and v are measured and recorded to the structure GM_Measurements_Type.

**Note** Used formula: see Appendix, Geodesy Math. Formulas.
5. Standard Functions

5.6. ExCentMeas in eccentric observation

- **ExCentMeas in eccentric observation**

<table>
<thead>
<tr>
<th>ExCentMeas in eccentric observation</th>
<th>eccentric observation</th>
</tr>
</thead>
</table>

- **ExCentElems in eccentric observation**

- **Center**
  - coordinate of the centre

- **Target**
  - coordinate of the target

- **CentMeas**
  - onto the centre re-centred measurement-element

**Return Codes**

- **GM_IDENTICAL_PTS**
  - Center and Target are identical. Calculation is not possible.

**Example**

Calculate the point in the circle.

```basic
DIM StationPt AS GM_Point_Type
DIM TargetPt AS GM_Point_Type
DIM ExcElems AS GM_Excenter_Elems_Type
DIM ExcenterMeas AS GM_Measurements_Type
DIM CenterMeas AS GM_Measurements_Type

'initialize StationPt, TargetPt,
' ExcElems, ExcenterMeas

GM_ConvertExcentricHzV( StationPt, TargetPt,
                        ExcElems, ExcenterMeas,
                        CenterMeas )
```

### 5.6.28 GM_ConvertExcentricHzVDist

**Description**

Re-centration of hz- and v-direction and distance.

**Declaration**

```basic
GM_ConvertExcentricHzVDist(  
ExCentMeas AS GM_Measurements_Type,  
ExCentElems AS GM_Excenter_Elems_Type,  
CentMeas AS GM_Measurements_Type )
```

**Remarks**

With this function, the measured values (which are measured to the excenter) could be re-centred to the centre. The difference to the function GM_ConvertExcentricHzV is, that in addition
to the directions hz and v, the slope distance to the target point is measured and recorded to the structure GM_Measurements_Type.

**Note** Used formula: see Appendix, Geodesy Math. Formulas.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExCentMeas</td>
<td>in eccentric observation</td>
</tr>
<tr>
<td>ExCentElems</td>
<td>in height difference between the centre and the excenter [m] and horizontal distance between the centre and the excenter [m]</td>
</tr>
<tr>
<td>CentMeas</td>
<td>out onto the centre re-centred measurement-element</td>
</tr>
</tbody>
</table>

### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>always OK</td>
</tr>
</tbody>
</table>

### Example

Calculate the point in the circle.
DIM ExcElems AS GM_Excenter_Elems_Type
DIM ExcenterMeas AS GM_Measurements_Type
DIM CenterMeas AS GM_Measurements_Type

'initialize ExcElems, ExcenterMeas
GM_ConvertExcentricHzVDist( ExcElems,
ExcenterMeas,
CenterMeas)

5.6.29 GM_ConvertPressure

**Description**  Conversion of pressure from one system into the other.

**Declaration**
```
GM_ConvertPressure(
    ByVal iOldSys AS Integer,
    ByVal dPresOldSys AS Double,
    ByVal iNewSys AS Integer,
    dPresNewSys AS Double )
```

**Remarks** This function is converting pressure-values from one standard system into the other.

**Note** Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**

- `iOldSys` **in** standard system of the given pressure
  - GM_MM_HG mercury column [mm]
  - GM_M_BAR millibar
  - GM_ATMOS atmosphere

- `dPresOldSys` **in** pressure to convert

- `iNewSys` **in** standard system of the wanted pressure

- `dPresNewSys` **out** converted pressure

**Return Codes**

- GM_INVALID_PRES_SYSTEM One of the pressure standard systems was invalid.
  - There was no conversion. The
recovered value was not defined.

Example  Convert dPresOldSys from atmosphere to millibar.

```
DIM dPresOldSys AS Double
DIM dPresNewSys AS Double
DIM iOldsys  AS Integer
DIM iNewsys  AS Integer

'initialize values
iOldsys   = GM_ATMOS
dPresOldSys = 1.0
iNewsys   = GM_M_BAR

GM_ConvertPressure( iOldsys, dPresOldSys, iNewsys, dPresNewSys )
```

5.6.30  GM_ConvertTemp

Description  Conversion of temperature from one system into the other.

Declaration  GM_ConvertTemp(
  ByVal iOldSys   AS Integer,
  ByVal dTempOldSys AS Double
  ByVal iNewSys   AS Integer,
  dTempNewSys AS Double)

Remarks  This function is converting temperature-values from one standard system into the other.

Note  Used formula: see Appendix, Geodesy Math. Formulas.
Parameters

ioOldSys in  standard system of the given temperature
    GM_KELVIN  Kelvin
    GM_CELSIUS  Celsius
    GM_FAHRENHEIT  Fahrenheit

dTempOldSys in  temperature to convert
iNewSys in  standard system of the wanted temperature
dTempNewSys out  converted temperature

Return Codes

GM_INVALID_TEMP_SYSTEM  One of the temperature standard systems was invalid.
                        There was no conversion. The recovered value was not defined.

Example

Convert dTempOldSys from [Celsius] to [Fahrenheit].

DIM dTempOldSys AS Double
DIM dTempNewSys AS Double
DIM iOldSys AS Integer
DIM iNewSys AS Integer

'initialize values
iOldSys = GM_CELSIUS
dTempOldSys = 1.8
iNewSys = GM_FAHRENHEIT

GM_ConvertTemp( iOldSys, dTempOldSys,
                iNewSys, dTempNewSys )
5.6.31 GM_ConvertVDirection

**Description**
Conversion of v-directions from one system into the other.

**Declaration**
```
GM_ConvertVDirection(
    byVal OldSys AS Integer,
    byVal dVOldSys AS Double,
    byVal NewSys AS Integer,
    dVNewSys AS Double )
```

**Remarks**
This function is converting v-distance-values from one standard system into the other.

**Note**
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**
- `iOldSys` in standard system of the given v-direction
  - GM_ZENITH: zenith direction [rad]
  - GM_NADIR: nadir direction [radians]
  - GM_V_ANGLE_RAD: height angle [rad]
  - GM_V_ANGLE_PERCENT: height angle [%]
- `dVOldSys` in v-distance to convert
- `iNewSys` in standard system of the wanted v-distance
- `dVNewSys` out converted v-distance

**Return Codes**
- GM_INVALID_V_SYSTEM: One of the standard systems was invalid.
- There was no conversion. The recovered value was not defined
Example
Convert dVOldSys.

```basic
DIM dVOldSys AS Double
DIM dVNewSys AS Double
DIM iOldsys AS Integer
DIM iNewsys AS Integer

'initialize values
iOldsys = GM_ZENITH
dVOldSys = Pi
iNewsys = GM_V_ANGLE_RAD

GM_ConvertVDirection( iOldsys, dVOldSys,
                       iNewsys, dVNewSys )
```

### 5.6.32 GM_ConvertSexaDec

**Description**
Conversion of value from the sexagesimal into the decimal system.

**Declaration**
```basic
GM_ConvertSexaDec(
    ByVal dValueSexa AS Double,
    dValueDec AS Double
)
```

**Remarks**
This function is converting the value from the sexagesimal into the decimal system.

**Note**
Used formula: see Appendix, Geodesy Math. Formulas.

**Parameters**
- `dValueSexa` in sexagesimal value
- `dValueDec` out decimal value

**Return Codes**
- `RC_OK` always OK

**Example**
Convert the angle. The following variables have to be defined:

```basic
DIM dAngleDec AS Double

GM_ConvertSexaDec( 99.9, dAngleDec )
```
5.6.33 GM_TransformPoints

Description  Transformation of point.

Declaration  

\[
\text{GM_TransformPoints(}
\quad \text{OldPt AS GM_Point_Type,}
\quad \text{Param AS GM_4Transform_Param_Type,}
\quad \text{NewPt AS GM_Point_Type}
\text{)}
\]

Remarks  This function transforms a point from one coordinate system into an other after the transformation parameters are calculated. In addition the coordinate systems have to be in the same sense.

Note  Used formula: see Appendix, Geodesy Math. Formulas.

Parameters  

- **OldPt** in  point to be transformed
- **Param** in  transformation parameters
- **NewPt** out  transformed point

Return Codes  

- **RC_OK**  always OK

Example  Calculate the point in the circle.

```
DIM OldPt AS GM_Point_Type
DIM NewPt AS GM_Point_Type
DIM Param AS GM_4Transform_Param_Type

'initialize OldPt, NewPt, Param
GM_TransformPoints( OldPt, Param, NewPt )
```

5.6.34 GM_SamePoint

Description  Test if two points are equal.

Declaration  

\[
\text{GM_SamePoint( Point1 AS GM_Point_Type,}
\quad \text{Point2 AS GM_Point_Type,}
\quad \text{lSame AS Logical)}
\]

Example  Calculate the point in the circle.

```
DIM OldPt AS GM_Point_Type
DIM NewPt AS GM_Point_Type
DIM Param AS GM_4Transform_Param_Type

'initialize OldPt, NewPt, Param
GM_TransformPoints( OldPt, Param, NewPt )
```
Remarks
The function checks, if the two given points are the same (coordinate difference < GM_THRESHOLD).

<table>
<thead>
<tr>
<th>Note</th>
<th>Height is ignored in the comparison.</th>
</tr>
</thead>
</table>

Parameters
- **Point1** in 1. point to be tested
- **Point2** in 2. point
- **lSame** out TRUE: difference of each coordinate < GM_THRESHOLD

Return Codes
- **RC_OK** always OK

Example
Test if the 2 points are the same.
```
DIM Pt1 AS GM_Point_Type
DIM Pt2 AS GM_Point_Type
DIM lSame AS Logical

'initialize Pt1, Pt2
GM_TransformPoints( Pt1, Pt2, lSame )
```

5.6.35 GM_CopyPoint

Description
Copy the contents of a point.

Declaration
```
GM_CopyPoint( Pt1 AS GM_Point_Type, 
              Pt2 AS GM_Point_Type )
```

Remarks
Copy the contents of Pt1 to Pt2.

Parameters
- **Pt1** in point to be copied
- **Pt2** out taken copy

Return Codes
- **RC_OK** always OK
Example

Copy point.

```
DIM Pt1 AS GM_Point_Type
DIM Pt2 AS GM_Point_Type

'initialize Pt1, Pt2
GM_CopyPoint( Pt1, Pt2 )
```

5.6.36 GM_AngleFromThreePoints

Description

Calculate enclosed angle from three points.

Declaration

```
GM_AngleFromThreePoints(
    StartPoint AS GM_Point_Type,
    Vertex AS GM_Point_Type,
    EndPoint AS GM_Point_Type,
    dAngle AS Double )
```

Remarks

This function calculates the angle enclosed by the 3 given points (counter clockwise).

Note

The height is ignored.

Parameters

- **StartPoint** in 1. point for angle definition
- **Vertex** in 2. point (middle)
- **EndPoint** in 3. point
- **dAngle** out calculated enclosed angle

Return Codes

- **GM_IDENTICAL_PTS** at least 2 points are identical (GM_SamePoint), calculation not possible
Example Calculate the point in the circle.

DIM StartPt AS GM_Point_Type
DIM Vertex AS GM_Point_Type
DIM EndPt AS GM_Point_Type
DIM dAngle AS Double

'initialize StartPt, Vertex, EndPt

GM.AngleFromThreePoints( StartPt, Vertex, EndPt, dAngle )

5.6.37 GM_AdjustAngleFromZeroToTwoPi

Description Normalise angle to \( [0, 2\pi] \).

Declaration

\[ \text{GM_AdjustAngleFromZeroToTwoPi}( \text{dAngle} \text{ AS Double} ) \]

Remarks This function adjusts the angle to be \( 0 \leq \text{dAngle} < 2\pi \).

Parameters

- dAngle in out angle to be transformed

Return Codes

- RC_OK always OK

Example Convert angle.

DIM dAngle AS Double

'initialize dAngle
dAngle = 4*Pi

GM_AdjustAngleFromZeroToTwoPi( dAngle )
5.6.38 GM_LineAzi

**Description**  Calculate azimuth of a line.

**Declaration**  
```vbnet
GM_LineAzi( Line AS GM_Line_Type,
            dAzimuth AS Double )
```

**Remarks**  This function calculates the azimuth of the line from `Line.FirstPt`.

**Parameters**  
- `Line`: a line
- `dAzimuth`: the azimuth of the line from `Line.FirstPt`

**Return Codes**
- `GM_IDENTICAL_PTS`: The points in the line are identical. Calculation not possible.

**Example**  Calculate the azimuth of the line.
```vbnet
DIM Line AS GM_Line_Type
DIM dAzi AS Double
'initialize Line
GM_LineAzi( Line, dAzi )
```

5.6.39 GM_MathOrSurveyorsAngleConv

**Description**  Adjusts a math angle in radians to a surveyors angle in radians or vice versa.

**Declaration**  
```vbnet
GM_MathOrSurveyorsAngleConv( dAngle AS Double )
```

**Remarks**  Converts the angle from surveyors convention (azimuth) to a math direction (x/y axis) or vice versa.

**Parameters**  
- `dAngle`: angle to be transformed
Return Codes

RC_OK always OK

Example

Calculate the point in the circle.

```
DIM dAngle AS Double
 dAngle = Pi
 GM_MathOrSurveyorsAngleConv( dAngle )
```

5.6.40 GM_Traverse3D

Description

Convert a point in polar coordinates to Cartesian coordinates.

Declaration

```
GM_Traverse3D(
    StartPt AS GM_Point_Type,
    Polar AS GM_Measurements_Type,
    NewPt AS GM_Point_Type)
```

Remarks

This function converts a point given in polar coordinates relative to `StartPt` to Cartesian coordinates (`NewPt`).

**Note** Used formula: see Appendix, Geodesy Math. Formulas.

Parameters

- **StartPt** in relative origin for `Polar`
- **Polar** in point in polar coordinates
- **NewPt** out transformed point in Cartesian coordinates

Return Codes

RC_OK always OK
Example

Convert a point in polar to Cartesian coordinates.

```plaintext
DIM StartPt AS GM_Point_Type
DIM NewPt AS GM_Point_Type
DIM Polar AS GM_Measurements_Type

'initialize StartPt, Polar
GM_Traverse3D( StartPt, Polar, NewPt )
```

### 5.6.41 GM_InitQXXMatrix

**Description**
Initialise the QXX-Matrix for a point structure.

**Declaration**

```
GM_InitQXXMatrix( Point AS GM_Point_Type )
```

**Remarks**
This function sets all values in the QXX-matrix of a point to zero.

**Parameters**

- **Point** in out: point of which the QXX-matrix is to be initialised

**Return Codes**

- **RC_OK**: always OK

Example

Initialise QXX-matrix of a point.

```plaintext
DIM Point AS GM_Point_Type

GM_InitQXXMatrix( Point )
```

### 5.6.42 GM_CalcAziZenAndDist
Description
Convert a point given in Cartesian coordinates to polar coordinates.

Declaration
GM_CalcAziZenAndDist(
    Point AS GM_Point_Type,
    Point2 AS GM_Point_Type,
    Polar AS GM_Measurements_Type
)

Remarks
This function converts a point given in Cartesian coordinates relative to Pt1 to polar coordinates (Polar).

Note
Used formula: see Appendix, Geodesy Math. Formulas.

Parameters
- Point1 in relative origin for Point2
- Point2 in point in Cartesian coordinates
- Polar out transformed point in polar coordinates

Return Codes
- RC_OK always OK

Example
Convert a point in Cartesian to polar coordinates.

```plaintext
dim point1 as gm_point_type
    dim point2 as gm_point_type
    dim polar as gm_measurements_type

    'initialize point1, point2
    gm_calcAzizenAndDist( point1, point2, polar )
```

6. System Functions

6.1 MMI Functions

6.1.1 Summarising Lists of MMI Types and Procedures

6.1.2 MMI Data Types

6.1.3 MMI_CreateMenuItem

6.1.4 MMI_CreateGBMenu

6.1.5 MMI_CreateGBMenuItem

6.1.6 MMI_CreateGBMenuStr

6.1.7 MMI_CreateGBMenuItemStr

6.1.8 MMI_DeleteGBMenu

6.1.9 MMI_SelectGBMenuItem

6.1.10 MMI_AddGBMenuButton

6.1.11 MMI_CreateTextDialog

6.1.12 MMI_CreateGraphDialog

6.1.13 MMI_DeleteDialog

6.1.14 MMI_CheckButton

6.1.15 MMI_GetButton

6.1.16 MMI_AddButton

6.1.17 MMI_DeleteButton

6.1.18 MMI_PrintStr

6.1.19 MMI_PrintTok

6.1.20 MMI_PrintVal

6.1.21 MMI_PrintInt

6.1.22 MMI_InputStr

6.1.23 MMI_InputVal

6.1.24 MMI_InputInt

6.1.25 MMI_InputList

6.1.26 MMI_FormatVal

6.1.27 MMI_WriteMsg
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.28 MMI_WriteMsgStr</td>
<td>6-47</td>
</tr>
<tr>
<td>6.1.29 MMI_DrawLine</td>
<td>6-49</td>
</tr>
<tr>
<td>6.1.30 MMI_DrawRect</td>
<td>6-50</td>
</tr>
<tr>
<td>6.1.31 MMI_DrawCircle</td>
<td>6-52</td>
</tr>
<tr>
<td>6.1.32 MMI_DrawText</td>
<td>6-53</td>
</tr>
<tr>
<td>6.1.33 MMI_DrawBusyField</td>
<td>6-54</td>
</tr>
<tr>
<td>6.1.34 MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong</td>
<td>6-55</td>
</tr>
<tr>
<td>6.1.35 MMI_StartVarBeep</td>
<td>6-55</td>
</tr>
<tr>
<td>6.1.36 MMI_SwitchVarBeep</td>
<td>6-56</td>
</tr>
<tr>
<td>6.1.37 MMI_GetVarBeepStatus</td>
<td>6-57</td>
</tr>
<tr>
<td>6.1.38 MMI_SwitchAFKey</td>
<td>6-58</td>
</tr>
<tr>
<td>6.1.39 MMI_SwitchIconsBeep</td>
<td>6-59</td>
</tr>
<tr>
<td>6.1.40 MMI_SetAngleRelation</td>
<td>6-60</td>
</tr>
<tr>
<td>6.1.41 MMI_GetAngleRelation</td>
<td>6-61</td>
</tr>
<tr>
<td>6.1.42 MMI_SetVAngleMode</td>
<td>6-61</td>
</tr>
<tr>
<td>6.1.43 MMI_GetVAngleMode</td>
<td>6-62</td>
</tr>
<tr>
<td>6.1.44 MMI_SetAngleUnit</td>
<td>6-62</td>
</tr>
<tr>
<td>6.1.45 MMI_GetAngleUnit</td>
<td>6-64</td>
</tr>
<tr>
<td>6.1.46 MMI_SetDistUnit</td>
<td>6-64</td>
</tr>
<tr>
<td>6.1.47 MMI_GetDistUnit</td>
<td>6-66</td>
</tr>
<tr>
<td>6.1.48 MMI_SetPressUnit</td>
<td>6-66</td>
</tr>
<tr>
<td>6.1.49 MMI_GetPressUnit</td>
<td>6-68</td>
</tr>
<tr>
<td>6.1.50 MMI_SetTempUnit</td>
<td>6-68</td>
</tr>
<tr>
<td>6.1.51 MMI_GetTempUnit</td>
<td>6-69</td>
</tr>
<tr>
<td>6.1.52 MMI_SetDateFormat</td>
<td>6-70</td>
</tr>
<tr>
<td>6.1.53 MMI_GetDateFormat</td>
<td>6-71</td>
</tr>
<tr>
<td>6.1.54 MMI_SetTimeFormat</td>
<td>6-71</td>
</tr>
<tr>
<td>6.1.55 MMI_GetTimeFormat</td>
<td>6-72</td>
</tr>
<tr>
<td>6.1.56 MMI_SetCoordOrder</td>
<td>6-73</td>
</tr>
<tr>
<td>6.1.57 MMI_GetCoordOrder</td>
<td>6-74</td>
</tr>
<tr>
<td>6.1.58 MMI_SetLanguage</td>
<td>6-74</td>
</tr>
<tr>
<td>6.1.59 MMI_GetLanguage</td>
<td>6-75</td>
</tr>
<tr>
<td>6.1.60 MMI_GetLangName</td>
<td>6-76</td>
</tr>
</tbody>
</table>
6.2 BASIC APPLICATIONS BAP

6.2.1 Summarizing Lists of BAP Types and Procedures
6.2.2 BAP_SetAccessoriesDlg
6.2.3 BAP_MeasDistAngle
6.2.4 BAP_MeasRec
6.2.5 BAP_FineAdjust
6.2.6 BAP_SearchPrism
6.2.7 BAP_SetManDist
6.2.8 BAP_SetPpm
6.2.9 BAP_SetPrism
6.2.10 BAP_SetMeasPrg
6.2.11 BAP_GetMeasPrg
6.2.12 BAP_PosTelescope
6.2.13 BAP_SetHz

6.3 Measurement Functions TMC

6.3.1 Summarizing Lists of TMC Types and Procedures
6.3.2 TMC Data Structures
6.3.3 TMC_DoMeasure
6.3.4 TMC_GetPolar
6.3.5 TMC_GetCoordinate
6.3.6 TMC_GetAngle
6.3.7 TMC_GetAngle_WInc
6.3.8 TMC_QuickDist
6.3.9 TMC_GetSimpleMea
6.3.10 TMC_Get/SetAngleFaceDef
6.3.11 TMC_Get/SetHzOffset
6.3.12 TMC_Get/SetDistPpm
6.3.13 TMC_Get/SetGeomProjection
6.3.14 TMC_Get/SetGeomReduction
6.3.15 TMC_Get/SetAtmCorr
6.3.16 TMC_Get/SetHeight
6.3.17 TMC_Get/SetRefractiveCorr
6.3.18 TMC_Get/SetRefractiveMethod
6.3.19 TMC_Get/SetStation ................................................................. 6-125
6.3.20 TMC_IFDistTapeMeasured ....................................................... 6-125
6.3.21 TMC_SetHandDist .................................................................. 6-126
6.3.22 TMC_SetDistSwitch ................................................................. 6-127
6.3.23 TMC_GetDistSwitch ................................................................ 6-127
6.3.24 TMC_SetOffsetDist ................................................................. 6-128
6.3.25 TMC_GetOffsetDist ................................................................ 6-129
6.3.26 TMC_IFOffsetDistMeasured .................................................... 6-129
6.3.27 TMC_GetFace1 ....................................................................... 6-130
6.3.28 TMC_SetAngSwitch................................................................. 6-130
6.3.29 TMC_GetAngSwitch ................................................................ 6-131
6.3.30 TMC_SetInclineSwitch............................................................. 6-131
6.3.31 TMC_GetInclineSwitch ............................................................ 6-132
6.3.32 TMC_GetInclineStatus ............................................................. 6-132

6.4 Functions for GSI .................................................................................... 6-134
6.4.1 Summarizing Lists of GSI Types and Procedures ....................... 6-134
6.4.2 Constants for WI values .............................................................. 6-136
6.4.3 Constants for Measurement Dialog Definition ............................. 6-139
6.4.4 Relationship of GSI_ID’s to GSI_PAR’s ......................................... 6-143
6.4.5 Data Structures for GSI Functions ................................................ 6-146
6.4.6 GSI_GetRunningNr ..................................................................... 6-148
6.4.7 GSI_SetRunningNr ..................................................................... 6-149
6.4.8 GSI_GetIndivNr ......................................................................... 6-149
6.4.9 GSI_SetIndivNr ......................................................................... 6-150
6.4.10 GSI_IsRunningNr ..................................................................... 6-150
6.4.11 GSI_SetIVPtNrStatus ............................................................... 6-151
6.4.12 GSI_IncPNumber ...................................................................... 6-152
6.4.13 GSI_Coding .............................................................................. 6-152
6.4.14 GSI_SelectCode ........................................................................ 6-153
6.4.15 GSI_GetQCodeAvailable ........................................................... 6-153
6.4.16 GSI_SetQCodeMode .................................................................. 6-154
6.4.17 GSI_ExecQCoding ................................................................... 6-154
6.4.18 GSI_SetRecOrder ...................................................................... 6-156
GeoBASIC Reference Manual

6. System Functions

6.4.19 GSI_GetRecOrder................................................................. 6-156
6.4.20 GSI_QuickSet .................................................................. 6-157
6.4.21 GSI_SetRecPath................................................................. 6-157
6.4.22 GSI_GetRecPath................................................................. 6-158
6.4.23 GSI_SetDataPath............................................................... 6-159
6.4.24 GSI_GetDataPath............................................................... 6-160
6.4.25 GSI_GetWiEntryText ......................................................... 6-160
6.4.26 GSI_GetWiEntry................................................................. 6-161
6.4.27 GSI_SetWiEntry ............................................................... 6-162
6.4.28 GSI_GetRecMask ............................................................. 6-163
6.4.29 GSI_SetRecMask ............................................................. 6-164
6.4.30 GSI_SetRecMaskNr........................................................... 6-165
6.4.31 GSI_GetRecMaskNr ......................................................... 6-166
6.4.32 GSI_DefineRecMaskDlg ................................................... 6-166
6.4.33 GSI_ManCoordDlg ............................................................ 6-166
6.4.34 GSI_ImportCoordDlg ....................................................... 6-169
6.4.35 GSI_SetLineSysMDlg ....................................................... 6-172
6.4.36 GSI_GetLineSysMDlg ....................................................... 6-173
6.4.37 GSI_GetMDlgNr ............................................................... 6-174
6.4.38 GSI_GetMDlgNr ............................................................... 6-175
6.4.39 GSI_CreateMDlg ............................................................. 6-175
6.4.40 GSI_SetLineMDlg ............................................................ 6-177
6.4.41 GSI_SetLineMDlgText ...................................................... 6-178
6.4.42 GSI_SetLineMDlgPar ....................................................... 6-179
6.4.43 GSI_UpdateMDlg ............................................................. 6-181
6.4.44 GSI_DefineMDlg ............................................................. 6-182
6.4.45 GSI_UpdateMeasurement ................................................. 6-182
6.4.46 GSI_Measure ................................................................. 6-183
6.4.47 GSI_ExecuteAutoDist ...................................................... 6-184
6.4.48 GSI_CheckTracking ......................................................... 6-184
6.4.49 GSI_RecordRecMask ....................................................... 6-185

6.5 Central Service Functions CSV .................................................. 6-187
6.5.1 Summarizing Lists of CSV Types and Procedures ................. 6-187
6.5.2 Data Structures for the Central Service Functions ............... 6-189
6.5.3 CSV_GetDateTime .............................................................. 6-191
6.5.4 CSV_GetTemperature ....................................................... 6-192
6.5.5 CSV_GetInstrumentName ................................................... 6-192
6.5.6 CSV_GetInstrumentNo ....................................................... 6-193
6.5.7 CSV_GetInstrumentFamily ............................................... 6-193
6.5.8 CSV_GetSWVersion ......................................................... 6-194
6.5.9 CSV_GetGBIVersion .......................................................... 6-195
6.5.10 CSV_GetElapseSysTime .................................................... 6-196
6.5.11 CSV_GetSysTime ............................................................. 6-197
6.5.12 CSV_GetLRStatus ........................................................... 6-197
6.5.13 CSV_SetGuideLight .......................................................... 6-198
6.5.14 CSV_Laserpointer ............................................................ 6-199
6.5.15 CSV_MakePositioning ..................................................... 6-199
6.5.16 CSV_ChangeFace ............................................................ 6-200
6.5.17 CSV_SetLockStatus ......................................................... 6-201
6.5.18 CSV_GetLockStatus ........................................................ 6-201
6.5.19 CSV_LockIn ................................................................. 6-202
6.5.20 CSV_LockOut ................................................................. 6-203
6.5.21 CSV_SetATRStatus .......................................................... 6-204
6.5.22 CSV_GetATRStatus .......................................................... 6-204
6.5.23 CSV_Delay ................................................................. 6-205
6.5.24 CSV_SetTargetType ....................................................... 6-205
6.5.25 CSV_GetTargetType ....................................................... 6-206
6.5.26 CSV_SetPrismType .......................................................... 6-206
6.5.27 CSV_GetPrismType .......................................................... 6-207
6.5.28 CSV_SetLaserPlummet .................................................... 6-208
6.5.29 CSV_GetLaserPlummet .................................................... 6-209
6.5.30 CSV_CheckAltUserTask ..................................................... 6-210
6.5.31 CSV_ResetAltUserTask ..................................................... 6-210
6.5.32 CSV_SysCall ................................................................. 6-211
6.5.33 CSV_SysCallAvailable ................................................... 6-212
6.5.34 CSV_LibCall ................................................................. 6-213
6.5.35 CSV_LibCallAvailable ............................................................. 6-213
6.1 MMI FUNCTIONS

6.1.1 Summarising Lists of MMI Types and Procedures

6.1.1.1 Types

<table>
<thead>
<tr>
<th>Type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListArray</td>
<td>List field Data structure</td>
</tr>
<tr>
<td>sLine</td>
<td>Display line</td>
</tr>
</tbody>
</table>

6.1.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_AddButton</td>
<td>Add a Button to a dialog.</td>
</tr>
<tr>
<td>MMI_AddGBMenuButton</td>
<td>Adds a button to a menu</td>
</tr>
<tr>
<td>MMI_BeepAlarm</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_BeepLong</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_BeepNormal</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_CheckButton</td>
<td>Checks if a button was pressed.</td>
</tr>
<tr>
<td>MMI_CreateGBMenu</td>
<td>Creates a menu</td>
</tr>
<tr>
<td>MMI_CreateGBMenuItem</td>
<td>Creates an item to an existing menu</td>
</tr>
<tr>
<td>Str</td>
<td>Creates an item with a variable string</td>
</tr>
<tr>
<td>MMI_CreateGBMenuStr</td>
<td>Creates a menu with variable strings</td>
</tr>
<tr>
<td>MMI_CreateGraphDialog</td>
<td>Create and show a graphics dialog.</td>
</tr>
<tr>
<td>MMI_CreateMenuItem</td>
<td>Creates a menu item on the Theodolite menu.</td>
</tr>
<tr>
<td>MMI_CreateTextDialog</td>
<td>Create and show a text dialog.</td>
</tr>
<tr>
<td>MMI_DeleteButton</td>
<td>Delete a button from a dialog.</td>
</tr>
<tr>
<td>MMI_DeleteDialog</td>
<td>Deletes a dialog.</td>
</tr>
<tr>
<td>MMI_DeleteGBMenu</td>
<td>Deletes a menu</td>
</tr>
<tr>
<td>MMI_DrawBusyField</td>
<td>Shows or hides the Busy-Icon</td>
</tr>
<tr>
<td>MMI_DrawCircle</td>
<td>Draw a circle / ellipse.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>MMI_DrawLine</td>
<td>Draw a line.</td>
</tr>
<tr>
<td>MMI_DrawRect</td>
<td>Draw a rectangle.</td>
</tr>
<tr>
<td>MMI_DrawText</td>
<td>Draw / delete text.</td>
</tr>
<tr>
<td>MMI_FormatVal</td>
<td>Convert a value to a string.</td>
</tr>
<tr>
<td>MMI_GetAngleRelation</td>
<td>Request the current angle relationships.</td>
</tr>
<tr>
<td>MMI_GetAngleUnit</td>
<td>Return the currently displayed unit of angle.</td>
</tr>
<tr>
<td>MMI_GetButton</td>
<td>Get the button identifier of the pressed button.</td>
</tr>
<tr>
<td>MMI_GetCoordOrder</td>
<td>Retrieve the co-ordinate order.</td>
</tr>
<tr>
<td>MMI_GetDateFormat</td>
<td>Retrieves the date display format.</td>
</tr>
<tr>
<td>MMI_GetDistUnit</td>
<td>Return the currently displayed unit of distance.</td>
</tr>
<tr>
<td>MMI_GetLangName</td>
<td>Gets the name to a language number.</td>
</tr>
<tr>
<td>MMI_GetLanguage</td>
<td>Query the current language.</td>
</tr>
<tr>
<td>MMI_GetPressUnit</td>
<td>Return the currently displayed unit of pressure.</td>
</tr>
<tr>
<td>MMI_GetTempUnit</td>
<td>Return the currently displayed unit of temperature.</td>
</tr>
<tr>
<td>MMI_GetTimeFormat</td>
<td>This function retrieves the format used to display the time.</td>
</tr>
<tr>
<td>MMI_GetVAngleMode</td>
<td>Returns the V-Angle mode</td>
</tr>
<tr>
<td>MMI_GetVarBeepStatus</td>
<td>Read the switch status for a variable signal beep.</td>
</tr>
<tr>
<td>MMI_InputInt</td>
<td>Get an integer input value in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputList</td>
<td>Shows a list field in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputStr</td>
<td>Get a string input in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputVal</td>
<td>Get a numerical input value in a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintInt</td>
<td>Print an integer value on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintStr</td>
<td>Print a string on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintTok</td>
<td>Print a token on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintVal</td>
<td>Print a value on a text dialog.</td>
</tr>
<tr>
<td>MMI_SelectGBMenuItem</td>
<td>Select a menu item</td>
</tr>
<tr>
<td>MMI_SetAngleRelation</td>
<td>Set the angle relationship.</td>
</tr>
<tr>
<td>MMI_SetAngleUnit</td>
<td>Set the displayed unit of angle.</td>
</tr>
<tr>
<td>MMI_SetCoordOrder</td>
<td>Set the co-ordinate order.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MMI_SetDateFormat</td>
<td>Set the date display format.</td>
</tr>
<tr>
<td>MMI_SetDistUnit</td>
<td>Set the displayed unit of distance.</td>
</tr>
<tr>
<td>MMI_SetLanguage</td>
<td>Set the display language.</td>
</tr>
<tr>
<td>MMI_SetPressUnit</td>
<td>Set the displayed unit of pressure.</td>
</tr>
<tr>
<td>MMI_SetTempUnit</td>
<td>Set the displayed unit of temperature.</td>
</tr>
<tr>
<td>MMI_SetTimeFormat</td>
<td>Set the time display format.</td>
</tr>
<tr>
<td>MMI_SetVAngleMode</td>
<td>Set the V-Angle mode.</td>
</tr>
<tr>
<td>MMI_StartVarBeep</td>
<td>Start beep sequences with configurable interrupts.</td>
</tr>
<tr>
<td>MMI_SwitchAFKey</td>
<td>Switch aF… key</td>
</tr>
<tr>
<td>MMI_SwitchIconsBeep</td>
<td>Switches measurement icons and special beeps</td>
</tr>
<tr>
<td>MMI_SwitchVarBeep</td>
<td>Switch a varying beep.</td>
</tr>
<tr>
<td>MMI_WriteMsg</td>
<td>Output to a message window. Parameter is a token.</td>
</tr>
<tr>
<td>MMI_WriteMsgStr</td>
<td>Output to a message window. Parameter is a string.</td>
</tr>
</tbody>
</table>

### 6.1.2 MMI Data Types

#### 6.1.2.1 ListArray – List field data structure

**Description**  
This array is used for list fields and consists of 200 elements of the type STRING30.

**Note**  
Each variable of this data type reserves 6400 Bytes.

#### 6.1.2.2 sLine – Display line

**Description**  
This type is used to define a string with 29 characters, which is necessary to print variable strings on the display. The length depends on the actual display width, which is 29 for TPS1100 instruments.
6.1.3 MMI_CreateMenuItem

Description
Creates a system menu item on the Theodolite menu to establish
the invocation of a GeoBASIC application.

Declaration
MMI_CreateMenuItem(
   BYVAL sAppName AS String,
   BYVAL sFuncName AS String,
   BYVAL iMenuNum AS Integer,
   BYVAL sMenuText AS _Token )

Remarks
The CreateMenuItem creates a menu item in a system menu
with the text MenuText on the chosen entry point MenuNum in
the menu-system. By clicking the new menu item on the
Theodolite, the subroutine with the name FuncName in the
ProgramAppName will be executed. The number of applications
which can be loaded at a time are limited to 25. The maximum
number of entry points over all applications (C and GeoBASIC
applications) is 50. All GLOBAL declared subroutines count as
entry points. Be aware of the fact that the interpreter and a
possible Coding function also count for the number of application.
The same is true for any C-application which has been loaded onto
the TPS.

Note
The subroutine denoted in sFuncName must be declared
as GLOBAL.
The intended use for this procedure is during the
installation phase only!

Parameters
   sAppName in The name of the program where the
              function or subroutine is defined.
   sFuncName in The name of the global function or
              subroutine to be called.
   iMenuNum in Defines in which menu the menu-entry is
               generated. There are three possible menus
               where a menu item can be added. For
               multiple menu items the menus can be
               combined with ‘+’-operator.
valid menus | meaning
---|---
MMI_MENU_PROGRAMS | Add to menu "Main menu"
MMI_MENU_PROGMENU | Add to "PROG" - Key menu
MMI_MENU_AUTOEXEC | Add to menu "Autoexec"

sMenuText in | The text of the menu-entry which should be displayed on the Theodolite.

Return-Codes

**RC_OK** | Successful termination.

**Note** | Since this procedure will be called during installation phase you do not have the possibility to do any error handling. Only the loader will report an error which may be caused by an erroneous call.

Example

The example uses the MMI_CreateMenuItem routine to create a menu entry named "START THE PROGRAM" under the main menu. The function "Main" in the GeoBASIC program "ExampleProgram" will be called when this menu item is selected.

```c
MMI_CreateMenuItem( "ExampleProgram", "Main", MMI_MENU_PROGRAMS, "START THE PROGRAM" )
```

6.1.4 **MMI_CreateGBMenu**

**Description** | Creates a menu.

**Declaration**

```c
MMI_CreateGBMenu( 
    BYVAL sMenuName AS _Token, 
    iMenuId AS Integer )
```

**Remarks** | This routine creates an empty menu and the caption sMenuName. The function MMI_CreateGBMenuItem adds items to a menu.
Note Before terminating a GeoBASIC program, all menus must be deleted.

The GeoBASIC menus system has the following limitations:

- The maximal number of menus for a GeoBASIC program is 5.
- The maximal number of items / menu is 49.
- The maximal number of items over all menus plus menus is 254.

Parameters

- **sMenuName** in The caption of the menu.
- **iMenuId** out Returned menu identifier. It is the handle for using this menu.

Return-Codes

- **RC_OK** Successful termination.
- **MMI_NOMORE_MENUS** No more menus available

See Also

- MMI_CreateGBMenuItem, MMI_DeleteGBMenu, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example

The example creates a menu with a button. For a complete example see sample program MENU.GBS

```plaintext
CONST MHELP = "Help for measurement type...."
DIM iMenu AS Integer ' menu identifier
DIM iSelection AS Integer ' selected item
DIM iButton AS Integer ' used button

'Create main menu
MMI_CreateGBMenu("MEASUREMENT TYPE", iMenu)
```
'Create menu items - all items use
've the same help text
MMI_CreateGBMenuItem(iMenu, 
"Polygon", MHELP) 
MMI_CreateGBMenuItem(iMenu, 
"Border point", MHELP) 
MMI_CreateGBMenuItem(iMenu, 
"Situation point", MHELP) 

'Create the button supported in this menu
MMI_AddGBMenuButton(iMenu, MMI_F5_KEY, "EXIT ")

'show and execute menu
MMI_SelectGBMenuItem(iMenu, "TEST", 
iSelection, iButton)
SELECT CASE iSelection 
CASE 1 ' Polygon 
' ... 
CASE ELSE
    MMI_BeepAlarm() 
END SELECT 
MMI_DeleteGBMenu(iMenu)

6.1.5 MMI_CreateGBMenuItem

Description Creates an item in an existing menu.

Declaration 
MMI_CreateGBMenuItem( 
    BYVAL iMenuId AS Integer, 
    BYVAL sMenuItemName AS _Token, 
    BYVAL sHelpText AS _Token )

Remarks This function adds one menu item to an existing menu iMenuId. This item will be displayed as the last item.

Parameters 
iMenuId in Menu identifier
sMenuItemName in Displayed text
sHelpText in Help text; only visible if the help functionality of theodolite is enabled

Return-Codes
RC_OK  Successful termination.
BAS_MENU_ID_INVALID  Bad iMenuId
BAS_MENU_TABLE_FULL  No more free menu items

See Also  MMI_CreateGBMenu, MMI_DeleteGBMenu,
           MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example  see MMI_CreateGBMenu

6.1.6  MMI_CreateGBMenuStr

Description  Creates a menu with variable strings as menu name and menu items.

Declaration  MMI_CreateGBMenuStr(
    BYVAL sMenuName AS sLine,
    iMenuId AS Integer )

Remarks  This routine creates an empty menu and the caption sMenuName. sMenuName need not be constant, it can be generated during the execution of the program. The function MMI_CreateGBMenuItemStr adds items to this kind of menu.

Note  Before terminating a GeoBASIC program, all menus must be deleted.

    The GeoBASIC menus system has the following limitations:

    The maximal number of menus for a GeoBASIC program is 5.
    The maximal number of items / menu is 49.
    The maximal number of items over all menus plus menus is 254.

Parameters  

    sMenuName  in  The caption of the menu.
iMenuId out Returned menu identifier. It is the handle for using this menu.

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>MMI_NOMORE_MENUS</td>
<td>No more menus available</td>
</tr>
</tbody>
</table>

See Also

MMI_CreateGBMenuItemStr, MMI_DeleteGBMenu, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example

The example creates a menu with a button. The menu name is a composition with a constant string and the instrument name. The menu item names are extended with the current language name.

```plaintext
CONST MHELP = "Help for measurement type...."

DIM iMenu AS Integer ' menu identifier
DIM iSelection AS Integer ' selected item
DIM iButton AS Integer ' used button
DIM sMenuName AS sLine ' menu name
DIM sMenuItemName1 AS sLine ' menu item 1 name
DIM sMenuItemName2 AS sLine ' menu item 2 name
DIM iLangNr AS Integer ' language number
DIM sLangName AS String20 ' language name
DIM sInstrumentName AS String30 ' instrument name

' generate menu name
CSV_GetInstrumentName(sInstrumentName)
sMenuName = "Programs on " + sInstrumentName
' Create menu
MMI_CreateGBMenuStr(sMenuName, iMenu)
' generate menu item names
MMI_GetLanguage(iLangNr, sLangName)
sMenuItemName1 = "Polygon in " + sLangName
sMenuItemName2 = "Border point in " + sLangName
' Create menu items - all items use
' the same help text
MMI_CreateGBMenuItemStr(iMenu, sMenuItemName1, MHELP)
MMI_CreateGBMenuItemStr(iMenu, sMenuItemName2, MHELP)
```
'Create the button supported in this menu
MMI_AddGBMenuButton(iMenu, MMI_F5_KEY, "EXIT ")

'show and execute menu
MMI_SelectGBMenuItem(iMenu, "TEST",
iSelection, iButton)
SELECT CASE iSelection
   CASE 1 ' Polygon
   ... 
   CASE ELSE
      MMI_BeepAlarm()
   END SELECT
MMI_DeleteGBMenu(iMenu)

6.1.7  MMI_CreateGBMenuItemStr

Description  Creates an item with a variable string in an existing menu.

Declaration  MMI_CreateGBMenuItemStr(
              BYVAL iMenuId AS Integer,
              BYVAL sMenuItemName AS _Token )

Remarks  This routine adds one menu item to an existing menu iMenuId. This item will be displayed as the last item. The menu must be created with MMI_CreateGBMenuStr. sMenuItemName need not be constant, it can be generated during the execution of the program.

Parameters  

   iMenuId  in  Menu identifier
   sMenuItemName  in  Displayed text
   sHelpText  in  Help text; only visible if the help functionallity of the theodolite is enabled

Return-Codes

   RC_OK  Successful termination.
   BAS_MENU_ID_INVALID  Bad iMenuId
**6. System Functions**

### 6.1.8 MMI_DeleteGBMenu

**Description**
Deletes a menu.

**Declaration**
```plaintext
MMI_DeleteGBMenu( BYVAL iMenuId AS Integer )
```

**Remarks**
This function deletes the menu `iMenuId`.

**Parameters**
- `iMenuId` in Menu identifier

**Return-Codes**
- **RC_OK** Successful termination.
- **BAS_MENU_ID_INVALID** Bad `iMenuId`

**See Also**
- `MMI_CreateGBMenu`, `MMI_CreateGBMenuItem`, `MMI_SelectGBMenuItem`, `MMI_AddGBMenuButton`

**Example**
See `MMI_CreateGBMenu`

### 6.1.9 MMI_SelectGBMenuStr

**Description**
Select a menu item.

**Declaration**
```plaintext
MMI_SelectGBMenuStr(  
    BYVAL iMenuId AS Integer,  
    BYVAL sCaptionLeft AS _Token,  
    iSelItem AS Integer,  
    iButtonId AS Integer )
```

**Remarks**
This function shows and executes a menu `iMenuId` and returns the selected item `iSelItem` or pressed button `iButtonId`.

**See Also**
- `MMI_CreateGBMenu`, `MMI_CreateGBMenuItem`, `MMI_SelectGBMenuItem`, `MMI_AddGBMenuButton`

**Example**
See `MMI_CreateGBMenu`

---

BAS_MENU_TABLE_FULL

No more free menu items

**See Also**
- `MMI_CreateGBMenuStr`, `MMI_DeleteGBMenu`, `MMI_SelectGBMenuItem`, `MMI_AddGBMenuButton`

**Example**
see `MMI_CreateGBMenuStr`
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMenuId</td>
<td>Menu identifier</td>
</tr>
<tr>
<td>sCaptionLeft</td>
<td>The maximal five-character long part of the title bar displayed left of the menu title, with a separation symbol.</td>
</tr>
<tr>
<td>iSelItem</td>
<td>Selected item</td>
</tr>
<tr>
<td>iButtonItem</td>
<td>Pressed button</td>
</tr>
</tbody>
</table>

Return-Codes

- RC_OK: Successful termination.
- BAS_MENU_ID_INVALID: Bad iMenuId

See Also

MMI_CreateGBMenu, MMI_CreateGBMenuItem, MMI_DeleteGBMenu, MMI_AddGBMenuButton

Example

see MMI_CreateGBMenu

6.1.10 MMI_AddGBMenuButton

Description

Adds a button to a menu.

Declaration

```plaintext
MMI_AddGBMenuButton(
    BYVAL iMenuId AS Integer,
    BYVAL iButtonId AS Integer,
    BYVAL sCaption AS _Token )
```

Remarks

This function adds a button with the identifier iButtonId to the menu iMenuId and shows the caption sCaption.
6. System Functions

### Parameters

- **iMenuId in**
  - Menu identifier
  - Identifier of the button to be added.
  - Valid buttons are `MMI_F1_KEY..MMI_F6_KEY` and `MMI_SHF2_KEY..MMI_SHF6_KEY`.
- **iButtonId in**
  - Identifier of the button to be added.
- **sCaption in**
  - Text placed onto the button (max. 5 characters)

### Return-Codes

- **RC_OK**
  - Successful termination.
- **BAS_MENU_ID_INVALID**
  - Bad iMenuId

### See Also

- `MMI_CreateGBMenu`
- `MMI_CreateGBMenuItem`
- `MMI_DeleteGBMenu`
- `MMI_SelectGBMenuItem`

### Example

See `MMI_CreateGBMenu`

---

#### 6.1.11 MMI_CreateTextDialog

**Description**

Create and show a text dialog.

**Declaration**

```basic
MMI_CreateTextDialog(
    BYVAL iLines AS Integer,
    BYVAL sCaptionLeft AS _Token,
    BYVAL sCaptionRight AS _Token,
    BYVAL sHelptext AS _Token )
```

**Remarks**

The routine creates and shows a dialog with `iLines` lines, the left part of the title bar `sCaptionLeft`, the caption `sCaptionRight` and the help text `sHelptext`. Only one text dialog can exist at the same time. If `MMI_CreateTextDialog` is called while already a text dialog or a measurement dialog exists, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

**Note**
Only a text dialog or a measurement dialog is valid at a time. They cannot be defined at the same time. A graphic dialog overrides a text or measurement dialog but does not delete the definition of it.

On the dialog field strings, numerical values and list fields can be displayed or edited using the routines `MMI_PrintStr`, `MMI_PrintVal`, `MMI_PrintInt`, `MMI_InputStr`, `MMI_InputVal`, `MMI_InputInt` and `MMI_InputList`.

**Parameters**

- **iLines in** The number of lines of the dialog. There are up to 12 lines possible. If the dialog has more than 6 lines, a scrollbar on the right side appears and it is possible to scroll up and down with the cursor keys.

- **sCaptionLeft in** The maximal five-character long part of the title bar displayed left of the CaptionRight, with a separation symbol.

- **sCaptionRight in** The caption of the dialog.

- **sHelpText in** This text is shown, when the help button SHIFT-F1 is pressed and the help functionality of the theodolite is enabled.

**Return-Codes**

- **RC_OK** Successful termination.

**See Also**

- `MMI_DeleteDialog`, `MMI_CreateGraphDialog`, `GSI_CreateMDlg`, `MMI_PrintVal`, `MMI_PrintStr`, `MMI_PrintTok`, `MMI_PrintInt`, `MMI_InputVal`, `MMI_InputStr`, `MMI_InputInt`, `MMI_InputList`
Example  The example uses the MMI_CreateTextDialog routine to create and display a text dialog.

Define a help text containing the 'inverse written word "Help"

\[
\text{CONST Helptext} = \text{MMI\_INVERSE\_ON} + \\
\text{"Help"} + \text{MMI\_INVERSE\_OFF} + \\
\text{" Test"}
\]

\[
\text{MMI\_CreateTextDialog(5, "TEXT", "DIALOG\_CAPTION", Helptext)}
\]

6.1.12 MMI_CreateGraphDialog

Description  Create and show a graphics dialog.

Declaration  

\[
\text{MMI\_CreateGraphDialog(}
\text{BYVAL sCaptionLeft AS _Token,}
\text{BYVAL sCaptionRight AS _Token,}
\text{BYVAL sHelpText AS _Token )}
\]

Remarks  The routine creates and shows a graphics dialog filled with the left part of the title bar sCaptionLeft, the caption sCaptionRight and the help text sHelpText for later use of MMI graphics functions. The size of the field is the whole dialog display area = 232 x 48 pixels. Only one graphics dialog can exist at the same time. If CreateGraphDialog is called while already a graphics dialog exists, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

Note  Only a text dialog or a measurement dialog is valid at a time. They cannot be defined at the same time. A graphic dialog overrides a text or measurement dialog but does not delete the definition of it.
Parameters

sCaptionLeft in The maximal five-character long part of the title bar displayed left of the sCaptionRight, with a separation symbol
sCaptionRight in The caption of the dialog.
shHelpText in This text is shown, when the help button Shift-F1 is pressed and the help functionality of the theodolite is enabled.

Return-Codes

RC_OK Successful termination.

See Also

MMI_DeleteDialog, MMI_CreateTextDialog, GSI_CreateMDlg, MMI Graphic Functions

Example

The example uses the MMI_CreateGraphDialog routine to create and display a graphic dialog field.

MMI_CreateGraphDialog("GRAPH","DIALOG CAPTION","This is a help text")

6.1.13 MMI_DeleteDialog

Description

Deletes a dialog.

Declaration

MMI_DeleteDialog()

Remarks

The routine deletes the currently active dialog. It makes no distinction between graphic, measure and text dialog. By deleting the dialog all user defined buttons added with MMI_AddButton are deleted as well.

Return-Codes

RC_OK Successful termination.
BAS_NO_DLG_EXIST No dialog exists for this operation.

See Also

MMI_CreateTextDialog, MMI_CreateGraphDialog, GSI_CreateMDlg
Example

The example uses the MMI_DeleteDialog routine to delete a text, measure or graphic dialog.

```
MMI_DeleteDialog()
```

6.1.14 MMI_CheckButton

Description

Checks if a button was pressed.

Declaration

```
MMI_CheckButton( lKeyPressed AS Logical )
```

Remarks

The routine `MMI_CheckButton` checks the keyboard buffer for pressed buttons. If a button was pressed, the routine returns `KeyPressed = TRUE`, otherwise `KeyPressed = FALSE` is returned.

```
Note The routine MMI_CheckButton does not wait until a button was pressed. It only checks the keyboard buffer.
```

Parameters

- `lKeyPressed` In
  - `lKeyPressed = TRUE` is returned, if a valid button was pressed.
  - Otherwise the value of `lKeyPressed` is `FALSE`.

Return-Codes

- `RC_OK` Successful termination.
- `BAS_NO_DLG_EXIST` No dialog exists for this operation.

See Also

- `MMI_AddButton`
- `MMI_GetButton`
Example

The example uses the MMI_CheckButton routine to wait until a (valid) key was pressed.

```plaintext
DIM lKeyPressed AS Logical
DO
  MMI_CheckButton( lKeyPressed )
LOOP UNTIL lKeyPressed
' do something ..
```

6.1.15 MMI_GetButton

Description

Get the button identifier of the pressed button.

Declaration

```plaintext
MMI_GetButton( iButtonId AS Integer,
               BYVAL lAllKeys AS Logical )
```

Remarks

Waits until a valid key is pressed and returns the button identifier `iButtonId` of the pressed button.

If `lAllKeys = FALSE`, the keys ESC, ENTER, ON/OFF or any assigned button (added with `MMI_AddButton`) terminates this function and the `iButtonId` of the pressed button is returned. If `lAllKeys = TRUE`, additional keys i.e. the cursor keys terminates this routine too. For details see table below.

Note: This function relates to the currently active dialog.

Parameters

- `iButtonId` Out  The identifier of the pressed button. For values of `iButtonId` see the table below.
- `lAllKeys` In   Determines which keys exit the routine. If `lAllKeys = TRUE` any valid pressed key exit the routine, otherwise only normal ones.
### Button pressed

<table>
<thead>
<tr>
<th>Button pressed</th>
<th>iButtonId returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>1AllKeys = TRUE</td>
<td>iAllKeys = FALSE</td>
</tr>
<tr>
<td>assigned (using MMI_AddButton)</td>
<td>MMI_F1_KEY, MMI_F6_KEY,</td>
</tr>
<tr>
<td>&quot;F1&quot;..&quot;F6&quot;, &quot;SHIFT-F2&quot;..</td>
<td>MMI_SHF2_KEY, MMI_SHF6_KEY</td>
</tr>
<tr>
<td>&quot;SHIFT-F6&quot;</td>
<td></td>
</tr>
<tr>
<td>unassigned &quot;F1&quot;..&quot;F6&quot;,</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>&quot;SHIFT-F2&quot;.. &quot;SHIFT-F6&quot;</td>
<td></td>
</tr>
<tr>
<td>assigned &quot;CODE&quot;</td>
<td>MMI_CODE_KEY, MMI_CODE_KEY</td>
</tr>
<tr>
<td>unassigned &quot;CODE&quot;</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>&quot;ENTER&quot; within dialog, focus</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>on a field</td>
<td></td>
</tr>
<tr>
<td>&quot;ENTER&quot; within dialog, no</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>focus</td>
<td></td>
</tr>
<tr>
<td>&quot;ENTER&quot; after editing</td>
<td>MMI_EDIT_ENTER_KEY, MMI_EDIT_ENTER_KEY</td>
</tr>
<tr>
<td>&quot;ESC&quot; within dialog</td>
<td>MMI_ESC_KEY, MMI_ESC_KEY</td>
</tr>
<tr>
<td>&quot;ESC&quot; after editing</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>&quot;SHIFT&quot;, focus on spin/list-</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>field</td>
<td></td>
</tr>
<tr>
<td>&quot;0&quot;..&quot;9&quot;, focus</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>on spin/list-field</td>
<td></td>
</tr>
<tr>
<td>&quot;0&quot;..&quot;9&quot;, no focus</td>
<td>MMI_NUM0_KEY, MMI_NUM9_KEY</td>
</tr>
<tr>
<td>&quot;CE&quot;</td>
<td>MMI_UNASS_KEY, no return</td>
</tr>
<tr>
<td>cursor keys</td>
<td>MMI_UP_KEY, MMI_DOWN_KEY,</td>
</tr>
<tr>
<td></td>
<td>MMI_RIGHT_KEY, MMI_LEFT_KEY</td>
</tr>
</tbody>
</table>

### Return-Codes

- **RC_OK**  Successful termination.
- **BAS_NO_DLG_EXIST**  No dialog exists for this operation.

### See Also

MMI_AddButton, MMI_CheckButton
Example
The example uses the `MMI_GetButton` routine to react to a pressed button. To make a function key valid for `MMI_GetButton` it must be added to the dialog (with `MMI_AddButton`).

```plaintext
DIM iActionButton AS Integer
DIM iPressedButton AS Integer

iActionButton = MMI_F2_KEY
MMI_GetButton iPressedButton, TRUE
IF iPressedButton = iActionButton THEN
    'any actions
END IF
```

6.1.16 `MMI_AddButton`

**Description**
Add a button to a dialog.

**Declaration**

```plaintext
MMI_AddButton( BYVAL iButtonId AS Integer,  
   BYVAL sCaption AS _Token )
```

**Remarks**
The routine `MMI_AddButton` adds the button with the Identifier `iButtonId` to the actual dialog and places the text `sCaption` onto the button. These added buttons are valid for the routines `MMI_CheckButton` and `MMI_GetButton` and the input routines (`MMI_InputStr`, `MMI_InputVal`, `MMI_InputInt` and `MMI_InputList`) which means the according button identifier can be returned from these routines.

**Note**
Either a text dialog or a measurement dialog can be defined at a time. Additionally a graphics dialog can override one of these above. Then the functionality applies to the graphics dialog.

The added buttons can be deleted with the routine `MMI_DeleteButton` while the dialog exists. Closing the dialog with `MMI_DeleteDialog` deletes all buttons attached to this dialog.
Parameters

**iButtonId in**  Identifier of the button to be added. See for the values that can be used for the iButtonId under the routine description **MMI_GetButton**. Only **MMI_F1_KEY..MMI_F5_KEY**, **MMI_SHF2_KEY..MMI_SHF6_KEY** and **MMI_CODE_KEY** are available for the **AddButton** routine.

**sCaption in**  The text placed onto the button, left alignment (max. 5 characters).

Return-Codes

- **RC_OK**  Successful termination.
- **BAS_NO_DLG_EXIST**  No dialog exists for this operation.
- **MMI_BUTTON_ID_EXISTS**  This button has been defined already.

See Also  **MMI_GetButton**, **MMI_CheckButton**, **MMI_DeleteButton**

Example  The example uses the **MMI_AddButton** routine to add the **F2-KEY** with the caption "EXIT" to the dialog.

```
MMI_AddButton( MMI_F2_KEY, "EXIT" )
```

6.1.17 **MMI_DeleteButton**

Description  Delete a button from a dialog.

Declaration  ```
MMI_DeleteButton( iButtonId AS Integer )
```

Remarks  The routine **MMI_DeleteButton** deletes the button with the Identifier **iButtonId** from the actual dialog. Only a button that was added with **MMI_AddButton** can be deleted. Closing the dialog with **MMI_DeleteDialog** deletes all buttons attached to this dialog.
Parameters

*iButtonId* in Identifier of the button to be deleted. See for the values that can be used for *iButtonId* under the routine description "MMI_GetButton".

Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.
- **MMI_ILLEGAL_BUTTON_ID** This button has not been defined by "MMI_AddButton".

See Also

- "MMI_AddButton"

Example

The example uses the "MMI_DeleteButton" routine to delete the F2-KEY from the dialog.

```
MMI_DeleteButton( MMI_F2_KEY )
```

6.1.18 MMI_PrintStr

Description

Print a string on a text dialog.

Declaration

```
MMI_PrintStr( BYVAL iColumn AS Integer,
              BYVAL iLine AS Integer,
              BYVAL sText AS String30,
              BYVAL lValid AS Logical )
```

Remarks

The text string *sText* is placed on position *iColumn* and *iLine* on the text dialog. If *lValid* is not TRUE, then the symbols for invalid values are displayed. Too long text strings are truncated, illegal co-ordinates are adjusted.

**Note**

A text dialog must already exist. Only display length number of character will be displayed, hence 29.

Parameters

*iColumn* in The horizontal position (0..28)
**6. System Functions**

### iLine in
The vertical position (0..number of lines defined with MMI_CreateTextDialog)

### sText in
The text string to display

### lValid in
Determines if the value should be shown as valid. If lValid = TRUE the value sText is displayed, otherwise the symbols for invalid values are displayed.

### Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

### See Also

MMI_InputStr

### Example

The example uses the MMI_PrintStr routine to print the text string „Hello World“ in the first line on row 2 of the actual text dialog.

```plaintext
MMI_PrintStr( 2, 0, "Hello World", TRUE )
```

### 6.1.19 MMI_PrintTok

**Description**
Print a string on a text dialog.

**Declaration**

```plaintext
MMI_PrintTok( BYVAL iColumn AS Integer, BYVAL iLine AS Integer, BYVAL sText AS _Token )
```

**Remarks**

The text token sText is placed on position iColumn and iLine on the text dialog. Too long text strings are truncated, illegal co-ordinates are adjusted. This routine may be used instead of MMI_PrintStr to support internationalisation of multiple language applications.

**Note**
A text dialog must already exist.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn</td>
<td>The horizontal position (0..28)</td>
</tr>
<tr>
<td>iLine</td>
<td>The vertical position (0..number of lines defined with MMI_CreateTextDialog)</td>
</tr>
</tbody>
</table>
The text string to display

**Return-Codes**

- **RC_OK**: Successful termination.
- **BAS_NO_DLG_EXIST**: No dialog exists for this operation.
- **TXT_UNDEF_TOKEN**: The given token could not be found in the database. Most probably an old version is loaded either on TPS or simulator.
- **RC_IVPARAM**: No text token database is loaded with the currently set language.

**See Also**

- `MMI_PrintfStr`

**Example**

The example uses the `MMI_PrintfTok` routine to print the text string "Hello World" in the first line on row 2 of the actual text dialog.

```plaintext
MMI_PrintfTok( 2, 0, "Hello World" )
```

---

**6.1.20 MMI_PrintfVal**

**Description**

Print a value on a text dialog.

**Declaration**

```plaintext
MMI_PrintfVal( BYVAL iColumn AS Integer, 
        BYVAL iLine AS Integer, 
        BYVAL iLen AS Integer, 
        BYVAL iDecimals AS Integer, 
        BYVAL dVal AS Double, 
        BYVAL lValid AS Logical, 
        BYVAL iMode AS Integer )
```

**Remarks**

This routine can be used to display double values (or values with equal type, e.g. dimension). If `lValid = TRUE` the value `dVal` is placed on position `iColumn` and `iLine` on the text dialog, otherwise the symbols for invalid values "-----" are displayed. Too long value strings are truncated, illegal co-ordinates are adjusted. If `iMode = MMI_DIM_ON`, a dimension field is automatically displayed when the type of `dVal` has units.
If the dVal cannot be displayed in iLen characters, then "xxx" will be displayed instead.

**Note** A text dialog must already exist.

## Parameters

- **iColumn** in  The horizontal position (0..28).
- **iLine** in  The vertical position (0..number of lines defined with CreateTextDialog).
- **iLen** in  The length of the value consisting of a sign, the characters before and after the comma and the comma itself. The dimension field is not included.
- **iDecimals** in  The number of decimals. If iDecimals = -1 then the number of decimals set by the system is taken.
- **dVal** in  The value to display. Use this routine to display double (and equal to double) values with the correct units. For integer values a separate routine (MMI_PrintInt) exists.
- **lValid** in  Determines if the value should be shown as valid. If lValid = TRUE the value dVal is displayed, otherwise the symbols for invalid values are displayed.
- **iMode** in  Determines the display of the dimension. If Mode = MMI_DIM_ON a dimension field is automatically displayed when the type dVal has units. Otherwise use MMI_DEFAULT_MODE.

## Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

## See Also

- MMI_PrintInt, MMI_InputVal

## Example

The example uses the MMI_PrintVal routine to print the value of TestVal as distance (with corresponding dimension) in the first line on row 2 of the currently open text dialog.
DIM TestVal AS Distance
TestVal = 287.47

MMI_PrintVal( 2, 0, 10, 2, TestVal, TRUE,
       MMI_DIM_ON )

6.1.21 MMI_PrintInt

Description
Print an integer value on a text dialog.

Declaration
MMI_PrintInt( BYVAL iColumn AS Integer,
       BYVAL iLine AS Integer,
       BYVAL iLen AS Integer,
       BYVAL iVal AS Integer,
       BYVAL lValid AS Logical )

Remarks
This routine can be used to display integer values. Too long value
strings are truncated, illegal co-ordinates are adjusted. If
lValid = TRUE the value iVal is placed on position
iColumn and iLine on the text dialog, otherwise the symbols
for invalid values are displayed.
If the iVal can not be displayed in iLen characters, then "xxx"
will be displayed instead.

Note
A text dialog must already exist.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn</td>
<td>The horizontal position (0..28).</td>
</tr>
<tr>
<td>iLine</td>
<td>The vertical position (0..number of lines defined with MMI_CreateTextDialog).</td>
</tr>
<tr>
<td>iLen</td>
<td>The length of the value plus the sign.</td>
</tr>
<tr>
<td>iVal</td>
<td>The value to display. Use this routine to display integer values. For double values a separate routine (MMI_PrintVal) exists.</td>
</tr>
<tr>
<td>lValid</td>
<td>Determines if the value should be shown as valid. If lValid = TRUE the value iVal is displayed, otherwise the symbols for invalid values are displayed.</td>
</tr>
</tbody>
</table>
Return-Codes

RC_OK
Successful termination.

BAS_NO_DLG_EXIST
No dialog exists for this operation.

See Also
MMI_PrintVal
MMI_InputInt

Example
The example uses the MMI_PrintInt routine to print the value of TestVal in the first line on row 2 of the currently open text dialog.

DIM TestVal AS Integer
TestVal = 1000

MMI_PrintInt( 2, 0, 5, TestVal, TRUE )

6.1.22 MMI_InputStr

Description
Get a string input in a text dialog.

Declaration
MMI_InputStr( BYVAL iColumn AS Integer,
BYVAL iLine AS Integer,
BYVAL iLen AS Integer,
BYVAL iMode AS Integer,
sText AS String30,
lValid AS Logical,
iButtonId AS Integer )

Remarks
If lValid = TRUE the text string sText is placed on position iColumn and iLine on the text dialog, otherwise the symbols for invalid values are displayed.
Illegal co-ordinates are adjusted. If the length of the string exceeds the given length iLen the string is truncated at position iLen.
After the edit process the string is returned and the text is placed right aligned on the display. If the length iLen <= 0 or no part of the field is in the dialog area the Text is not edited and the routine exits.

The string can be edited by pressing αEDIT or a numerical key. If iMode = MMI_DEFAULT_MODE the keys ESC, ENTER,
ON/OFF or any user defined button (added with 
MMI_AddButton) terminates the edit process and the 
iButtonId of the pressed button is returned. If iMode = 
MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys 
terminates MMI_InputStr too. For details see 
MMI_GetButton.

Note  A text dialog must already exist.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn</td>
<td>The horizontal position (0..28).</td>
</tr>
</tbody>
</table>
| iLine       | The vertical position (0..number of lines defined with 
             | MMI_CreateTextDialog).                          |
| iLen        | The length of the input field.                   |
| iMode       | Defines the editing mode.                        |
|             | MMI_DEFAULT_MODE defines normal editing          |
|             | MMI_SPECIALKEYS_ON allows editing with full cursor control |
| sText       | The text string to edit.                        |
| lValid      | Determines if the value should be shown as valid. 
             | If lValid=TRUE the string sText is displayed, otherwise 
             | the symbols for invalid values are displayed. |
| iButtonId   | The identifier of the pressed valid button to exit the edit process. |

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

See Also  MMI_PrintStr
Example

The example uses the `MMI_InputStr` routine to get the text string `sInputString` in the first line on row 2 of the actual text dialog.

```plaintext
DIM sInputString AS String30
DIM iButton AS Integer
DIM lValid AS Logical

sInputString = "The input text"
lValid = TRUE
MMI_InputStr( 2, 0, 20, MMI_DEFAULT_MODE, 
              sInputString, lValid, iButton )
```

6.1.23 MMI_InputVal

Description

Get a numerical input for double values in a text dialog.

Declaration

```plaintext
MMI_InputVal( BYVAL iColumn AS Integer, 
              BYVAL iLine AS Integer, 
              BYVAL iLen AS Integer, 
              BYVAL iDecimals AS Integer, 
              BYVAL dMin AS Double, 
              BYVAL dMax AS Double, 
              BYVAL iMode AS Integer, 
              dVal AS Double, 
              lValid AS Logical, 
              iButtonId AS Integer )
```

Remarks

If `lValid = TRUE` then the value `dVal` is placed on position `iColumn` and `iLine` on the text dialog, otherwise the symbols for invalid values are displayed. Illegal co-ordinates are adjusted. If `iMode = MMI_DIM_ON`, a dimension field is automatically displayed when the type of `dVal` has units. If the length `iLen <= 0` or no part of the field is in the dialog area the value is not edited and the routine exits.

The value within the bounds `dMin` and `dMax` can be edited by pressing `EDIT` or the numerical block keys. If `iMode = MMI_DEFAULT_MODE` the keys `ESC, ENTER, ON/OFF` or any user defined button (added with `MMI_AddButton`) terminates...
the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputVal too. For details see MMI_GetButton.

| Note | A text dialog must already exist. |

**Parameters**

- **iColumn in**: The horizontal position (0..28).
- **iLine in**: The vertical position (0..number of lines defined with MMI_CreateTextDialog).
- **iLen in**: The length of the value inclusive decimals, sign and the comma, exclusive the dimension field.
- **iDecimals in**: The number of decimals. If iDecimals = -1 the number of decimals set by the system is taken.
- **dMin in**: The lower and upper bounds.
- **dMax**: Defines the editing mode.
  - **MMI_DEFAULT_MODE**: defines normal editing.
  - **MMI_SPECIALKEYS_ON**: allows editing with full cursor control.
  - **MMI_DIM_ON**: shows a dimension field if dVal has units.
  - Modes can be added, i.e.
    - **MMI_SPECIALKEYS_ON + MMI_DIM_ON**
- **dVal inout**: The value to edit. Use this routine to edit double (and equal to double) values. For integer values a separate routine (MMI_InputInt) exists.
lValid  inout  Determines if the value should be shown as valid. If lValid=TRUE the value dVal is displayed, otherwise the symbols for invalid values are displayed.

iButtonId  out  The identifier of the pressed valid button to exit the edit process.

Return-Codes

RC_OK  Successful termination.
BAS_NO_DLG_EXIST  No dialog exists for this operation.

See Also

MMI_InputInt
MMI_PrintVal

Example  See example file "cursor.gbs" too.

The example uses the MMI_InputVal routine to get the distance of TestVal with default decimal places. Input field is placed in the second line on row 2 of the actual text dialog. The entered values must lie in the range 0..1000.

CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM TestVal AS Distance
DIM iButton AS Integer
DIM lValid AS Logical

lValid = FALSE

MMI_InputVal( 2, 1, 8, -1, 0, 1000, MODE,
TestVal, lValid, iButton )
6.1.24 MMI_InputInt

**Description**  
Get an integer input value in a text dialog.

**Declaration**  
```
MMI_InputInt( BYVAL iColumn AS Integer,
BYVAL iLine AS Integer,
BYVAL iLen AS Integer,
BYVAL iMin AS Integer,
BYVAL iMax AS Integer,
BYVAL iMode AS Integer,
iVal AS Integer,
iVarid AS Logical,
iButtonId AS Integer )
```

**Remarks**  
If `iVarid = TRUE` then the integer value `iVal` is placed on position `iColumn` and `iLine` on the text dialog. Illegal coordinates are adjusted. If the length `iLen` ≤ 0 or no part of the field is in the dialog area the value is not edited and the routine exits.

The integer value within the bounds `iMin` and `iMax` can be edited by pressing `EDIT` or the numerical block keys. If `iMode = MMI_DEFAULT_MODE` the keys `ESC`, `ENTER`, `ON/OFF` or any user defined button (added with `MMI_AddButton`) terminates the edit process and the `iVarid` of the pressed button is returned. If `iMode = MMI_SPECIALKEYS_ON` additional keys i.e. the cursor keys terminates `MMI_InputInt` too.

**Note**  
A text dialog must already exist.

**Parameters**

- **iColumn**  
The horizontal position (0..28).
- **iLine**  
The vertical position (0..number of lines defined with `MMI_CreateTextDialog`).
- **iLen**  
The length of the value plus the sign.
- **iMin**  
The lower and upper bounds.
- **iMax**
iMode in
Defines the editing mode.

MMI_DEFAULT_MODE defines normal editing
MMI_SPECIALKEYS_ON allows editing with full cursor control

iVal inout
The value to display. Use this routine to edit integer values. For double values a separate routine (MMI_InputVal) exists.

lValid inout
Determines if the value should be shown as valid. If lValid=TRUE the value iVal is displayed, otherwise the symbols for invalid values are displayed.

iButtonId out
The identifier of the pressed valid button to exit the edit process.

Return-Codes
RC_OK Successful termination.
BAS_NO_DLG_EXIST No dialog exists for this operation.

See Also
MMI_PrintInt, MMI_InputVal

Example
See example file "cursor.gbs" too.
The example uses the MMI_InputInt routine to get the value of iTestVal in the second line on row 2 of the actual text dialog. The entered values must lie in the range 0..1000.

CONST MODE = MMI_DEFAULT_MODE 'define editmode
DIM iTestVal AS Integer
DIM iButton AS Integer
DIM lValid AS Logical

lValid = FALSE
MMI_InputInt( 2,1,5,0,1000,
  MODE,iTestVal,lValid,iButton )
6.1.25 MMI_InputList

Description  Shows a list field in a text dialog.

Declaration  
```
MMI_InputList( BYVAL iColumn AS Integer,
          BYVAL iLine AS Integer,
          BYVAL iLen AS Integer,
          BYVAL iElements AS Integer,
          BYVAL iMode AS Integer,
          List AS ListArray,
          iIndex AS Integer,
          lValid AS Logical,
          iButtonId AS Integer )
```

Remarks  If lValid = TRUE then a list field is placed on position 
iColumn and iLine on the text dialog. Too long list elements 
are truncated, illegal co-ordinates are adjusted. The ListArray 
is an array of String30 with LIST_ARRAY_MAX_ELEMENT 
Elements. Only the first iElements are displayed. The value of 
iIndex defines which element is shown first. 
The list can be edited by pressing F6 (LIST). With the cursor 
keys UP and DOWN a field element can be selected. If the list 
elements are numbered (begins with a number), then the elements 
can be selected directly by pressing numerical buttons. If iMode 
= MMI_DEFAULT_MODE the keys ESC, ENTER, ON/OFF or any 
user defined button (added with MMI_AddButton) terminates 
the edit process and the iButtonId of the pressed button is 
returned. If iMode = MMI_SPECIALKEYS_ON additional keys 
i.e. the cursor keys terminates MMI_InputList too.

Note  A text dialog must already exist.

Parameters  
```
iColumn   in   The horizontal position (0..28).
iLine     in   The vertical position (0..number of 
              lines defined with 
              MMI_CreateTextDialog).
iLen      in   The displayed length of the list 
              elements.
```
iElements in The number of list elements. The maximum number is limited to
LIST_ARRAY_MAX_ELEMENT.

iMode in Defines the editing mode.
MMI_DEFAULT_MODE defines normal editing
MMI_SPECIALKEYS_ON allows editing with full cursor control

List in The array of the list elements.

iIndex inout Index (number of the line) of the first shown and selected field respectively.
Possible value for iIndex are in the range of 1 up to Elements.

lValid inout Determines if the value should be shown as valid. If lValid=TRUE the a value is displayed, otherwise the symbols for invalid values are displayed.

iButtonId out The identifier of the pressed valid button to exit the list process.

Return-Codes
RC_OK Successful termination.
BAS_NO_DLG_EXIST No dialog exists for this operation.

Example See example file "cursor.gbs" too.

The example uses the MMI_InputList routine to get the value of the selected list element (the selected line) of a list field displayed in the second line on row 2 of the actual text dialog. The first displayed line is the line with the number Index.
CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM iLen AS Integer
DIM iElements AS Integer
DIM List AS ListArray
DIM iIndex AS Integer
DIM iButton AS Integer
DIM lValid AS Logical

'initialize the variables
iLen = 10 'displayed length of the list
iElements = 7 'number of available fields
iIndex = 3 'number of the first shown list element
lValid = TRUE

List(1) = "1 Line No.: 1"
List(2) = "2 Line No.: 2"
List(3) = "3 Line No.: 3"
List(4) = "4 Line No.: 4"
List(5) = "5 Line No.: 5"
List(6) = "6 Line No.: 6"
List(7) = "7 Line No.: 7"

InputList( 5, 1, iLen, iElements, MODE,
            List, iIndex, lValid, iButton)

6.1.26 MMI_FormatVal

**Description**
Convert a value to a string and use TPS system formatting rules.

**Declaration**

```
MMI_FormatVal( BYVAL iType AS Integer,
                BYVAL iLen AS Integer,
                BYVAL iDecimals AS Integer,
                BYVAL dVal AS Double,
                BYVAL lValid AS Logical,
                BYVAL iMode AS Integer,
                sValStr AS String30 )
```

**Remarks**
If lValid = TRUE then this routine converts a double value (or values with equal type, e.g. dimension) to a text string, otherwise the symbols for invalid values are returned. The returned string
sValStr contains the value string in the same kind as it would be displayed on the Theodolite: the value is placed right aligned with the number iDecimals of decimals. If iMode = MMI_DIM_ON, a dimension field is appended to the output string when the type iTYPE allows it.

If the dVal can not be displayed in iLen characters, then "xxx" will be returned instead.

This routine is useful, if numeric values should be written on files (see chapter file handling for further information).

**Parameters**

**iType** in

The type of the numerical field. The type defines if a dimension field is available. Following values for the type can be used:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_FFORMAT_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>MMI_FFORMAT_DISTANCE</td>
<td>distance</td>
</tr>
<tr>
<td>MMI_FFORMAT_SUBDISTANCE</td>
<td>sub-distance [mm]</td>
</tr>
<tr>
<td>MMI_FFORMAT_ANGLE</td>
<td>angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_VANGLE</td>
<td>vertical angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_HZANGLE</td>
<td>horizontal angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_TEMPERATURE</td>
<td>temperature</td>
</tr>
<tr>
<td>MMI_FFORMAT_TIME</td>
<td>time 12h/24h-format</td>
</tr>
<tr>
<td>MMI_FFORMAT_DATE</td>
<td>date</td>
</tr>
<tr>
<td>MMI_FFORMAT_DATE_TIME</td>
<td>date/time</td>
</tr>
</tbody>
</table>

**iLen** in

The length of the value consisting of a sign, the characters before and after the comma and the comma itself. The dimension field is not included.

**iDecimals** in

The number of decimals. If iDecimals = -1 the number of decimals set by the system is taken.
dVal in  The value to convert. Use this routine to convert double (and equal to double) values.

iMode in  If iMode = MMI_DIM_ON a dimension string is automatically added to sValStr when the type dVal has units. Otherwise use MMI_DEFAULT_MODE.

sValStr out  sValStr contains the string representation of the value dVal.

Return-Codes

RC_OK  Successful termination.
RC_IVRESULT  The result is not valid due to an illegal input value.

See Also  sFormatVal

Example  The example uses the MMI_FormatVal routine to convert the value dTestVal as distance (with corresponding dimension).

```
DIM dTestVal AS Distance
DIM sVString AS String30

dTestVal = 287.47

MMI_FormatVal( MMI_FFORMAT_DISTANCE, 10, -1, dTestVal, TRUE, MMI_DIM_ON, sVString )
```

6.1.27 MMI_WriteMsg

Description  Output to a message window.

Declaration  MMI_WriteMsg( BYVAL sText AS _Token, BYVAL sCaption AS _Token, BYVAL iMsgType AS Integer, BYVAL iRetKey AS Integer )

Remarks  The function opens a message window on the display, which shows the text specified by sText. Lines that are too long to fit into the window are split automatically.
sText may contain a carriage return (character code 10) which breaks a line explicitly. The predefined constants MMI_INVERSE_ON and MMI_INVERSE_OFF can be used for inverse text.

Text lines, that exceed the size of the window, are not displayed. A title text, which will be printed on the first line of the message box, can be set with sCaption, which may not be longer than one line and contain neither font attributes nor type information.

Parameters

- **sText** in
  Text-token to be displayed on the window (on the Theodolite).

- **sCaption** in
  Text-token that will be displayed as title of the window.

- **iMsgType** in
  Defines the type of the message window to be displayed, with the corresponding text on the buttons; possible types:
  
  - MMI_MB_OK
  - MMI_MB_ABORT
  - MMI_MB_OK_ABORT
  - MMI_MB_ABORT_RETRY_CONT
  - MMI_MB_YES_NO_ABORT
  - MMI_MB_YES_NO
  - MMI_MB_RETRY_ABORT
  - MMI_MB_ABORT_CONT
  - MMI_MB_ABORT_RETRY_IGNORE
  - MMI_MB_ABORT_IGNORE

- **iRetKey** out
  Returns the button pressed, i.e.
  
  - MMI_MB_RET_OK
  - MMI_MB_RET_ABORT
  - MMI_MB_RET_RETRY
  - MMI_MB_RET_CONT
  - MMI_MB_RET_YES
  - MMI_MB_RET_NO
  - MMI_MB_RET_IGNORE
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

Example

The example uses the MMI_WriteMsg routine to display a message box with the title text "Warning" and the text "timed out" and shows the buttons "Retry", "Abort" returning the button-id in iRetKey.

```plaintext
MMI_WriteMsg( "Warning", "timeout", 
               MMI_MB_RETRY_ABORT, iMBRetKey )
```

6.1.28 MMI_WriteMsgStr

Description
Output to a message window.

Declaration

```plaintext
MMI_WriteMsgStr( BYVAL sText AS String255, 
                 BYVAL sCaption AS _Token, 
                 BYVAL iMsgType AS Integer, 
                 iRetKey AS Integer )
```

Remarks
The function opens a message window on the display, which shows the text specified by sText. Lines, which are too long to fit into the window, are split automatically. sText may contain a carriage return (character code 10) which breaks a line explicitly. The predefined constants MMI_INVERSE_ON and MMI_INVERSE_OFF can be used for inverse text. Text lines, that exceed the size of the window, are not displayed. A title text, which will be printed on the first line of the message box, can be set with sCaption, which may not be longer than one line and contain neither font attributes nor type information.

Note
This routine is different to MMI_WriteMsg in such a way that sText may be computed. But, of course, sText will not be entered into the text token data base.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>Text string to be displayed in a message box.</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td>sCaption</td>
<td>string</td>
</tr>
<tr>
<td>iMsgType</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>iRetKey</td>
<td>integer</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Return-Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

**See Also**

MMI_WriteMsg
The example uses the MMI_WriteMsgStr routine to display a message box with the title text "Warning" and the text:

```
MessageStr
time out in 10 seconds
```

and shows the buttons "Retry", "Abort" returning the button-id in iRetKey.

```
CONST iTimeOut AS Integer = 10
DIM sMessage As String255
DIM iMBRetKey AS Integer

sMessage = "MessageStr\010time out in " + Str$(iTimeOut) + "seconds"
MMI_WriteMsgStr( "Warning", sMessage, 
    MMI_MB_RETRY_ABORT,iMBRetKey )
```

### 6.1.29 MMI_DrawLine

#### Description
Draw a line.

#### Declaration
```
MMI_DrawLine( BYVAL iX1 AS Integer,
    BYVAL iY1 AS Integer,
    BYVAL iX2 AS Integer,
    BYVAL iY2 AS Integer,
    BYVAL iPen AS Integer )
```

#### Remarks
The function draws a line within the graphic field using the line-style iPen.

```
[Note] A graphics dialog has to be set up before.
```

#### Parameters
- **iX1** in x-co-ordinate of the beginning of the line [pixel]
- **iY1** in y-co-ordinate of the beginning of the line [pixel]
- **iX2** in x-co-ordinate of the end of the line [pixel]
- **iY2** in y-co-ordinate of the end of the line [pixel]
iPen in  Line-style; possible values:
   MMI_PEN_WHITE
   MMI_PEN_BLACK
   MMI_PEN_DASHED

Return-Codes

RC_OK       Successful termination.
BAS_NO_DLG_EXIST   No graphics dialog exists for this operation.

See Also   MMI_CreateGraphDialog, MMI_DrawRect,
            MMI_DrawCircle, MMI_DrawText

Example  The example uses the MMI_DrawLine routine to draw a line
          with the specified attributes.
          
          MMI_DrawLine( 10, 10, 100, 50, MMI_PEN_BLACK )

6.1.30  MMI_DrawRect

Description  Draw a rectangle.

Declaration  
            MMI_DrawRect( BYVAL iX1 AS Integer,
                         BYVAL iY1 AS Integer,
                         BYVAL iX2 AS Integer,
                         BYVAL iY2 AS Integer,
                         BYVAL iBrush AS Integer,
                         BYVAL iPen AS Integer )

Remarks  This function draws a rectangle in the graphic field using the fill-
          style iBrush and the line-style iPen.

Note  A graphics dialog has to be set up before.
Parameters

- `iX1` in x-co-ordinate at the upper left-hand corner of the rectangle [pixel]
- `iY1` in y-co-ordinate at the upper left-hand corner of the rectangle [pixel]
- `iX2` in x-co-ordinate at the bottom right-hand corner of the rectangle [pixel]
- `iY2` in y-co-ordinate at the bottom right-hand corner of the rectangle [pixel]
- `iBrush` in Fill-style for the rectangle; possible values:
  - `MMI_BRUSH_WHITE`
  - `MMI_BRUSH_BLACK`
  - `MMI_NO_BRUSH`
- `iPen` in Line-style:
  - `MMI_PEN_WHITE`
  - `MMI_PEN_BLACK`
  - `MMI_PEN_DASHED`

Return-Codes

- `RC_OK` Successful termination.
- `BAS_NO_DLG_EXIST` No graphics dialog exists for this operation.

See Also

- `MMI_CreateGraphDialog`, `MMI_DrawLine`, `MMI_DrawCircle`, `MMI_DrawText`

Example

The example uses the `MMI_DrawRect` routine to draw a rectangle with the specified attributes.

```
MMI_DrawRect( 10, 10, 100, 50, MMI_NO_BRUSH, MMI_PEN_BLACK )
```
6.1.31 MMI_DrawCircle

**Description**  
Draw a circle / ellipse.

**Declaration**  
```plaintext
MMI_DrawCircle( BYVAL iX AS Integer,
                 BYVAL iY AS Integer,
                 BYVAL iRx AS Integer,
                 BYVAL iRy AS Integer,
                 BYVAL iBrush AS Integer,
                 BYVAL iPen AS Integer )
```

**Remarks**  
This function draws a circle in the graphic field, using the radius `iRx`, the fill-style `iBrush`, and the line-style `iPen`, as long as `iRx = iRy`. Otherwise, an ellipse is drawn, where `iRx` and `iRy` are the lengths of the perpendicular radii.

**Note**  
A graphics dialog has to be set up before.

**Parameters**

- `iX` in x-co-ordinate at the centre of the circle/ellipse [pixel]
- `iY` in y-co-ordinate at the centre of the circle/ellipse [pixel]
- `iRx` in Radius of the circle, horizontal radius [pixel]
- `iRy` in Radius of the circle, vertical radius [pixel]
- `iBrush` in Fill-style for the rectangle; possible values:
  - MMI_BRUSH_WHITE
  - MMI_BRUSH_BLACK
  - MMI_NO_BRUSH
- `iPen` in Line-style; possible values:
  - MMI_PEN_WHITE
  - MMI_PEN_BLACK
  - MMI_PEN_DASHED
Return-Codes

RC_OK       Successful termination.
BAS_NO_DLG_EXIST  No graphics dialog exists for this operation.

See Also  MMI_CreateGraphDialog, MMI_DrawLine, MMI_DrawRect, MMI_DrawText

Example  Draw a circle with a radius of 10.

MMI_DrawCircle( 80, 25, 10, 10, 
               MMI_BRUSH_BLACK, 
               MMI_PEN_BLACK )

6.1.32 MMI_DrawText

Description  Draw / delete text.

Declaration  MMI_DrawText( BYVAL iX AS Integer, 
                      BYVAL iY AS Integer, 
                      BYVAL sText AS String20, 
                      BYVAL iAttr AS Integer, 
                      BYVAL iPen AS Integer )

Remarks  This function either draws (iPen = MMI_PEN_BLACK) or deletes (iPen = MMI_PEN_WHITE) a text string in graphic field. The co-ordinates (iX, iY) correspond to the upper left-hand corner of the first character. The character size is 6 x 8 pixel.

Note  A graphics dialog has to be set up before.

Parameters  
  iX  in  x-co-ordinate at the upper left-hand corner of the first character [pixel]
  iY  in  y-co-ordinate at the upper left-hand corner of the first character [pixel]
  sText  in  Pointer to the text string
  iAttr  in  Text attribute
               MMI_TXT_NORMAL  normal text
               MMI_TXT_INVERSE  inverted text
iPen  in  MMI_PEN_BLACK   draw text
        MMI_PEN_WHITE   delete text

**Return-Codes**

- **RC_OK**  Successful termination.
- **BAS_NO_DLG_EXIST**  No graphics dialog exists for this operation.

**See Also**

- `MMI_CreateGraphDialog`, `MMI_DrawLine`, `MMI_DrawRect`, `MMI_DrawCircle`

**Example**

Print a text at position 10, 10.

```bas
DIM sOutput AS String20
sOutput = "distance"
MMI_DrawText( 10, 10, sOutput, MMI_TXT_NORMAL, MMI_PEN_BLACK )
```

### 6.1.33  `MMI_DrawBusyField`

**Description**

Shows or hides the Busy-Icon.

**Declaration**

```bas
MMI_DrawBusyField(
  BYVAL lVisible as Logical )
```

**Remarks**

This function controls the Busy-Icon (Hourglass).

**Parameters**

- `lVisible`  in  TRUE: Icon is visible

**Return-Codes**

- **RC_OK**  Successful termination.
Example

The example shows and hides the Busy-Icon

```
MMI_DrawBusyField(TRUE) ' show icon
' time consuming function....
MMI_DrawBusyField(FALSE) ' hide icon
```

### 6.1.34 MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong

<table>
<thead>
<tr>
<th>Description</th>
<th>Create an alert beep.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td><code>MMI_BeepAlarm()</code></td>
</tr>
<tr>
<td></td>
<td><code>MMI_BeepNormal()</code></td>
</tr>
<tr>
<td></td>
<td><code>MMI_BeepLong()</code></td>
</tr>
<tr>
<td>Remarks</td>
<td>The functions create one or a sequence of alert beeps with configurable volume, if the boxes are turned on. Any previously set continuous signal beep will be finished.</td>
</tr>
<tr>
<td>Return-Codes</td>
<td><code>RC_OK</code> Successful termination.</td>
</tr>
<tr>
<td>See Also</td>
<td>MMI_StartVarBeep</td>
</tr>
<tr>
<td></td>
<td>MMI_SwitchVarBeep</td>
</tr>
<tr>
<td></td>
<td>MMI_GetVarBeepStatus</td>
</tr>
<tr>
<td>Example</td>
<td>The example uses the <code>MMI_BeepNormal</code> to sound a signal beep.</td>
</tr>
</tbody>
</table>

```
MMI_BeepNormal()
```

### 6.1.35 MMI_StartVarBeep

<table>
<thead>
<tr>
<th>Description</th>
<th>Start beep sequences with configurable interrupts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td><code>MMI_StartVarBeep( BYVAL iRate AS Integer )</code></td>
</tr>
<tr>
<td>Remarks</td>
<td>The function creates sequences of beeps with configurable interrupts.</td>
</tr>
</tbody>
</table>

TPS1100-Version 1.30
If previously a continuous signal beep has been set, the new rate will be established.

### Parameters

- **iRate** in frequency in [%]; 0 is very slow, 100 is very fast

### Return-Codes

- **RC_OK**: Successful termination.

### See Also

- MMI_BeepAlarm,
- MMI_BeepNormal,
- MMI_BeepLong,
- MMI_SwitchVarBeep,
- MMI_GetVarBeepStatus

### Example

The example uses the `MMI_StartVarBeep` to create a very fast sequence of signal beeps.

```plaintext
MMI_StartVarBeep( 100 )
```

---

### 6.1.36 MMI_SwitchVarBeep

#### Description

Switch a varying beep.

#### Declaration

```plaintext
MMI_SwitchVarBeep( BYVAL lOn AS Logical )
```

#### Remarks

The function allows the general switching (on/off) of a signal beep. A continuous signal beep will be switched off immediately.

#### Parameters

- **lOn** in switches the beep on or off

  - **lOn** meaning
    - FALSE: the beep is switched off generally
    - TRUE: beep is on; the functions `MMI_BeepNormal` etc. will only work if the beep is switched on.

#### Return-Codes

- **RC_OK**: Successful termination.
6.1.37 **MMI_GetVarBeepStatus**

**Description**
Read the switch status for a variable signal beep.

**Declaration**
```
MMI_GetVarBeepStatus( lOn AS Logical )
```

**Remarks**
The function retrieves the state of the general signal beep switch.

**Parameters**
- `lOn` out state of the switch
  - `FALSE` off
  - `TRUE` on

**Return-Codes**
- `RC_OK` Successful termination.

**See Also**
- `MMI_BeepNormal`
- `MMI_BeepLong`
- `MMI_BeepAlarm`
- `MMI_StartVarBeep`
- `MMI_SwitchVarBeep`

---

**Example**
The example uses the `MMI_SwitchVarBeep` to switch off the beep.

```
MMI_SwitchVarBeep( TRUE )
```
Example

The example uses the `MMI_GetVarBeepStatus` to revert the beep status (i.e. switch on when it is off and vice versa).

```plaintext
DIM lOn AS Logical
MMI_GetVarBeepStatus(lOn)
MMI_SwitchVarBeep( NOT lOn )
```

### 6.1.38 MMI_SwitchAFKey

**Description**
Switch the aF... key on or off.

**Declaration**
`MMI_SwitchAFKEY( BYVAL lOn AS Logical )`

**Remarks**
The function allows the switching (on/off) off the aF... key. Normally it is enabled, but during tracking distances it is disabled.

**Parameters**
- `lOn` in switches the beep on or off
  - `lOn` meaning
    - `FALSE` Key is switched off generally
    - `TRUE` Key is active

**Return-Codes**
- `RC_OK` Successful termination.

**See Also**
- `BAP_MeasRec`
- `BAP_MeasDistAng`

**Example**
The example uses the `MMI_SwitchAFKey` to disable the aF... key.

```plaintext
MMI_SwitchAFKey( FALSE )
```
### 6.1.39 MMI_SwitchIconsBeep

**Description**
Switches measurement icons and special beeps on or off.

**Declaration**
```
MMI_SwitchIconsBeep( BYVAL lOn AS Logical )
```

**Remarks**
The function allows the switching (on/off) of the measurement icons and special beeps (sector and lost lock).

**Parameters**
- `lOn` in switches the icons and beep on or off
  - **lOn** meaning
    - FALSE: no measurement icons and no special beep
    - TRUE: the measurement icons will be updated and the beeps are enabled. This is the normal state during a measurement dialog with continuous measurements.

**Return-Codes**
- **RC_OK** Successful termination.

**See Also**
- BAP_MeasRec
- BAP_MeasDistAng

**Example**
The example uses the `MMI_SwitchIconsBeep` to disable the icons and beeps.

```
MMI_SwitchIconsBeep( FALSE )
```
6.1.40 MMI_SetAngleRelation

**Description**
Set the angle relationship.

**Declaration**
```
MMI_SetAngleRelation(
    BYVAL iVertRel AS Integer,
    BYVAL iHorzRel AS Integer)
```

**Remarks**
This function sets the relationship of the vertical and horizontal angles. Fields already displayed are not updated.

**Parameters**
- `iVertRel` in
  Relationship of the vertical angle; valid values:
  - MMI_VANGLE_IN_PERCENT
  - MMI_VANGLE_REL_HORIZON
  - MMI_VANGLE_REL_ZENIT
- `iHorzRel` in
  Relationship of the horizontal angle; valid values:
  - MMI_HANGLE_CLOCKWISE
  - MMI_HANGLE_ANTICLOCKWISE
  - MMI_HANGLE_CLOCKWISE_SOUTH
  - MMI_HANGLE_BEARING

**Return Codes**
- **RC_OK**
  Successful termination.
- **RC_IVPARAM**
  The function has been called with an invalid parameter

**See Also**
MMI_GetAngleRelation

**Example**
Set the angle relations (with internal default values).

```
MMI_SetAngleRelation(
    MMI_VANGLE_IN_PERCENT,
    MMI_HANGLE_CLOCKWISE)
```
6.1.41 MMI_GetAngleRelation

**Description**  
Request the current angle relationships.

**Declaration**  
```  
MMI_GetAngleRelation(iVertRel AS Integer,  
iHorzRel AS Integer)  
```  

**Remarks**  
This function returns the current vertical- and horizontal-angle relationships.

**Parameters**  
- `iVertRel` out  
  Relationship of the vertical angle  
- `iHorzRel` out  
  Relationship of the horizontal angle

**Return Codes**  
none

**See Also**  
MMI_SetAngleRelation

**Example**  
Get the angle relations.
```  
DIM iVertRel AS Integer  
DIM iHorzRel AS Integer  
MMI_GetAngleRelation( iVertRel, iHorzRel )  
```  

6.1.42 MMI_SetVAngleMode

**Description**  
Set the V-Angle mode.

**Declaration**  
```  
MMI_SetVAngleMode(BYVAL lAngleFree AS Logical)  
```  

**Remarks**  
This function sets the vertical angle mode. Normally (lAngleFree=FALSE), the vertical angle is fix if there is a valid distance available. If lAngleFree=TRUE, the vertical angle will be updated including all corresponding values (slope distance, vertical distance, coordinates etc)
6.1.43  MMI_GetVAngleMode

Description  Returns the V-Angle mode.

Declaration  MMI_GetVAngleMode(lAngleFree AS Logical)

Remarks  This function returns the vertical angle mode.

Parameters  
lAngleFree in  TRUE: V-Angle is free (running)

Return Codes  
RC_OK  Successful termination.

See Also  MMI_SetVAngleMode

Example  See example file „meas.gbs“.

6.1.44  MMI_SetAngleUnit

Description  Set the displayed unit of angle.

Declaration  MMI_SetAngleUnit(BYVAL iUnit AS Integer,  
            BYVAL iDigits AS Integer)

Remarks  This function sets the displayed unit of angle. Existing display  
fields are not updated. If iDigits is greater than the maximal  
number it will be reset to it without notifying the user. A negative  
value of iDigits is not allowed.

Note  The maximal number of decimal digits depends on the  
Theodolite class.
Parameters

iUnit in Specified unit of angle; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_ANGLE_GON</td>
<td>400 Gon</td>
</tr>
<tr>
<td>MMI_ANGLE_DEC</td>
<td>360 Decimal</td>
</tr>
<tr>
<td>MMI_ANGLE_SEXADEC</td>
<td>360 Sexadecimal</td>
</tr>
<tr>
<td>MMI_ANGLE_MIL</td>
<td>6400 Mil</td>
</tr>
<tr>
<td>MMI_ANGLE_PERCENT</td>
<td>-300 ≤ x ≤ 300; only for vertical angles</td>
</tr>
</tbody>
</table>

digits in Number of decimal places. The maximum number of decimal places (digits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_ANGLE_GON</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_DEC</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_SEXADEC</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_MIL</td>
<td>0-3</td>
</tr>
<tr>
<td>MMI_ANGLE_PERCENT</td>
<td>don’t care</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

See Also

MMI_GetAngleUnit

Example

Set the angle unit.

```plaintext
MMI_SetAngleUnit( MMI_ANGLE_GON, 3 )
```
6.1.45 MMI_GetAngleUnit

**Description**
Return the currently displayed unit of angle.

**Declaration**

```plaintext
MMI_GetAngleUnit(iUnit AS Integer, iDigits AS Integer)
```

**Remarks**
This function returns the current unit of angle.

**Parameters**

<table>
<thead>
<tr>
<th>iUnit</th>
<th>out</th>
<th>Specified unit of angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>iDigits</td>
<td>out</td>
<td>Number of decimal places.</td>
</tr>
</tbody>
</table>

**Return Codes**

| RC_OK | Successful termination. |

**See Also**
MMI_SetAngleUnit

**Example**
Get the angle unit.
```basic
DIM iUnit AS Integer
DIM iDigits AS Integer
MMI_GetAngleUnit( iUnit, iDigits )
```

6.1.46 MMI_SetDistUnit

**Description**
Set the displayed unit of distance.

**Declaration**

```plaintext
MMI_SetDistUnit(BYVAL iUnit AS Integer, BYVAL iDigits AS Integer)
```

**Remarks**
This function sets the display unit for distance. Fields already displayed are not updated. If `iDigits` is greater than the maximal number it will be reset to it without notifying the user. A negative value of `iDigits` is not allowed.

**Note**
The maximal number of decimal digits depends on the Theodolite class.
Parameters

iUnit in  Specified unit of distance; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_DIST_METER</td>
<td>Meter</td>
</tr>
<tr>
<td>MMI_DIST_FOOT</td>
<td>normal foot</td>
</tr>
<tr>
<td>MMI_DIST_FOOT_INCH</td>
<td>normal foot / inch / 1/8inch</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT</td>
<td>US-foot</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT_INCH</td>
<td>US-foot / inch / 1/8inch</td>
</tr>
<tr>
<td>MMI_DIST_MM</td>
<td>Millimetre</td>
</tr>
<tr>
<td>MMI_DIST_INCH</td>
<td>inches</td>
</tr>
</tbody>
</table>

iDigits in  Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_DIST_METER</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_DIST_FOOT</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_DIST_FOOT_INCH</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT_INCH</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_DIST_MM</td>
<td>0</td>
</tr>
<tr>
<td>MMI_DIST_INCH</td>
<td>0-3</td>
</tr>
</tbody>
</table>

Return Codes

RC_OK  Successful termination.

RC_IVPARAM  The function has been called with an invalid parameter

See Also  MMI_GetDistUnit

Example  Set the distance unit.

```
MMI_SetDistUnit( MMI_DIST_METER, 4 )
```
6.1.47 MMI_GetDistUnit

**Description**
Return the currently displayed unit of distance.

**Declaration**
```
MMI_GetDistUnit( iUnit AS Integer, iDigits AS Integer)
```

**Remarks**
This function returns the current unit of distance.

**Parameters**
- **iUnit out**
  Specified unit of distance
- **iDigits out**
  Number of decimal places.

**Return Codes**
- **RC_OK**
  Successful termination.

**See Also**
MMI_SetDistUnit

**Example**
Get the distance unit.
```
DIM iUnit AS Integer
DIM iDigits AS Integer

MMI_GetDistUnit( iUnit, iDigits )
```

6.1.48 MMI_SetPressUnit

**Description**
Set the displayed unit of pressure.

**Declaration**
```
MMI_SetPressUnit(BYVAL iUnit AS Integer, BYVAL iDigits AS Integer)
```

**Remarks**
This function sets the display unit for pressure. Fields already displayed are not updated. If `iDigits` is greater than 1 it will be reset to it without notifying the user. A negative value of `iDigits` is not allowed.
Parameters

iUnit in  Specified unit of pressure; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_PRESS_MBAR</td>
<td>MilliBar</td>
</tr>
<tr>
<td>MMI_PRESS_MMHG</td>
<td>Millimetre mercury</td>
</tr>
<tr>
<td>MMI_PRESS_INCHHG</td>
<td>Inch mercury</td>
</tr>
<tr>
<td>MMI_PRESS_HPA</td>
<td>Hekto-Pascal</td>
</tr>
<tr>
<td>MMI_PRESS_PSI</td>
<td>PSI</td>
</tr>
</tbody>
</table>

iDigits in  Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_PRESS_MBAR</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_MMHG</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_INCHHG</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_HPA</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_PSI</td>
<td>0-1</td>
</tr>
</tbody>
</table>

Return Codes

RC_OK  Successful termination.
RC_IVPARAM  The function has been called with an invalid parameter

See Also  MMI_GetPressUnit

Example  Set the pressure unit.

```
MMI_SetPressUnit(MMI_PRESS_MBAR, 1)
```
6.1.49  MMI_GetPressUnit

**Description**  Return the currently displayed unit of pressure.

**Declaration**  
```
MMI_GetPressUnit(iUnit AS Integer,
                 iDigits AS Integer)
```

**Remarks**  This function returns the current unit of pressure.

**Parameters**

- `iUnit` _out_  Specified unit of pressure
- `iDigits` _out_  Number of decimal places.

**Return Codes**

- **RC_OK**  Successful termination.

**See Also**  MMI_SetPressUnit

**Example**  Get the pressure unit.
```
DIM iUnit AS Integer
DIM iDigits AS Integer
MMI_GetPressUnit( iUnit, iDigits )
```

6.1.50  MMI_SetTempUnit

**Description**  Set the displayed unit of temperature.

**Declaration**  
```
MMI_SetTempUnit(BYVAL iUnit AS Integer,
                 BYVAL iDigits AS Integer)
```

**Remarks**  This function sets the display unit for temperature. Fields already displayed are not updated. If `iDigits` is greater than 1 it will be reset to it without notifying the user. A negative value of `iDigits` is not allowed.
Parameters

**iUnit** in Specified unit of temperature; possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TEMP_C</td>
<td>Celsius</td>
</tr>
<tr>
<td>MMI_TEMP_F</td>
<td>Fahrenheit</td>
</tr>
</tbody>
</table>

**iDigits** in Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>Angle unit</th>
<th>Places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TEMP_C</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_TEMP_F</td>
<td>0-1</td>
</tr>
</tbody>
</table>

Return Codes

**RC_OK** Successful termination.

**RC_IVPARAM** The function has been called with an invalid parameter.

See Also

MMI_GetTempUnit

Example

Set the temperature unit.

```
MMI_SetTempUnit( MMI_TEMP_C, 1 )
```
Return Codes

RC_OK  Successful termination.

See Also

MMI_SetTempUnit

Example

Get the temperature unit.

DIM iUnit AS Integer
DIM iDigits AS Integer

MMI_GetTempUnit( iUnit, iDigits )

6.1.52  MMI_SetDateFormat

Description  Set the date display format.

Declaration  MMI_SetDateFormat(BYVAL iFormat AS Integer)

Remarks  This function sets the format in which the date is to be displayed. Existing fields remain unchanged.

Parameters

iFormat  in  Specified date format; possible values:

value | meaning
--- | ---
MMI_DATE_EU | European: DD.MM.YY
MMI_DATE_US | US: MM/DD/YY
MMI_DATE_JP | Japanese: YY/MM/DD

Return Codes

RC_OK  Successful termination.
RC_IVPARAM  The function has been called with an invalid parameter

See Also

MMI_GetDateFormat
Example  Set the date format (internal default value).

\[ \text{MMI\_SetDateFormat( MMI\_DATE\_EU )} \]

### 6.1.53 MMI\_GetDateFormat

**Description**  Retrieves the date display format.

**Declaration**  

\[
\text{MMI\_GetDateFormat(iFormat AS Integer)}
\]

**Remarks**  This function retrieves the format used to display the date.

**Parameters**

- `iFormat`  out  Specified date format

**Return Codes**

- `RC_OK`  Successful termination.

**See Also**  

MMI\_SetDateFormat

**Example**  Get the date format.

```basic
DIM iFormat AS Integer
MMI\_GetDateFormat( iFormat )
```

### 6.1.54 MMI\_SetTimeFormat

**Description**  Set the time display format.

**Declaration**  

\[
\text{MMI\_SetTimeFormat(BYVAL iFormat AS Integer)}
\]

**Remarks**  This function sets the format in which the time is to be displayed. Existing fields remain unchanged.

**Parameters**

- `iFormat`  in  Specified time format; possible values:
  - `MMI\_TIME\_12H`  12 hour display
  - `MMI\_TIME\_24H`  24 hour display
Return Codes

RC_OK Successful termination.
RC_IVPARAM The function has been called with an invalid parameter.

See Also

MMI_GetTimeFormat

Example

Set the time format (internal default value).

MMI_SetTimeFormat( MMI_TIME_12H )

6.1.55 MMI_GetTimeFormat

Description
Retrieves the time display format.

Declaration

MMI_GetTimeFormat(iFormat AS Integer)

Remarks
This function retrieves the format used to display the time.

Parameters

iFormat out Specified time format

Return Codes

RC_OK Successful termination.
RC_IVPARAM The function has been called with an invalid parameter.

See Also

MMI_SetTimeFormat

Example

Get the time format.

DIM iFormat AS Integer

MMI_GetTimeFormat( iFormat )
6.1.56  MMI_SetCoordOrder

Description  Set the co-ordinate order.

Declaration  MMI_SetCoordOrder(BYVAL iOrder AS Integer)

Remarks  This function sets the order of co-ordinates. The fields already displayed are not changed.

Parameters

iOrder  in  Specifies the co-ordinate order; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_COORD_N_E</td>
<td>Order North East</td>
</tr>
<tr>
<td>MMI_COORD_E_N</td>
<td>Order East North</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

See Also  MMI_GetCoordOrder

Example  Set the co-ordinate order (internal default value).

MMI_SetCoordOrder( MMI_COORD_N_E )
6.1.57 MMI_GetCoordOrder

**Description**
Retrieve the co-ordinate order.

**Declaration**
MMI_GetCoordOrder(iOrder AS Integer)

**Remarks**
This function retrieves the order in which co-ordinates are displayed.

**Parameters**
- iOrder out: Specified co-ordinate order

**Return Codes**
- RC_OK: Successful termination.

**See Also**
MMI_SetCoordOrder

**Example**
Get the co-ordinate order.
DIM iOrder AS Integer
MMI_GetCoordOrder( iOrder )

6.1.58 MMI_SetLanguage

**Description**
Set the display language.

**Declaration**
MMI_SetLanguage(BYVAL iLanguageNr AS Integer)

**Remarks**
This function sets the current language. All displayed text are immediately shown in the new language.

**Parameters**
- iLanguageNr in: Specifies the language number; possible values:
  - Value | Meaning
  - MMI_REF_LANGUAGE | Reference language (English) = 1
  - MMI_MAX_LANGUAGE | Language numbers
Return Codes

RC_OK	Successful termination.
RC_IVPARAM	The function has been called with an invalid parameter.
TXT_UNDEF_LANG	The given language is not defined.

See Also

MMI_GetLanguage

Example

Set the language for the display (internal default value).

```plaintext
MMI_SetLanguage( MMI_REF_LANGUAGE )
```
### 6.1.60 MMI_GetLangName

**Description**
Gets the name to a language number.

**Declaration**

```plaintext
MMI_GetLangName(
    ByVal iLangNr AS Integer,
    sLangName AS String20)
```

**Remarks**
This routine delivers the name associated with the number `iLangNr`.

**Parameters**

- `iLangNr` in Language number
- `sLangName` out Language description

**Return Codes**

- **RC_OK** Successful termination.
- **RC_IVPARAM** `iLangNr` is invalid

**See Also**

- `MMI_SetLanguage`
- `MMI_GetLanguage`

**Example**

Get the name of a language.

```plaintext
DIM sLangName AS String20

MMI_GetLangName( 2, sLangName )
```
### 6.2 BASIC APPLICATIONS BAP

#### 6.2.1 Summarizing Lists of BAP Types and Procedures

#### 6.2.1.1 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SetAccessoriesDlg</td>
<td>Sets the used accessories</td>
</tr>
<tr>
<td>BAP_FineAdjust</td>
<td>Automatic target positioning</td>
</tr>
<tr>
<td>BAP_GetMeasPrg</td>
<td>Get the current distance measure program.</td>
</tr>
<tr>
<td>BAP_MeasDistAngle</td>
<td>Measures distance and angles.</td>
</tr>
<tr>
<td>BAP_MeasRec</td>
<td>Measures and record distance and angles.</td>
</tr>
<tr>
<td>BAP_PosTelescope</td>
<td>Positioning of the Telescope.</td>
</tr>
<tr>
<td>BAP_SearchPrism</td>
<td>Searches the prism.</td>
</tr>
<tr>
<td>BAP_SetHz</td>
<td>Sets the horizontal angle to 0 or another given value.</td>
</tr>
<tr>
<td>BAP_SetManDist</td>
<td>Set the distance manually.</td>
</tr>
<tr>
<td>BAP_SetMeasPrg</td>
<td>Set the distance measure program.</td>
</tr>
<tr>
<td>BAP_SetPpm</td>
<td>Sets the ppm for distance measurements.</td>
</tr>
<tr>
<td>BAP_SetPrism</td>
<td>Sets the current prism type and constant.</td>
</tr>
</tbody>
</table>
6.2.2 BAP_SetAccessoriesDlg

**Description**
Sets the used accessories.

**Declaration**
BAP_SetAccessoriesDlg()

**Remarks**
This function displays the accessories dialog.

**Parameters**
-

**Return-Codes**
- **RC_OK**
  Successful termination.

**Example**
The example displays the accessories dialog
BAP_SetAccessoriesDlg()

6.2.3 BAP_MeasDistAngle

**Description**
Measures distance and angles.

**Declaration**
BAP_MeasDistAngle( iDistMode AS Integer, 
dHz AS Angle, 
dV AS Angle, 
dDist AS Distance, 
BYVAL lDisplayOn AS Logical, 
BYVAL sCaptionLeft AS _Token )

**Remarks**
Measures distance and angles and updates the data pool after correct measurements. It controls the special beep (Sector or Lost Lock) and switches measurement icons and disables the aF... key during tracking.
### Parameters

<table>
<thead>
<tr>
<th>iDistMode</th>
<th>Distance measuring modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode as Input</strong></td>
<td><strong>Meaning</strong></td>
</tr>
<tr>
<td>BAP_NO_MEAS</td>
<td>No new measurement, get last one</td>
</tr>
<tr>
<td>BAP_NO_DIST</td>
<td>No distance measurement, get only angles</td>
</tr>
<tr>
<td>BAP_DEF_DIST</td>
<td>Measure distance and angles using default measurement program</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Measure distance and angles using the tracking measurement program</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Measure distance and angles using the fast tracking measurement program</td>
</tr>
<tr>
<td>BAP_STOP_TRK</td>
<td>Stop tracking, no measurement. No valid results returned.</td>
</tr>
<tr>
<td>BAP_CLEAR_DIST</td>
<td>Clear distance (Theodolite data-pool), no measurement. No valid results returned.</td>
</tr>
<tr>
<td>BAP_RED_TRK_DIST</td>
<td>Measure distance and angles using the tracking with red laser measurement program</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Mode returned</strong></th>
<th><strong>Meaning</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_DEF_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td><strong>All other modes</strong></td>
<td>Returns BAP_DEF_DIST.</td>
</tr>
</tbody>
</table>

\[
dHz, dV \quad \text{out}\quad \text{Angles [rad], depends on}\]
### GeoBASIC Reference Manual

#### 6. System Functions

**iDistMode**

Distance [m], depends on

**sCaptionLeft**

Left caption for the distance measurement display.

**lDisplayOn**

TRUE: shows the distance measurement display during distance measurement.

### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Measurement executed successfully</td>
</tr>
<tr>
<td>AUT_RC_ANGLE_ ERROR</td>
<td>Angle measurement error</td>
</tr>
<tr>
<td>AUT_RC_BAD_ ENVIRONMENT</td>
<td>Bad Environment conditions</td>
</tr>
<tr>
<td>AUT_RC_CALACC</td>
<td>ATR-calibration failed</td>
</tr>
<tr>
<td>AUT_RC_DETECTOR_ ERROR</td>
<td>Error in target acquisition</td>
</tr>
<tr>
<td>AUT_RC_DETENT_ ERROR</td>
<td>Positioning not possible due to mounted EDM</td>
</tr>
<tr>
<td>AUT_RC_DEV_ERROR</td>
<td>Deviation measurement error</td>
</tr>
<tr>
<td>AUT_RC_INCACC</td>
<td>Position not exactly reached</td>
</tr>
<tr>
<td>AUT_RC_MOTOR_ ERROR</td>
<td>Motorization error</td>
</tr>
<tr>
<td>AUT_RC_MULTIPLE_ TARGETS</td>
<td>Multiple targets detected</td>
</tr>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>No target detected</td>
</tr>
<tr>
<td>AUT_RC_TIMEOUT</td>
<td>Position not reached</td>
</tr>
<tr>
<td>BAP_CHANGE_ALL_ TO_DIST</td>
<td>No prism has been found during distance measurement with ATR, command changed from &quot;All&quot; to &quot;Dist&quot;</td>
</tr>
<tr>
<td>TMC_ACCURACY_ GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ ACCURACY_ GUARANTEE</td>
<td>Info, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>Warning, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC submodule already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>An error occurred during distance measurement.</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>Error, wrong setting of PPM or MM on EDM</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>TMC_SIGNAL_ERROR</td>
<td>Error, no signal on EDM (only in signal mode)</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Error, measurement aborted</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

**See Also**
BAP_MeasRec

**Example**
See example file „meas.gbs“.

The example uses the `BAP_MeasDistAngle` routine to measure a distance and angles.

```basic
DIM iDistMode AS Integer
DIM dHz AS Angle
DIM dV AS Angle
DIM dDist AS Distance

iDistMode = BAP_DEF_DIST
BAP_MeasDistAngle(iDistMode, dHz, dV, dDist, TRUE, "TEST")
```
6.2.4 BAP_MeasRec

**Description**
Measures distance and angles records.

**Declaration**
BAP_MeasRec(iDistMode AS Integer, BYVAL lDisplayOn AS Logical, BYVAL sCaptionLeft AS _Token)

**Remarks**
Measures distance and angles and updates the Theodolite data pool after correct measurements and records values according the predefined record mask. After recording, a running point number will be incremented.

It controls the special beep (Sector or Lost Lock), switches Measurement icons and disables aF... Key during tracking.

**Parameters**

<table>
<thead>
<tr>
<th>iDistMode</th>
<th>Distance measuring modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode as Input</strong></td>
<td><strong>Meaning</strong></td>
</tr>
<tr>
<td>BAP_NO_MEAS</td>
<td>No new measurement before recording</td>
</tr>
<tr>
<td>BAP_NO_DIST</td>
<td>No distance measurement before recording (only new angles)</td>
</tr>
<tr>
<td>BAP_DEF_DIST</td>
<td>Use default distance measurement program and record values</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Use the tracking measurement program and record values</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Use the fast tracking measurement program and record values</td>
</tr>
<tr>
<td>BAP_STOP_TRK</td>
<td>Stop tracking, no measurement and no recording</td>
</tr>
<tr>
<td>BAP_CLEAR_DIST</td>
<td>Clear distance (Theodolite data pool), no measurement and no recording.</td>
</tr>
<tr>
<td>BAP_RED_TRK_DIST</td>
<td>Use the tracking with red laser measurement program and record values</td>
</tr>
<tr>
<td>Mode returned</td>
<td>Meaning</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BAP_DEF_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>All other modes</td>
<td>Returns BAP_DEF_DIST.</td>
</tr>
</tbody>
</table>

sCaptionLeft in TRUE: shows the distance measurement display during distance measurement.

Return Codes

- **RC_OK**
  - Successful termination.
- **WIR_NO_MEDIUM**
  - No storage medium is available.
- **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
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>No target detected</td>
</tr>
<tr>
<td>AUT_RC_TIMEOUT</td>
<td>Position not reached</td>
</tr>
<tr>
<td>BAP_CHANGE_ALL_TO_DIST</td>
<td>No prism has been found during distance measurement with ATR, command changed from &quot;All&quot; to &quot;Dist&quot;</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ACCURACY_GUARANTEE</td>
<td>Info, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>Warning, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC sub-module already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>An error occurred during distance measurement.</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>Error, wrong setting of PPM or MM on EDM</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>TMC_SIGNAL_ERROR</td>
<td>Error, no signal on EDM (only in signal mode)</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Error, measurement aborted</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

**See Also**  
BAP_MeasDistAngle, GSI_SetRecMask
Example

See example file „meas.gbs“.

The example uses the BAP_MeasMeasRec routine to record actual distance and angles (no new measurement).

```vba
DIM iDistMode AS Integer

iDistMode = BAP_NO_MEAS ' no measurement
BAP_MeasRec(iDistMode, FALSE, "")
```

6.2.5 BAP_FineAdjust

Description

Automatic target positioning.

Declaration

BAP_FineAdjust(  
  BYVAL dSearchHz AS Angle,  
  BYVAL dSearchV AS Angle )

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 dSearchV in V-direction and by parameter dSearchHz in Hz-direction. If no target is found, the instrument turns back to the initial start position. The ATR mode must be enabled for this functionality, see CSV_SetATRStatus and CSV_GetATRStatus.

Parameters

- dSearchHz in Search range Hz
- dSearchV in Search range V

Return Codes

- RC_OK: Successful termination.
- AUT_RC_TIMEOUT: Timeout while positioning of one or both axes. The position fault lies above 100[cc].
- AUT_RC_MOTOR_ERROR: Instrument has no ‘motorization’.
- RC_FATAL: Fatal error.
- RC_ABORT: Function aborted.
- AUT_RC_NO_TARGET: No target found.
6. System Functions

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  ATR error, at repeated occur call service.

See Also  CSV_SetATRStatus, CSV_GetATRStatus

Example  The example see sample TRACKING.GBS.

6.2.6 BAP_SearchPrism

Description  Searches the prism.

Declaration  BAP_SearchPrism(
            BYVAL lShowMessages As Logical )

Remarks  This procedure searches the prism. The searching area depends on the defined searching area and on the setting of the additional working area. This routine works only in ATR instruments and needs at least Firmware-Release 2.00.

Parameters  

            lShowMessages in  TRUE: show error-messages if there are problems to find the prism

Return Codes  

            RC_OK  Successful termination.
            AUT_RC_TIMEOUT  Timeout while positioning of one or both axes. The position fault lies above 100[cc].
6. System Functions

### BAP_SetManDist

**Description**
Set the distance manually.

**Declaration**
```
BAP_SetManDist(
    BYVAL sCaptionLeft AS _Token,
    BYVAL dDistance AS Double,
    iButtonId AS Integer )
```

**Remarks**
The BAP_SetManDist routine starts a dialog with the caption `sCaption` where the user can enter a horizontal distance. The distance will be stored into the Theodolite data pool.

- **TPS_Sim**
  - Has no effect. `iButtonId` will be set to `MMI_UNASS_KEY`.

**Parameters**
- `sCaptionLeft` in
  - Left caption string of the dialog
- `dDistance` in
  - Initial value for the distance. A negative value will be displayed as "----"
iButtonId  out  identifier of the pressed valid button to exit the dialog

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC sub-module already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

See Also

TMC_IFDistTapeMeasured, TMC_SetHandDist, TMC_GetPolar, TMC_GetCoordinate

Example

The example uses the BAP_SetManDist routine to enter a distance.

DIM iButton AS Integer
DIM dInitDist AS Distance

dInitDist = 15.0 'initial value

BAP_SetManDist( "BASIC", dInitDist, iButton )

6.2.8 BAP_SetPpm

Description  Sets the PPM for distance measurements.

Declaration  BAP_SetPpm()

Remarks  The BAP_SetPpm routine opens a dialog which the user can complete in order to calculate the PPM (parts per million) correction to be used to reduce the distance measured by the EDM.
GeoBASIC Reference Manual

6. System Functions

### TPS_Sim

**Has no effect.**

**Return Codes**

- **RC_OK**  
  Successful termination.
- **RC_SET_INCOMPL**  
  Parameter set-up for subsystem incomplete.

**See Also**

BAP_SetManDist, BAP_SetPrism

**Example**

The example uses the BAP_SetPpm routine to open the PPM dialog.

BAP_SetPpm()

---

### 6.2.9 BAP_SetPrism

**Description**

Sets the current prism type and constant.

**Declaration**

BAP_SetPrism()

**Remarks**

The BAP_SetPrism routine opens a dialog which the user can complete in order to choose one of five prism types/constants. Two types are LEICA defaults, whereas the other three can be named and the constant values given/changed by the user. The prism constants are always given and displayed in millimetres, regardless of the distance units in use at the time.

**Return Codes**

- **RC_OK**  
  Successful termination.

**See Also**

BAP_SetManDist, BAP_SetPpm

**Example**

The example uses the BAP_SetPrism routine to open the Prism dialog.
6.2.10 BAP_SetMeasPrg

**Description**
Set the distance measure program.

**Declaration**
```
BAP_SetMeasPrg( BYVAL iMeasPrg AS Integer )
```

**Remarks**
The `BAP_SetMeasPrg` routine sets the program for the distance measurement.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMeasPrg</td>
<td>Integer</td>
<td>Distance measure program</td>
</tr>
</tbody>
</table>

**Valid measure programs**

<table>
<thead>
<tr>
<th>Measure Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SINGLE_REF_</td>
<td>Single measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>STANDARD</td>
<td></td>
</tr>
<tr>
<td>BAP_SINGLE_REF_</td>
<td>Single measurement, with reflector, fast</td>
</tr>
<tr>
<td>FAST</td>
<td></td>
</tr>
<tr>
<td>BAP_SINGLE_REF_</td>
<td>Single measurement, with reflector and red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_SINGLE_RLESS</td>
<td>Single measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_CONT_REF_</td>
<td>Continuous measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>STANDARD</td>
<td></td>
</tr>
<tr>
<td>BAP_CONT_REF_</td>
<td>Continuous measurement, with reflector, fast</td>
</tr>
<tr>
<td>FAST</td>
<td></td>
</tr>
<tr>
<td>BAP_CONT_RLESS_</td>
<td>Continuous measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_AVG_REF_</td>
<td>Average measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>STANDARD</td>
<td></td>
</tr>
<tr>
<td>BAP_AVG_REF_</td>
<td>Average measurement, with reflector and red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_AVG_RLESS_</td>
<td>Average measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**
BAP_GetMeasPrg
Example

The example uses the BAP_SetMeasPrg routine to set the distance measurement program on single measurement without reflector.

\[
\text{BAP\_SetMeasPrg}(\text{BAP\_SINGLE\_RLESS\_VISIBLE})
\]

6.2.11 BAP_GetMeasPrg

Description
Get the current distance measure program.

Declaration

\[
\text{BAP\_GetMeasPrg}( \text{iMeasPrg AS Integer} )
\]

Remarks
The BAP_GetMeasPrg routine fetches the current program for the distance measurement.

Parameters

<table>
<thead>
<tr>
<th>iMeasPrg</th>
<th>out</th>
<th>Distance measure program</th>
</tr>
</thead>
</table>

Valid measure programs

<table>
<thead>
<tr>
<th>BAP_SINGLE_REF_STANDARD</th>
<th>Single measurement, with reflector, standard speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SINGLE_REF_FAST</td>
<td>Single measurement, with reflector, fast</td>
</tr>
<tr>
<td>BAP_SINGLE_REF_VISIBLE</td>
<td>Single measurement, with reflector and red laser</td>
</tr>
<tr>
<td>BAP_SINGLE_RLESS_VISIBLE</td>
<td>Single measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>BAP_CONT_REF_STANDARD</td>
<td>Continuous measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>BAP_CONT_REF_FAST</td>
<td>Continuous measurement, with reflector, fast</td>
</tr>
<tr>
<td>BAP_CONT_RLESS_VISIBLE</td>
<td>Continuous measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>BAP_AVG_REF_STANDARD</td>
<td>Average measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>BAP_AVG_REF_VISIBLE</td>
<td>Average measurement, with reflector and red laser</td>
</tr>
<tr>
<td>BAP_AVG_RLESS_VISIBLE</td>
<td>Average measurement, reflectorless, with red laser</td>
</tr>
</tbody>
</table>
See Also  BAP_SetMeasPrg

Example  The example uses the BAP_GetMeasPrg routine to fetch the current distance measurement program.
DIM iMeasPrg AS Integer
BAP_GetMeasPrg(iMeasPrg)

6.2.12  BAP_PosTelescope

Description  Positioning of the Telescope.

Declaration  BAP_PosTelescope(
    BYVAL eMode AS Integer,
    BYVAL eDspMode AS Integer,
    BYVAL dHz AS Double,
    BYVAL dV AS Double,
    BYVAL dHzTolerance AS Double,
    BYVAL dVTolerance AS Double)

Remarks  This procedure positions the telescope according to the specified mode and angles.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMode</td>
<td>Positioning mode.</td>
</tr>
<tr>
<td>BAP_POSIT</td>
<td>positioning on Hz and V angle</td>
</tr>
<tr>
<td>BAP_POSIT_HZ</td>
<td>positioning on Hz angle</td>
</tr>
<tr>
<td>BAP_POSIT_V</td>
<td>positioning on V angle</td>
</tr>
<tr>
<td>BAP_CHANGE_FACE</td>
<td>change face</td>
</tr>
</tbody>
</table>

| TPS_Sim | Has no effect. |
**eDspMode**
Controls the context and layout of the display during manual positioning. This parameter has no effect on motorised Theodolites.

- **BAP_POS_NOMSG** No message will be displayed
- **BAP_POS_MSG** Only a message will be displayed
- **BAP_POS_DLG** Positioning will be guided with a dialog if it is a non motorised Theodolite

**dHz, dV**
Target position

**dHzTolerance, dV_Tolerance**
In case of manual positioning, the tolerances define the upper and lower boundaries of the target position. For successful termination of the positioning, the final target position must be within these boundaries. If the tolerance is lower then the default accuracy of the Theodolite, the tolerance will be the default accuracy.

There is no effect on the motorised Theodolites. The tolerances (and speed) of the positioning will be defined separately.

**Return Codes**
- **RC_OK** Positioning successful
- **RC_ABORT** Abnormal termination (No positioning possible, ESC-Key)

**See Also**
- CSV_MakePositioning
- CSV_ChangeFace

**Example**
Position the telescope.

```basic
BAP_PosTelescope(BAP_CHANGE_FACE, BAP_POS_DLG, 0, 0, .5, .5)
```
6.2.13  BAP_SetHz

**Description**  
Sets the horizontal angle to 0 or another given value.

**Declaration**  
BAP_SetHz( BYVAL sCaptionLeft AS _Token )

**Remarks**  
This procedure offers a dialogue which the user can complete in order to influence the angular offset provided by the TMC subsystem for the horizontal angle encoder. A button is provided for setting the angle to zero, directly, or the user may prefer to input another given value. Furthermore, the angle beep (at the quarter circle positions from 0°) can be turned on and off.

**Note**  
If the instrument is in Lock mode, then the instrument tries to lock first before it sets the angle to 0.

**Parameters**

- **sCaptionLeft**  
  Left caption text for dialog

**See Also**

**Return Codes**

- **RC_OK**  
  Horizontal angular offset correct.

**Example**

Set the horizontal angle.

BAP_SetHz("BASIC" )
### 6.3 MEASUREMENT FUNCTIONS TMC

This section contains the lower level measurement procedures.

#### 6.3.1 Summarizing Lists of TMC Types and Procedures

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_Ang Switch_Type</td>
<td>Angle measurement switches</td>
</tr>
<tr>
<td>TMC_Angle_Type</td>
<td>Data structure for measuring angles.</td>
</tr>
<tr>
<td>TMC_Coordinate_Type</td>
<td>Data structure for the co-ordinates (tracking and fixed co-ordinates).</td>
</tr>
<tr>
<td>TMC_Dist Switches_Type</td>
<td>Distance measurement switches</td>
</tr>
<tr>
<td>TMC_Distance_Type</td>
<td>Data structure for the distance measurement.</td>
</tr>
<tr>
<td>TMC_HZ_V_Ang_Type</td>
<td>Horizontal and vertical angle.</td>
</tr>
<tr>
<td>TMC_Incline_Type</td>
<td>Data structure for the inclination measurement.</td>
</tr>
<tr>
<td>TMC_Offset Dist_Type</td>
<td>Target offset</td>
</tr>
<tr>
<td>TMC_PPM Corr Type</td>
<td>Corrections for distance measurement: PPM values</td>
</tr>
<tr>
<td>TMC_Geom Projection Type</td>
<td>Corrections for distance measurement: to define PPM values of projection</td>
</tr>
<tr>
<td>TMC_Geom Reduction Type</td>
<td>Corrections for distance measurement: to define PPM values of reduction to the reference</td>
</tr>
<tr>
<td>TMC_Atmos Temperature Type</td>
<td>Corrections for distance measurement: to define PPM values of atmosphere</td>
</tr>
<tr>
<td>TMC_REFRACTION_Type</td>
<td>Refraction correction for distance measurement</td>
</tr>
<tr>
<td>TMC_STATION_Type</td>
<td>Station co-ordinates</td>
</tr>
</tbody>
</table>
## 6.3.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_DoMeasure</td>
<td>Start a measure program.</td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the current face definition.</td>
</tr>
<tr>
<td>TMC_SetAngleFaceDef</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the refractive correction for measuring the distance.</td>
</tr>
<tr>
<td>TMC_SetRefractiveCorr</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the method of refractive correction for measuring the distance.</td>
</tr>
<tr>
<td>TMC_SetRefractiveMethod</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the PPM values for distance measurement corrections.</td>
</tr>
<tr>
<td>TMC_SetDistPpm</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the projection part of distance measurement corrections.</td>
</tr>
<tr>
<td>TMC_SetGeomProjection</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the reduction to the reference part of distance measurement corrections.</td>
</tr>
<tr>
<td>TMC_SetGeomReduction</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the atmosphere part of distance measurement corrections.</td>
</tr>
<tr>
<td>TMC_SetAtmCorr</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the current height of the reflector.</td>
</tr>
<tr>
<td>TMC_SetHeight</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the current horizontal offset.</td>
</tr>
<tr>
<td>TMC_SetHzOffset</td>
<td></td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets station co-ordinates.</td>
</tr>
<tr>
<td>TMC_SetStation</td>
<td></td>
</tr>
<tr>
<td>TMC_GetAngle</td>
<td>Measure angles.</td>
</tr>
<tr>
<td>TMC_GetAngle_Winc</td>
<td></td>
</tr>
<tr>
<td>TMC_getAngSwitch</td>
<td>Returns the angle measurement correction switches</td>
</tr>
<tr>
<td>TMC_GetCoordinate</td>
<td>Calculate and read co-ordinates.</td>
</tr>
<tr>
<td>TMC_GetDistSwitch</td>
<td>Returns the distance measurement correction switches</td>
</tr>
<tr>
<td>TMC_GetFace1</td>
<td>Get face information of current telescope position</td>
</tr>
<tr>
<td>TMC_GetInclineStatus</td>
<td>Returns the inclination compensator status.</td>
</tr>
<tr>
<td>TMC_GetInclineSwitch</td>
<td>Returns the compensator switch</td>
</tr>
<tr>
<td>TMC_GetOffsetDist</td>
<td>Returns the distance measurement offset</td>
</tr>
<tr>
<td>TMC_GetPolar</td>
<td>Calculate and read polar co-ordinates.</td>
</tr>
</tbody>
</table>
### TMC Data Structures

#### 6.3.2.1 TMC_INCLINE - Data structure for the inclination measurement

```plaintext
TYPE TMC_Incline_Type
    dCrossIncline AS Double cross inclination
    dLengthIncline AS Double alongside inclination
    dAccuracyIncline AS Double accuracy of measuring
    InclineTime AS Integer time of measuring
END TMC_Incline_Type
```

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_GetSimpleMea</td>
<td>Gets the results of distance and angle measurement</td>
</tr>
<tr>
<td>TMC_IFDistTapeMeasured</td>
<td>Gets information about manual measurement.</td>
</tr>
<tr>
<td>TMC_IFOffsetDistMeasured</td>
<td>Returns the EDM measurement mode</td>
</tr>
<tr>
<td>TMC_QuickDist</td>
<td>Measure slope distance and angles</td>
</tr>
<tr>
<td>TMC_SetAngSwitch</td>
<td>Defines the angle measurement correction switches</td>
</tr>
<tr>
<td>TMC_SetDistSwitch</td>
<td>Defines the distance measurement correction switches</td>
</tr>
<tr>
<td>TMC_SetHandDist</td>
<td>Sets distance manually.</td>
</tr>
<tr>
<td>TMC_SetInclineSwitch</td>
<td>Defines the compensator switch</td>
</tr>
<tr>
<td>TMC_SetOffsetDist</td>
<td>Defines the distance measurement offset</td>
</tr>
</tbody>
</table>
6.3.2.2 TMC_ANGLE - Data structure for measuring angles

TYPE TMC_Angle_Type
    dHz AS Double  horizontal angle
    dV AS Double  vertical angle
    dAngleAccuracy AS Double  accuracy of angle
    iAngleTime AS Integer  time of measurement
    Incline AS TMC_Incline_Type
        Incline_Type
    iFace AS Integer  information about position of the telescope
END TMC_Angle_Type

6.3.2.3 TMC_DISTANCE - Data structure for the distance measurement

TYPE TMC_Distance_Type
    Angle AS TMC_Angle_Type  set of angles belonging to distance
    dSlopeDist AS Double  slope distance
    dSlopeDistAccuracy AS Double  accuracy of distance
    dHorizDist AS Double  horizontal distance
    dHeightDiff AS Double  difference in altitude
    AngleCont AS TMC_Angle_Type  set of angles, measured continuously
    dSlopeDistCont AS Double  slope distance, measured continuously
    dHeightDiffCont AS Double  distance in altitude, measured continuously
END TMC_Distance_Type
6.3.2.4 TMC_COORDINATE - Data structure for the coordinates
(tracking and fixed co-ordinates)

```plaintext
TYPE TMC_Coordinate_Type
    dE AS Double   east co-ordinate
    dN AS Double   north co-ordinate
    dH AS Double   height co-ordinate
    iCoordTime AS Integer time of measurement
    dE_Cont AS Double east coordinate, measured continuously
    dN_Cont AS Double north co-ordinate, measured continuously
    dH_Cont AS Double height co-ordinate, measured continuously
    iCoordContTime AS Integer time of continuous measurement
END TMC_Coordinate_Type
```

6.3.2.5 TMC_HZ_V_Ang - Horizontal and vertical angle

```plaintext
TYPE TMC_HZ_V_Ang_Type
    dHz AS Double horizontal angle
    dV AS Double vertical angle
END TMC_HZ_V_Ang_Type
```

6.3.2.6 TMC_PPM_CORR - Corrections for distance measurement
(PPM values)

```plaintext
TYPE TMC_PPM_CORR_Type
    dPpmI AS Double individual ppm
    dPpmA AS Double atmospheric ppm
    dPpmR AS Double height relative ppm
    dPpmP AS Double projection contortion ppm
END TMC_PPM_CORR_Type
```

6.3.2.7 TMC_GEOM_PROJECTION - to define PPM values of projection

```plaintext
TYPE TMC_GEOM_PROJECTION_Type
    dProjectionSpace AS Double distance to the reference
    dProjectionScale AS Double factor of projection
    dEarthRadius AS Double earth radius
END TMC_GEOM_PROJECTION_Type
```
6.3.2.8 TMC_GEOM_REDUCTION - to define PPM values of reduction to the reference

TYPE TMC_GEOM_REDUCTION_Type
    dHeightReference AS Double reference height
    dEarthRadius AS Double earth radius
END TMC_GEOM_REDUCTION_Type

6.3.2.9 TMC_ATM_TEMPERATURE - to define PPM values of atmosphere

TYPE TMC_ATM_TEMPERATURE_Type
    dLambda AS Double laser wave length
    dPressure AS Double atmospheric pressure
    dDryTemperature AS Double dry temperature
    dWetTemperature AS Double wet temperature
END TMC_ATM_TEMPERATURE_Type

6.3.2.10 TMC_STATION - Station coordinates

TYPE TMC_STATION_Type
    dE0 AS Double easting co-ordinate
    dN0 AS Double northing co-ordinate
    dH0 AS Double height co-ordinate
    dHi AS Double instrument height
END TMC_STATION_Type

6.3.2.11 TMC_REFRACTION - Refraction correction for distance measurement

TYPE TMC_REFRACTION_Type
    bOnOff AS Logical TRUE if refraction is valid
    dEarthRadius AS Double earth radius
    dRefractiveScale AS Double refraction coefficient
END TMC_REFRACTION_Type

6.3.2.12 TMC_DIST_SWITCH_Type - Distance measurement switches

TYPE TMC_DIST_SWITCHES_Type
6.3.13 TMC_ANGLE_SWITCH_Type – Angle measurement switches

TYPE TMC_ANG_SWITCH_Type
  lInclineCorr AS Logical ' Inclination correction
  lStandAxisCorr AS Logical ' Standing axis correction
  lCollimationCorr AS Logical ' Collimation error correction
  lTiltAxisCorr AS Logical ' Tilting axis correction
END TMC_ANG_SWITCH_Type

6.3.14 TMC_OFFSET_DIST_Type – Target offset

TYPE TMC_OFFSET_DIST_Type
  dLengthVal AS Distance ' Target - Offset Length
  dCrossVal AS Distance ' Target - Offset Cross
  dHeightVal AS Distance ' Target - Offset Height
END TMC_OFFSET_DIST_Type

6.3.3 TMC_DoMeasure

Description Start a measure program.

Declaration TMC_DoMeasure( BYVAL iCommand AS Integer )

Remarks With this function a measure program is started. The commands start a distance measurement and / or a test mode. In addition an angle- and an inclination-measure are done (not at measurement).

The tracking measure program performs e.g. as follows: Start the measure program with TMC_DoMeasure (TMC_TRK_DIST).
The electronic distance measuring device (EDM) begins to run. Now the co-ordinates can be read, e.g. with TMC_GetCoordinate(). Tracking can be stopped with TMC_DoMeasure(TMC_STOP). With TMC_DoMeasure(TMC_CLEAR) the function will be stopped and the distance cleared.

<table>
<thead>
<tr>
<th>Note</th>
<th>After calling a measure program, the last valid distance results will be cleared (as after TMC_STOP).</th>
</tr>
</thead>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>iCommand</th>
<th>in</th>
<th>start a measure program; possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_STOP</td>
<td>switch off EDM and finish program</td>
<td></td>
</tr>
<tr>
<td>TMC_DEF_DIST</td>
<td>do default distance measure</td>
<td></td>
</tr>
<tr>
<td>TMC_TRK_DIST</td>
<td>do tracking distance measure</td>
<td></td>
</tr>
<tr>
<td>TMC_RTRK_DIST</td>
<td>do fast tracking distance measure</td>
<td></td>
</tr>
<tr>
<td>TMC_CLEAR</td>
<td>clear distance and switch off EDM</td>
<td></td>
</tr>
<tr>
<td>TMC_SIGNAL</td>
<td>start signal measurement (test mode)</td>
<td></td>
</tr>
<tr>
<td>TMC_RED_TRK_DIST</td>
<td>do tracking distance measure with red laser</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

TMC_GetPolar
TMC_GetCoordinate

**Return Codes**

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>measure program started</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Measurement system is busy</td>
</tr>
</tbody>
</table>
Example  
Start a distance measure, do something, stop it and clear results.

The following variable has to be defined:

```
TMC_DoMeasure (TMC_DEF_DIST) ' ... do a measure
TMC_DoMeasure (TMC_CLEAR)
```

6.3.4 TMC_GetPolar

Description  
Calculate and read polar co-ordinates.

Declaration  
```
TMC_GetPolar(
    BYVAL iWaitTime AS Integer,
    Polar AS TMC_Distance_Type,
    iReturnCode AS Integer )
```

Remarks  
The function corrects and takes in calculation a measured distance.
Angle and possibly inclination are being calculated. The result is a
point in polar co-ordinates.

Simple and multiple measures (distance tracking, altitude
tracking) are supported. The horizontal and the inclined distance
with the difference in altitude are read. The delay (iWaitTime)
just works on the distance measure, not on the measure of the
angle. As long as no new measure program is started, the results
can be read. Additional to the normal return codes
iReturnCode delivers also informational return codes which
will not interrupt program execution.

Note  
The measure program must have been started (see
TMC_DoMeasure).
Parameters

```
iWaitTime   in  delay time [ms] until a result is available
            =0  returns results with an already measured distance.
            >0  waits maximal the time iWaitTime for a result. If
                iWaitTime is chosen big enough (e. g. 60000, which is surely longer
                than the time-out period of the device), the system will wait for a result or until an
                error occurs
            <0  Performs an automatic target acquisition (if possible) and then tries to measuring in a until a
                valid result or an irrecoverable error occurs. The value itself of
                iWaitTime is ignored.
```

```
Polar       out  point in polar co-ordinates
iReturnCode out  see Additional Codes below
```

See Also

```
TMC_GetCoordinates
```

Additional Codes in iReturnCode

```
RC_OK
TMC_ACCURACY_GUARANTEE
TMC_NO_FULL_CORRECTION
TMC_ANGLE_OK
```

measurement and values are OK
Accuracy is not guaranteed, because the results are consist of measuring data which accuracy could not be verified by the system. Co-ordinates are available.
The results are not corrected by all active sensors. Co-ordinates are available.
Angle values okay, but no valid distance. Co-ordinates are not available.
### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Measurement and values are OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy. Repeat measurement.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_ANGLE_ACCURACY_GUARANTEE</td>
<td>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.</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>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.</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>No measuring, because of missing target point, co-ordinates are not available. Aim target point and try it again</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

Perform a distance measurement first before you call this function.

No measuring, because of missing target point, co-ordinates are not available. Aim target point and try it again.

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.
Example

Start a distance measure, perform measure.

```basic
DIM iRetCode AS Integer
DIM iWaitTime AS Integer
DIM Polar AS TMC_Distance_Type
DIM lError AS Logical
DIM lDone AS Logical

' start distance measurement
ON ERROR RESUME ' to get valid angles
TMC_DoMeasure( TMC_DEF_DIST )

iWaitTime = -1
lDone = FALSE
lError = FALSE

DO ' display measured values
    TMC_GetPolar( iWaitTime, Polar, iRetCode )
    SELECT CASE iRetCode
        CASE RC_OK
            ' display all data
            ' e.g. set lDone here
        CASE else
            ' handle error
            lError = TRUE
    END SELECT
LOOP UNTIL lError OR lDone

' stop distance measurement
TMC_DoMeasure( TMC_CLEAR )
```
6.3.5 TMC_GetCoordinate

Description
Calculate and read co-ordinates.

Declaration
TMC_GetCoordinate(  
  BYVAL iWaitTime AS Integer,  
  Coordinate AS TMC_COORDINATE_Type,  
  iReturnCode AS Integer )

Remarks
The function calculates and outputs co-ordinates. Angle and possibly inclination are being measured. The co-ordinates are being corrected. The result is a point in Cartesian co-ordinates. The system calculates co-ordinates and tracking co-ordinates. Simple and multiple measurements (distance-, altitude- and coordinate-tracking) are supported. The delay (iWaitTime) just works on the distance measure, not on the measuring of the angle.

As far as no new measure program is started, the results can be read. Additional to the normal return codes iReturnCode delivers also informational return codes which will not interrupt program execution.

Note
The measure program must have been started (see TMC_DoMeasure).

Parameters
iWaitTime in delay time [ms] until a result is available
  =0 returns already measured values
  >0 waits the maximal time iWaitTime for a result
Coordinate out point in Cartesian co-ordinates (output)
iReturnCode out return code, see Additional Codes

See Also
TMC_GetPolar

Additional Codes in iReturnCode
RC_OK measurement and values are OK
TMC_ACCURACY_ Accuracy is not guaranteed, because the
GUARANTEE

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.

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.

Return Codes

RC_OK
measurement and values are OK

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.

Example
Start a distance measure, perform measurement.

```basic
DIM iretCode  AS Integer
DIM iWaitTime AS Integer
DIM Coord     AS TMC_COORDINATE_Type
DIM lError    AS Logical
DIM lDone     AS Logical

ON ERROR RESUME NEXT ' to get valid angle data
TMC_DoMeasure( TMC_DEF_DIST )
lDone = FALSE
lError = FALSE

DO   ' display measured values
    TMC_GetCoordinate( 5, Coord, iRetCode )
    SELECT CASE iRetCode
        CASE RC_OK
            ' display all data
            ' e.g. set lDone
        CASE ANGLE_OK
            ' display coordinate
        CASE ELSE
            ' handle error
            lError = TRUE
    END SELECT
LOOP UNTIL lError OR lDone
TMC_DoMeasure( TMC_CLEAR )
```
6.3.6 TMC_GetAngle

**Description**  
Measure angles.

**Declaration**  
```lisp
TMC_GetAngle( Angles AS TMC_ANGLE_Type,
iReturnCode AS Integer )
```

**Remarks**  
The function measures the horizontal and vertical angle and the possibly belonging inclination, if the inclination compensation is on. If the compensation is off and no valid inclination is present, there may be a delay if the inclination can't be measured immediately. The correction values for the inclination can be calculated with several methods.

As long as no new measure program is started, the results can be read. Additional to the normal return codes `iReturnCode` delivers also informational return codes which will not interrupt program execution.

**Parameters**  
- Angles out result of measuring the angle
- iReturnCode out return code, see Additional Codes

**See Also**  
TMC_DoMeasure

**Additional Codes in iReturnCode**

- **RC_OK**  
  Execution successful.

- **TMC_NO_FULL_CORRECTION**  
  The results are not corrected by all active sensors. Angle data are available.  
  This message is to be considered 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 considered as info.

**Return Codes**

- **RC_OK**  
  angle OK
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.

**Example**

Read the currently valid angle.

```basic
DIM Angles AS TMC_ANGLE_Type
DIM RetCode AS Integer
TMC_GetAngle( Angles, RetCode )
```

### 6.3.7 TMC_GetAngle_WInc

**Description**

Measure angles with inclination control.

**Declaration**

```basic
TMC_GetAngle_WInc( 
    iIncProg AS Integer, 
    Angle AS TMC_ANGLE, 
    iReturnCode AS Integer )
```

**Remarks**

The function measures the horizontal and vertical angle and in dependence of the configuration, the inclination.

As far as no new measure program is started, the results can be read. Additional to the normal return codes `iReturnCode` delivers also informational return codes, which will not interrupt program execution.

**Parameters**

- `iIncProg` in The manner of incline compensation. Following settings are possible:
  - **Incline Program** | **Meaning**
    - TMC_MEA_INC | get inclination (apriori sigma)
    - TMC_..._... | get inclination with
## AUTO_INC

**automatism (sensor/plane)**

Get inclination always with plane

- **Angle out**
  - Result of measuring the angle
- **iReturnCode out**
  - Return code, see Additional Codes

### See Also

- TMC_DoMeasure, TMC_GetAngle

### Additional Codes in iReturnCode

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Execution successful.</td>
</tr>
</tbody>
</table>
| TMC_NO_FULL_CORRECTION | The results are not corrected by all active sensors. Angle data are available.  
                          | This message is to be considered as a 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 considered as info.                                   |

### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>angle OK</td>
</tr>
</tbody>
</table>
| 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.                                       |

### Example

Read the currently valid angle.
DIM Angles AS TMC_Angle
DIM iRetCode AS Integer

TMC_GetAngle_WInc(TMC_AUTO_INC, Angles, iRetCode)

6.3.8 TMC_QuickDist

Description: Measure slope distance and angles.

Declaration:

TMC_QuickDist(
    Angle AS TMC_HZ_V_ANG_type,
    Dist AS Distance,
    iReturnCode AS Integer
)

Remarks:
The function measures the horizontal and vertical angle and in dependence of the configuration, the inclination.

The function waits until a new distance is measured and then it returns the angle and the slope-distance, but no co-ordinates. If 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 active with the standard tracking measuring program already, the measuring program will not be 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). If 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_WInc(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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>out</td>
<td>measured Hz- and V-angle</td>
</tr>
<tr>
<td>Distance</td>
<td>out</td>
<td>measured slope-distance</td>
</tr>
<tr>
<td>iReturnCode</td>
<td>out</td>
<td>return code, see Additional Codes</td>
</tr>
</tbody>
</table>

See Also: TMC_DoMeasure, TMC_GetAngle

Additional Codes in iReturnCode

- **RC_OK**: Execution successful.
- **TMC_NO_FULL_CORRECTION**: The results are not corrected by all active sensors. Angle data are available. This message is to be considered 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 considered 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_ANGLE_OK**: Angle measuring data are valid, but no distance data available. (Possible reasons are:...
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>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)</td>
</tr>
<tr>
<td>TMC_ANGLE_ACCURACY_GUARANTEE</td>
<td>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)</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>Because of missing target point no distance data available, but the angle data are valid respectively available. Aim target point and try it again.</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>No distance measurement respectively no distance data because of wrong EDM settings. The angle data are valid. Set EDM –ppm and –mm to 0.</td>
</tr>
</tbody>
</table>

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>angle OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy. Angle data are not available. Repeat measurement.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>
Example

Fast tracking with QuickDist. See example program TRACKING for more details.

DIM iRetCode AS Integer
DIM HzV AS TMC_HZ_V_ANG_Type
DIM dDist AS Distance

TMC_DoMeasure( TMC_CLEAR ) ' clear distances

' measurement loop
DO
  ' get measurement values
  TMC_QuickDist( HzV, dDist, iRetCode )
  IF iRetCode = RC_OK OR
    iRetCode = TMC_NO_FULL_CORRECTION OR
    iRetCode = TMC_ACCURACY_GUARANTEE THEN
    ' Angles and distance are valid
    ...
  ELSE
    ' only Angles are valid
    ...
  END IF
LOOP UNTIL ....

' terminate
TMC_DoMeasure( TMC_CLEAR ) ' stop measurement

6.3.9 TMC_GetSimpleMea

Description

Gets the results of distance and angle measurement.

Declaration

TMC_GetSimpleMea(
  Angles AS TMC_HZ_V_ANG_Type,
  dSlopeDist AS Double,
  iReturnCode AS Integer )

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.

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

**Parameters**

- `Angles` out: result of measuring: the angles
- `dSlopeDist` out: slope distance [m]
- `iReturnCode` out: return code, see Additional Codes

**See Also**

- `TMC_DoMeasure`

**Additional Codes in `iReturnCode`**

- `RC_OK`: Angle OK
  - The results are not corrected by all active sensors. Angle and distance data are available.
  - This message is to be considered as warning.

- `TMC_NO_FULL_CORRECTION`: 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 force an incline measurement perform or switch off the incline.
  - This message is to be considered 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.

Return Codes

RC_OK  
Angle OK

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 aborted.
Example

This example measures the slope distance and angles.

```plaintext
DIM Angle AS Double
DIM dSlope AS Double
DIM RetCode AS Integer

TMC_GetSimpleMea( Angle, dSlope, RetCode )
```

6.3.10 TMC_Get/SetAngleFaceDef

**Description**
Gets and sets the current face definition.

**Declaration**

```plaintext
TMC_GetAngleFaceDef( eFaceDef AS Integer )
TMC_SetAngleFaceDef(  
    ByVal eFaceDef AS Integer )
```

**Remarks**

**Note**
No distance may exist for setting the face definition. Call TMC_DoMeasure(TMC_CLEAR) before this function.

**Parameters**

- **eFaceDef**
  - **out/in**
  - TMC_FACE_NORMAL or TMC_FACE_TURN

**See Also**

- 

**Return Codes**

- **RC_OK**
  - Completed successfully.
- **TMC_BUSY**
  - Measurement system is busy (no valid results) or a distance exists
Example

The example reads the current definition and sets the opposite one.

```plaintext
DIM face AS TMC_FACE_DEF
TMC_GetAngelFaceDef(face)
IF (face = TMC_FACE_NORMAL) THEN
    TMC_SetAngelFaceDef(TMC_FACE_TURN)
ELSE
    TMC_SetAngelFaceDef(TMC_FACE_NORMAL)
END IF
```

6.3.11 TMC_Get/SetHzOffset

Description

Gets and sets the current horizontal offset.

Declaration

```plaintext
TMC_GetHzOffset( dHzOffset AS Double )
TMC_SetHzOffset( ByVal dHzOffset AS Double )
```

Remarks

Note

No distance may exist for setting the Hz-offset. Call `TMC_DoMeasure(TMC_CLEAR)` before this function.

Parameters

dHzOffset out/in Horizontal offset in radiant.

See Also

-

Return Codes

```plaintext
RC_OK Completed successfully.
TMC_BUSY measurement system is busy (no valid results) or a distance exists
```

Example

The example reads the current offsets and sets it to an increased value.
DIM off AS Double

TMC_GetHzOffset ( off )
TMC_SetHzOffset ( off + 1.0 )

6.3.12 TMC_Get/SetDistPpm

Description
Gets and sets the PPM values for distance measurement corrections.

Declaration
TMC_GetDistPpm( PpmCorr AS TMC_PPM_CORR_Type)
TMC_SetDistPpm( PpmCorr AS TMC_PPM_CORR_Type)

Parameters
PpmCorr out/in PPM values for distance measurement corrections.

Return Codes
RC_OK Completed successfully.
TMC_BUSY TMC is in use and can not be changed.

Example -
6.3.13 TMC_Get/SetGeomProjection

**Description**
Gets and sets the projection part of distance measurement corrections.

**Declaration**

```plaintext
TMC_GetGeomProjection ( GeomProj AS TMC_GEOM_PROJECTION_Type)
TMC_SetGeomProjection ( GeomProj AS TMC_GEOM_PROJECTION_Type)
```

**Parameters**

- **GeomProj** out/in
  Projection (distance to the reference, factor of projection, earth radius).

**Return Codes**

- **RC_OK** Completed successfully.
- **TMC_BUSY** TMC is in use and cannot be changed.

**Example**

- 

6.3.14 TMC_Get/SetGeomReduction

**Description**
Gets and sets the reduction to the reference part of distance measurement corrections.

**Declaration**

```plaintext
TMC_GetGeomReduction ( GeomRed AS TMC_GEOM_REDUCTION_Type)
TMC_SetGeomReduction ( GeomRed AS TMC_GEOM_REDUCTION_Type)
```

**Parameters**

- **GeomRed** out/in
  Reduction to the reference (reference height, earth radius).

**Return Codes**

- **RC_OK** Completed successfully.
- **TMC_BUSY** TMC is in use and cannot be changed.

**Example**

- 

6-122
6.3.15 TMC_Get/SetAtmCorr

**Description**
Gets and sets the atmosphere part of distance measurement corrections.

**Declaration**
```plaintext
TMC_GetAtmCorr ( AtmCorr AS TMC_ATM_TEMPERATURE_Type)
TMC_SetAtmCorr ( AtmCorr AS TMC_ATM_TEMPERATURE_Type)
```

**Parameters**
- `AtmCorr` out/in Atmosphere

**Return Codes**
- **RC_OK**: Completed successfully.
- **TMC_BUSY**: TMC is in use and cannot be changed.

**Example**
```
```

6.3.16 TMC_Get/SetHeight

**Description**
Gets and sets the current height of the reflector.

**Declaration**
```plaintext
TMC_GetHeight ( Height AS Double )
TMC_SetHeight ( ByVal Height AS Double )
```

**Parameters**
- `Height` out/in Height of reflector in Meters.

**Return Codes**
- **RC_OK**: Completed successfully.
- **TMC_BUSY**: Measurement system is busy (no valid results)

**Example**
The example sets the reflectors height to the value of 1.0 m.
```
TMC_SetHeight ( 1.0 )
```
### 6.3.17 TMC_Get/SetRefractiveCorr

**Description**
Gets and sets the refractive correction for measuring the distance.

**Declaration**

```plaintext
TMC_GetRefractiveCorr (Refraction AS TMC_REFRACTION_Type)
TMC_SetRefractiveCorr (Refraction AS TMC_REFRACTION_Type)
```

**Parameters**

| Refraction | out/in | Refraction correction value(s). |

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Completed successfully.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>measurement system is busy (no valid results)</td>
</tr>
</tbody>
</table>

**Example**

- 

### 6.3.18 TMC_Get/SetRefractiveMethod

**Description**
Gets and sets the method of refractive correction for measuring the distance.

**Declaration**

```plaintext
TMC_GetRefractiveMethod (Method AS Integer)
TMC_SetRefractiveMethod (byVal Method AS Integer)
```

**Parameters**

<table>
<thead>
<tr>
<th>Method</th>
<th>out/in</th>
<th>Method of refraction calculation:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1: method 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: method 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>else: undefined</td>
</tr>
</tbody>
</table>

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Completed successfully.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>measurement system is busy (no valid results)</td>
</tr>
</tbody>
</table>
6.3.19 TMC_Get/SetStation

Description
Gets and sets station co-ordinates.

Declaration
TMC_GetStation ( 
    Station AS TMC_STATION_Type )
TMC_SetStation ( 
    Station AS TMC_STATION_Type )

Remarks
Note
No distance may exist for setting a new station. Call
TMC_DoMeasure(TMC_CLEAR) before this function.

Parameters
Station out/in Station co-ordinates.

Return Codes
RC_OK Completed successfully.
TMC_BUSY measurement system is busy (no valid results)
or a distance exists.

Example
-

6.3.20 TMC_IfDistTapeMeasured

Description
Gets information about manual measurement.

Declaration
TMC_IfDistTapeMeasured ( 
    bTapeMeasured AS Logical )

Parameters
bTapeMeasured out TRUE: if measurement has been
done by hand.
FALSE: if measurement has been
done with EDM or if invalid.

Return Codes
RC_OK Completed successfully.

Example
-
6.3.21 TMC_SetHandDist

Description  Sets distance manually.

Declaration  TMC_SetHandDist(
            byVal dSlopeDistance  AS Double,
            byVal dHgtOffset      AS Double)

Parameters  
  dSlopeDistance  in  slope distance [m]
  dHgtOffset      in  Height to measured point. [m]

See Also  -

Return Codes  

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Execution successful.</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. This message is to be considers as warning.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>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.</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. At repeated occur call service.</td>
</tr>
</tbody>
</table>


6.3.22 TMC_SetDistSwitch

**Description**
Defines the distance measurement correction switches.

**Declaration**
```
TMC_SetDistSwitch(
    Switches AS TMC_DIST_SWITCH_Type )
```

**Remarks**
This procedure sets the distance measurement correction switches.

**Parameters**
- **Switches** in Distance switches

**Return-Codes**
- **RC_OK** Successful termination.

**See Also**
TMC_GetDistSwitch

6.3.23 TMC_GetDistSwitch

**Description**
Returns the distance measurement correction switches.

**Declaration**
```
TMC_GetDistSwitch(
    Switches AS TMC_DIST_SWITCH_Type )
```

**Remarks**
This procedure returns the distance measurement correction switches.

**Parameters**
- **Switches** out Distance switches
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See Also  TMC_SetDistSwitch

### 6.3.24 TMC_SetOffsetDist

**Description**
Defines the distance measurement offset.

**Declaration**

```pascal
TMC_SetOffsetDist(
    Offsets AS TMC_OFFSET_DIST_Type )
```

**Remarks**
This procedure defines the offset to the prism pole. The `dLengthVal` defines the offset away from the prism pole, positive means in the line from instrument to prism. `dCrossVal` means right from the prism pole and `dHeightVal` means higher than prism pole.

**Remarks**

| Note | No distance may exist for offset setting. Call TMC_DoMeasure(TMC_CLEAR) before this function. |

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offsets</td>
<td>in Target point offset</td>
</tr>
</tbody>
</table>

**Return-Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>measurement system is busy (no valid results) or a distance exists.</td>
</tr>
</tbody>
</table>

See Also  TMC_GetOffsetDist, BAP_Offset, TMC_IfOffsetDistMeasured
### 6.3.25 TMC_GetOffsetDist

**Description**
Returns the distance measurement offset.

**Declaration**
```plaintext
TMC_GetOffsetDist(
    Offsets AS TMC_OFFSET_DIST_Type )
```

**Remarks**
This procedure returns the actual offset to the prism pole. The `dLengthVal` defines the offset away from the prism pole, positive means in the line from instrument to prism. `dCrossVal` means right from the prism pole and `dHeightVal` means higher than prism pole.

**Parameters**
- `Offsets` out Target point offset

**Return-Codes**
- `RC_OK` Successful termination.

**See Also**
- TMC_SetOffsetDist, BAP_Offset, TMC_IsOffsetDistMeasured

### 6.3.26 TMC_IsOffsetDistMeasured

**Description**
Returns the EDM measurement mode.

**Declaration**
```plaintext
TMC_IsOffsetDistMeasured(
    lOffset AS Logical )
```

**Remarks**
This function returns TRUE if an offset is defined.

**Parameters**
- `lOffset` out Offset is valid

**Return-Codes**
- `RC_OK` Successful termination.

**See Also**
- TMC_SetOffsetDist, TMC_GetOffsetDist, BAP_Offset
6.3.27 TMC_GetFace1

Description: Get face information of current telescope position.

Declaration: TMC_GetFace1( lFace1 AS Logical )

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_GetFace1 call). Note that the instrument automatically turns into an inactive measurement state after a predefined timeout.

Parameters:

lFace1 out
  TRUE: Face I
  FALSE: Face II

Return-Codes:

RC_OK Successful termination.

6.3.28 TMC_SetAngSwitch

Description: Defines the angle measurement correction switches.

Declaration: TMC_SetAngSwitch(
  Switches AS TMC_ANG_SWITCH_Type )

Remarks: This procedure sets the angle measurement correction switches. No distance may exist for setting the angle switches. Call TMC_DoMeasure( TMC_CLEAR ) before this function.

Parameters:

Switches in angular switches

Return-Codes:

RC_OK Successful termination.
  TMC_BUSY A distance exists

See Also: TMC_GetAngSwitch
Example

Change switches

```
DIM AngSwitches AS TMC_ANG_SWITCH_Type

TMC_DoMeasure( TMC_CLEAR ) ' clear distances
TMC_GetAngSwitch( AngSwitches )
AngSwitches.lInclineCorr = TRUE
AngSwitches.lCollimationCorr = FALSE
TMC_SetAngSwitch( AngSwitches )
```

### 6.3.29 TMC_GetAngSwitch

**Description**

Returns the angle measurement correction switches.

**Declaration**

```
TMC_GetAngSwitch(
    Switches AS TMC_ANG_SWITCH_Type )
```

**Remarks**

This procedure returns the actual angle measurement correction switches.

**Parameters**

- `Switches` in
  Angular switches

**Return-Codes**

- `RC_OK` Successful termination.

**See Also**

TMC_SetAngSwitch

### 6.3.30 TMC_SetInclineSwitch

**Description**

Defines the compensator switch.

**Declaration**

```
TMC_SetAngSwitches( 1On AS Logical )
```

**Remarks**

This procedure enables or disables the dual axis compensator correction.

**Note**

No distance may exist for a switch setting. Call `TMC_DoMeasure(TMC_CLEAR)` before this function.

**Parameters**

- `1On` in
  Switch
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>A distance exists</td>
</tr>
</tbody>
</table>

See Also TMC_GetInclineSwitch

6.3.31 TMC_GetInclineSwitch

Description Returns the compensator switch.

Declaration TMC_GetInclineSwitches( lOn AS Logical )

Remarks This procedure returns the dual axis compensator correction state.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lOn</td>
<td>out</td>
<td>Switch</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See Also TMC_SetInclineSwitch

6.3.32 TMC_GetInclineStatus

Description Returns the inclination compensator status.

Declaration TMC_GetInclineStatus( iStatus AS Integer )

Remarks This procedure returns status of the inclination sensor.

Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iStatus</td>
<td>out</td>
<td>TMC_INC_OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Incline-sensor is switched off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMC_INC_OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inclination is ok, recording is allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMC_INC_TILT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Incline-sensor is out of working area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TMC_INC_OLD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Incline-values are not yet updated</td>
</tr>
</tbody>
</table>
TMC_INC_FAIL  Inclination - measurement fails

Return-Codes

RC_OK  Successful termination.

See Also  TMC_SetInclineSwitch

Example  See example file „meas.gbs“.
6.4 FUNCTIONS FOR GSI

6.4.1 Summarizing Lists of GSI Types and Procedures

6.4.1.1 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi_List</td>
<td>Array of GSI_WiDlg_Entry_Type.</td>
</tr>
<tr>
<td>GSI_Point_Coord_Type</td>
<td>Point co-ordinate data.</td>
</tr>
<tr>
<td>GSI_Rec_Id_List</td>
<td>Record mask array of integers (indicating WI-identifications)</td>
</tr>
<tr>
<td>GSI_WiDlg_Entry_Type</td>
<td>Dialog entry information.</td>
</tr>
</tbody>
</table>

6.4.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_Coding</td>
<td>Starts the active coding function of the TPS system.</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>Returns if distance tracking is running.</td>
</tr>
<tr>
<td>GSI_CreateMDlg</td>
<td>Creates and shows the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_DefineMDlg</td>
<td>Defines the entries of the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_DefineRecMaskDlg</td>
<td>Defines the recording mask dialog.</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>Executes an automatic distance measurement.</td>
</tr>
<tr>
<td>GSI_ExecQCoding</td>
<td>Executes the Quick-Coding.</td>
</tr>
<tr>
<td>GSI_GetDataPath</td>
<td>Get the name of the file with the import data.</td>
</tr>
<tr>
<td>GSI_GetIndivNr</td>
<td>Fetches the individual point number.</td>
</tr>
<tr>
<td>GSI_GetLineSysMDlg</td>
<td>Gets the definition of a line in the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>Returns the number of the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>This routine returns the status for Quick-</td>
</tr>
</tbody>
</table>

6-134
<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_GetRecMask</td>
<td>Coding.</td>
</tr>
<tr>
<td>Get the definition and the</td>
<td>format of a recording mask.</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>Returns the used recording mask.</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>Returns the recording order for Quick-Coding.</td>
</tr>
<tr>
<td>GSI_GetRecPath</td>
<td>Returns the recording path</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>Fetches the running point number and the increment.</td>
</tr>
<tr>
<td>GSI_GetWiEntryText</td>
<td>Get text-data from the Theodolite data pool.</td>
</tr>
<tr>
<td>GSI_GetWiEntry</td>
<td>Get data from the Theodolite data pool.</td>
</tr>
<tr>
<td>GSI_ImportCoordDlg</td>
<td>Show the co-ordinate import dialog.</td>
</tr>
<tr>
<td>GSI_IncPNumber</td>
<td>Automatically point number increment.</td>
</tr>
<tr>
<td>GSI_IsRunningNr</td>
<td>Queries if running number is being used.</td>
</tr>
<tr>
<td>GSI_ManCoordDlg</td>
<td>Show the manual co-ordinate input dialog.</td>
</tr>
<tr>
<td>GSI_Measure</td>
<td>Entry point for measure and registration dialog (measure and registration).</td>
</tr>
<tr>
<td>GSI_QuickSet</td>
<td>Show the Quickset dialog</td>
</tr>
<tr>
<td>GSI_RecordRecMask</td>
<td>Recording the given wi mask.</td>
</tr>
<tr>
<td>GSI_SelectCode</td>
<td>This routine shows the codelist-coding dialog.</td>
</tr>
<tr>
<td>GSI_SetDataPath</td>
<td>Set the file with the import data.</td>
</tr>
<tr>
<td>GSI_SetIndivNr</td>
<td>Sets the individual point number.</td>
</tr>
<tr>
<td>GSI_SetIvPtNrStatus</td>
<td>Switches the individual point number mode on/off.</td>
</tr>
<tr>
<td>GSI_SetLineMDlg</td>
<td>Sets one line in the user definable measurement dialog to system parameter.</td>
</tr>
<tr>
<td>GSI_SetLineMDlgPar</td>
<td>Sets a line in the user definable measurement dialog to an application parameter.</td>
</tr>
<tr>
<td>GSI_SetLineMDlgText</td>
<td>Puts a textline into the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_SetLineSysMDlg</td>
<td>Sets a line in the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_SetMDlgNr</td>
<td>Sets the number of the system measurement</td>
</tr>
</tbody>
</table>
procedure name | description
--- | ---
GSI_SetQCodeMode | Sets the Quick-Coding mode.
GSI_SetRecMask | Set the definition and the format of a recording mask.
GSI_SetRecMaskNr | Set the used recording mask.
GSI_SetRecOrder | Sets the recording order for Quick-Coding.
GSI_SetRecPath | Defines the recording path.
GSI_SetRunningNr | Sets the running point number and increment.
GSI_SetWiEntry | Set data to the Theodolite data pool.
GSI_UpdateMDlg | Updates the user definable measurement dialog.
GSI_UpdateMeasurement | Update the measurement data.

### 6.4.2 Constants for WI values

Definitions for WI values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_PTNR</td>
<td>String</td>
<td>Point number</td>
</tr>
<tr>
<td>GSI_ID_FNR</td>
<td>Double</td>
<td>Serial number</td>
</tr>
<tr>
<td>GSI_ID_TYPE</td>
<td>String</td>
<td>Device type</td>
</tr>
<tr>
<td>GSI_ID_TIME_1</td>
<td>String</td>
<td>First time art</td>
</tr>
<tr>
<td>GSI_ID_TIME_2</td>
<td>String</td>
<td>Second time art</td>
</tr>
<tr>
<td>GSI_ID_HZ</td>
<td>Double</td>
<td>Horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_V</td>
<td>Double</td>
<td>Vertical angle</td>
</tr>
<tr>
<td>GSI_ID_NHZ</td>
<td>Double</td>
<td>Nominal horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_DHZ</td>
<td>Double</td>
<td>Difference horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_NV</td>
<td>Double</td>
<td>Nominal vertical angle</td>
</tr>
<tr>
<td>GSI_ID_DV</td>
<td>Double</td>
<td>Difference vertical angle</td>
</tr>
<tr>
<td>Name</td>
<td>Data Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>GSI_ID_SLOPE</td>
<td>Double</td>
<td>Slope distance</td>
</tr>
<tr>
<td>GSI_ID_HOR</td>
<td>Double</td>
<td>Horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_HGT</td>
<td>Double</td>
<td>Height difference</td>
</tr>
<tr>
<td>GSI_ID_NHOR</td>
<td>Double</td>
<td>Nominal horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_DHOR</td>
<td>Double</td>
<td>Difference horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_NHGT</td>
<td>Double</td>
<td>Nominal height difference</td>
</tr>
<tr>
<td>GSI_ID_DHGT</td>
<td>Double</td>
<td>Difference height difference</td>
</tr>
<tr>
<td>GSI_ID_NSLOPE</td>
<td>Double</td>
<td>Nominal slope distance</td>
</tr>
<tr>
<td>GSI_ID_DSLOPE</td>
<td>Double</td>
<td>Difference slope distance</td>
</tr>
<tr>
<td>GSI_ID_CODE</td>
<td>String</td>
<td>Code information</td>
</tr>
<tr>
<td>GSI_ID_CODE_1</td>
<td>String</td>
<td>Information 1</td>
</tr>
<tr>
<td>GSI_ID_CODE_2</td>
<td>String</td>
<td>Information 2</td>
</tr>
<tr>
<td>GSI_ID_CODE_3</td>
<td>String</td>
<td>Information 3</td>
</tr>
<tr>
<td>GSI_ID_CODE_4</td>
<td>String</td>
<td>Information 4</td>
</tr>
<tr>
<td>GSI_ID_CODE_5</td>
<td>String</td>
<td>Information 5</td>
</tr>
<tr>
<td>GSI_ID_CODE_6</td>
<td>String</td>
<td>Information 6</td>
</tr>
<tr>
<td>GSI_ID_CODE_7</td>
<td>String</td>
<td>Information 7</td>
</tr>
<tr>
<td>GSI_ID_CODE_8</td>
<td>String</td>
<td>Information 8</td>
</tr>
<tr>
<td>GSI_ID_PPMM</td>
<td>String</td>
<td>mm and ppm</td>
</tr>
<tr>
<td>GSI_ID_SIGMA</td>
<td>String</td>
<td>Distance count and deviation</td>
</tr>
<tr>
<td>GSI_ID_MM</td>
<td>Double</td>
<td>mm</td>
</tr>
<tr>
<td>GSI_ID_PPM</td>
<td>Double</td>
<td>ppm</td>
</tr>
<tr>
<td>GSI_ID_REM_1</td>
<td>String</td>
<td>Remark 1</td>
</tr>
<tr>
<td>GSI_ID_REM_2</td>
<td>String</td>
<td>Remark 2</td>
</tr>
<tr>
<td>GSI_ID_REM_3</td>
<td>String</td>
<td>Remark 3</td>
</tr>
<tr>
<td>GSI_ID_REM_4</td>
<td>String</td>
<td>Remark 4</td>
</tr>
<tr>
<td>GSI_ID_REM_5</td>
<td>String</td>
<td>Remark 5</td>
</tr>
<tr>
<td>GSI_ID_REM_6</td>
<td>String</td>
<td>Remark 6</td>
</tr>
<tr>
<td>GSI_ID_REM_7</td>
<td>String</td>
<td>Remark 7</td>
</tr>
<tr>
<td>GSI_ID_REM_8</td>
<td>String</td>
<td>Remark 8</td>
</tr>
<tr>
<td>GSI_ID_REM_9</td>
<td>String</td>
<td>Remark 9</td>
</tr>
<tr>
<td>Name</td>
<td>Data Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>GSI_ID_E</td>
<td>Double</td>
<td>East co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_N</td>
<td>Double</td>
<td>North co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_H</td>
<td>Double</td>
<td>Height</td>
</tr>
<tr>
<td>GSI_ID_E0</td>
<td>Double</td>
<td>East station co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_N0</td>
<td>Double</td>
<td>North station co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_H0</td>
<td>Double</td>
<td>Station height</td>
</tr>
<tr>
<td>GSI_ID_HR</td>
<td>Double</td>
<td>Reflector height</td>
</tr>
<tr>
<td>GSI_ID_HI</td>
<td>Double</td>
<td>Instrument height</td>
</tr>
<tr>
<td>GSI_ID_INDIV</td>
<td>String</td>
<td>Individual point number</td>
</tr>
<tr>
<td>GSI_ID_PTLA</td>
<td>String</td>
<td>Number of the last recorded point</td>
</tr>
<tr>
<td>GSI_ID_STEP</td>
<td>Double</td>
<td>Increment of the running point number</td>
</tr>
<tr>
<td>GSI_ID_SPTNR</td>
<td>String</td>
<td>Station point number</td>
</tr>
<tr>
<td>GSI_ID_SHZ</td>
<td>Double</td>
<td>Hz angle with no sign change</td>
</tr>
<tr>
<td>GSI_ID_CD_DSC</td>
<td>String</td>
<td>Code description</td>
</tr>
<tr>
<td>GSI_ID_PTCD_DSC</td>
<td>String</td>
<td>Point code description</td>
</tr>
<tr>
<td>GSI_ID_PV_CD</td>
<td>String</td>
<td>Preview code</td>
</tr>
<tr>
<td>GSI_ID_PV_PTCD</td>
<td>String</td>
<td>Preview point code</td>
</tr>
<tr>
<td>GSI_ID_ACT_PTID</td>
<td>String</td>
<td>Actual point ID</td>
</tr>
<tr>
<td>GSI_ID_BACKID</td>
<td>String</td>
<td>Backside ID</td>
</tr>
<tr>
<td>GSI_ID_APPDATA0</td>
<td>String/Double</td>
<td>Application data 0</td>
</tr>
<tr>
<td>GSI_ID_APPDATA1</td>
<td>String/Double</td>
<td>Application data 1</td>
</tr>
<tr>
<td>GSI_ID_APPDATA2</td>
<td>String/Double</td>
<td>Application data 2</td>
</tr>
<tr>
<td>GSI_ID_APPDATA3</td>
<td>String/Double</td>
<td>Application data 3</td>
</tr>
<tr>
<td>GSI_ID_APPDATA4</td>
<td>String/Double</td>
<td>Application data 4</td>
</tr>
<tr>
<td>GSI_ID_APPDATA5</td>
<td>String/Double</td>
<td>Application data 5</td>
</tr>
<tr>
<td>GSI_ID_APPDATA6</td>
<td>String/Double</td>
<td>Application data 6</td>
</tr>
<tr>
<td>GSI_ID_APPDATA7</td>
<td>String/Double</td>
<td>Application data 7</td>
</tr>
<tr>
<td>GSI_ID_APPDATA8</td>
<td>String/Double</td>
<td>Application data 8</td>
</tr>
<tr>
<td>GSI_ID_APPDATA9</td>
<td>String/Double</td>
<td>Application data 9</td>
</tr>
<tr>
<td>GSI_ID_APPDATA10</td>
<td>String/Double</td>
<td>Application data 10</td>
</tr>
<tr>
<td>GSI_ID_APPDATA11</td>
<td>String/Double</td>
<td>Application data 11</td>
</tr>
</tbody>
</table>
### 6.4.3 Constants for Measurement Dialog Definition

Definition of (user definable) application parameters for measurement dialogs, either Double or String. See also GSI_SetLineMDlgPar and GSI_SetLineMDlgText.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_AppData0</td>
<td>Application parameter 0</td>
</tr>
<tr>
<td>GSI_PAR_AppData1</td>
<td>Application parameter 1</td>
</tr>
<tr>
<td>GSI_PAR_AppData2</td>
<td>Application parameter 2</td>
</tr>
<tr>
<td>GSI_PAR_AppData3</td>
<td>Application parameter 3</td>
</tr>
<tr>
<td>GSI_PAR_AppData4</td>
<td>Application parameter 4</td>
</tr>
<tr>
<td>GSI_PAR_AppData5</td>
<td>Application parameter 5</td>
</tr>
<tr>
<td>GSI_PAR_AppData6</td>
<td>Application parameter 6</td>
</tr>
<tr>
<td>GSI_PAR_AppData7</td>
<td>Application parameter 7</td>
</tr>
<tr>
<td>GSI_PAR_AppData8</td>
<td>Application parameter 8</td>
</tr>
<tr>
<td>GSI_PAR_AppData9</td>
<td>Application parameter 9</td>
</tr>
</tbody>
</table>
### System Functions

#### Definition of system (defined) parameters for measurement dialogs. See also `GSI_SetLineSysMDlg` and `GSI_SetLineMDlg`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GSI_PAR_AppData10</code></td>
<td>Application parameter 10</td>
</tr>
<tr>
<td><code>GSI_PAR_AppData11</code></td>
<td>Application parameter 11</td>
</tr>
<tr>
<td><code>GSI_PAR_AddConst</code></td>
<td>Prism constant</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib1</code></td>
<td>Point Code Attribute 1</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib2</code></td>
<td>Point Code Attribute 2</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib3</code></td>
<td>Point Code Attribute 3</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib4</code></td>
<td>Point Code Attribute 4</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib5</code></td>
<td>Point Code Attribute 5</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib6</code></td>
<td>Point Code Attribute 6</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib7</code></td>
<td>Point Code Attribute 7</td>
</tr>
<tr>
<td><code>GSI_PAR_Attrib8</code></td>
<td>Point Code Attribute 8</td>
</tr>
<tr>
<td><code>GSI_PAR_AvgMeasNo</code></td>
<td>Maximal number of distance measurements of the average mode</td>
</tr>
<tr>
<td><code>GSI_PAR_BacksideId</code></td>
<td>Last used Backside</td>
</tr>
<tr>
<td><code>GSI_PAR_Code</code></td>
<td>Last used Code</td>
</tr>
<tr>
<td><code>GSI_PAR_CodeDescr</code></td>
<td>Last used free Code Description</td>
</tr>
<tr>
<td><code>GSI_PAR_CodeList</code></td>
<td>Codelist management (select, create etc)</td>
</tr>
<tr>
<td><code>GSI_PAR_CodeListSelect</code></td>
<td>Codelist selection (of an existing codelist)</td>
</tr>
<tr>
<td><code>GSI_PAR_DataJobSelect</code></td>
<td>Data job selection (of an existing job)</td>
</tr>
<tr>
<td><code>GSI_PAR_Date</code></td>
<td>Current date of the instrument. The displayed format depends on the setting of the parameter &quot;Date form.&quot;</td>
</tr>
<tr>
<td><code>GSI_PAR_DisplayMask</code></td>
<td>Select display mask for standard measuring dialog. Max. 3 displaymasks can be defined for this dialog. The displaymasks can also be changed with the system function &quot;Next Displaymask&quot;.</td>
</tr>
<tr>
<td><code>GSI_PAR_DataJob</code></td>
<td>Data job management (select, create etc)</td>
</tr>
<tr>
<td><code>GSI_PAR_TargetEast</code></td>
<td>Target point Easting</td>
</tr>
<tr>
<td>Name</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_PAR_DistMeasProg</td>
<td>EDM measurement program selection. Attention: The available measurement programs depends on the selected target type and on the instrument type</td>
</tr>
<tr>
<td>GSI_PAR_TargetElev</td>
<td>Target point Elevation</td>
</tr>
<tr>
<td>GSI_PAR_ElevDiff</td>
<td>Elevation difference</td>
</tr>
<tr>
<td>GSI_PAR_HalfLineSpace</td>
<td>This item can be used to display a half line space in order to separate or group lines on instrument screen.</td>
</tr>
<tr>
<td>GSI_PAR_DistHoriz</td>
<td>Horizontal distance</td>
</tr>
<tr>
<td>GSI_PAR_AngleHz</td>
<td>Hz-Angle</td>
</tr>
<tr>
<td>GSI_PAR_PointIdIncr</td>
<td>defines the increment step. It is used to increment the Target Point Id after recording a target point.</td>
</tr>
<tr>
<td>GSI_PAR_IndivPointId</td>
<td>Individual point identifier</td>
</tr>
<tr>
<td>GSI_PAR_Info1</td>
<td>Shows the Free Code Info 1</td>
</tr>
<tr>
<td>GSI_PAR_Info2</td>
<td>Shows the Free Code Info 2</td>
</tr>
<tr>
<td>GSI_PAR_Info3</td>
<td>Shows the Free Code Info 3</td>
</tr>
<tr>
<td>GSI_PAR_Info4</td>
<td>Shows the Free Code Info 4</td>
</tr>
<tr>
<td>GSI_PAR_Info5</td>
<td>Shows the Free Code Info 5</td>
</tr>
<tr>
<td>GSI_PAR_Info6</td>
<td>Shows the Free Code Info 6</td>
</tr>
<tr>
<td>GSI_PAR_Info7</td>
<td>Shows the Free Code Info 7</td>
</tr>
<tr>
<td>GSI_PAR_Info8</td>
<td>Shows the Free Code Info 8</td>
</tr>
<tr>
<td>GSI_PAR_InstrHeight</td>
<td>Instrument Height (hi)</td>
</tr>
<tr>
<td>GSI_PAR_LastPointId</td>
<td>Last recorded target point identifier</td>
</tr>
<tr>
<td>GSI_PAR_MeasJobSelect</td>
<td>Measurement Job selection (of an existing Job or RS232 for online recording)</td>
</tr>
<tr>
<td>GSI_PAR_MeasJob</td>
<td>Measurement Job management (select, create, etc.)</td>
</tr>
<tr>
<td>GSI_PAR_NS</td>
<td>Number of measurements and standard deviation</td>
</tr>
<tr>
<td>GSI_PAR_TargetNorth</td>
<td>Target point Northing</td>
</tr>
<tr>
<td>GSI_PAR_OffsetCross</td>
<td>Cross Offset</td>
</tr>
<tr>
<td>Name</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_PAR_OffsetElev</td>
<td>Offset Elevation</td>
</tr>
<tr>
<td>GSI_PAR_OffsetLength</td>
<td>Offset Length</td>
</tr>
<tr>
<td>GSI_PAR_OffsetMode</td>
<td>Defines the resetting of the offset</td>
</tr>
<tr>
<td>GSI_PAR_PointCode</td>
<td>Actual Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_PointId</td>
<td>Actual Target point identifier, running or individual. The Value and the display text changes if an individual number is set.</td>
</tr>
<tr>
<td>GSI_PAR_PpmAtm</td>
<td>ppm atmospheric</td>
</tr>
<tr>
<td>GSI_PAR_PpmGeom</td>
<td>ppm geometric</td>
</tr>
<tr>
<td>GSI_PAR_PpmTotal</td>
<td>Total ppm</td>
</tr>
<tr>
<td>GSI_PAR_PpmMm</td>
<td>Total ppm and prism constant</td>
</tr>
<tr>
<td>GSI_PAR_PrevCode</td>
<td>Shows the second last used Code</td>
</tr>
<tr>
<td>GSI_PAR_PrevPointCode</td>
<td>Last used Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_PointCodeDescr</td>
<td>Shows the Point Code Description of the actual Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_RecMask</td>
<td>Selected Recording mask for target point measurements</td>
</tr>
<tr>
<td>GSI_PAR_ReflHeight</td>
<td>Reflector height (hr)</td>
</tr>
<tr>
<td>GSI_PAR_ReflName</td>
<td>Used reflector type</td>
</tr>
<tr>
<td>GSI_PAR_ReflSelection</td>
<td>reflector type selection. If there are user defined prism, then they will be added to this list. The User Refl1..User Refl3 are only valid, if these user definable prisms are defined.</td>
</tr>
<tr>
<td>GSI_PAR_RunningPointId</td>
<td>Running target point identifier</td>
</tr>
<tr>
<td>GSI_PAR_DistSlope</td>
<td>Slope distance</td>
</tr>
<tr>
<td>GSI_PAR_StationId</td>
<td>Identifies the Station</td>
</tr>
<tr>
<td>GSI_PAR_StationEast</td>
<td>Station Easting</td>
</tr>
<tr>
<td>GSI_PAR_StationElev</td>
<td>Station Elevation</td>
</tr>
<tr>
<td>GSI_PAR_StationNorth</td>
<td>Station Northing</td>
</tr>
<tr>
<td>GSI_PAR_TargetType</td>
<td>Definition of the target type (Reflector / reflectorless)</td>
</tr>
<tr>
<td>GSI_PAR_Time</td>
<td>Current time of the instrument. The displayed format depends on the setting of the</td>
</tr>
</tbody>
</table>
6.4.4 Relationship of GSI_ID’s to GSI_PAR’s

In general we can distinguish between two data value pools who are able to store values in it. Some of these values are shared between the two pools.

GSI_ID_-Ids describe the values which can be stored and requested in the (WI) data value pool. GSI_PAR_-Ids describe the values which can be used for displaying in a measurement dialog. Their sets of id's are not associated directly in all cases. Moreover their sets of Id's can be distinguished in their meaning.

Association in this context means that both pools, the data value pool and the data display pool, share their values directly. Nonassociated values are unique to either the data value pool or the data display pool.

Many of the GSI_IDs are record-able. Two types of record-able Ids can be distinguished:

a) Measurement block (“Meas”) (has to start with a GSI_ID_PTNR)
   b) Code block (“Code”) (has to start with a GSI_ID_CODE)

They may not be mixed.

<table>
<thead>
<tr>
<th>Record-able</th>
<th>GSI_ID_-Ids</th>
<th>GSI_PAR_-Ids</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GSI_ID_NHZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSI_ID_DHZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSI_ID_NV</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_AngleV</td>
<td>V-Angle</td>
</tr>
<tr>
<td>GSI_PAR_VangleFormat</td>
<td>Vertical angle display format: Zenith angle = 0gon for zenith, angles are positive. Elev. angle = 0gon for horizontal, (+) above horizon and (-) below horizon. Elev.angle% = 0% for horizon, 100% for 50gon. V-angle is displayed (+) above and (-) below horizon but as percentage of the gradient.</td>
</tr>
<tr>
<td>GSI_PAR_NONE</td>
<td>Designates a line that is unused.</td>
</tr>
<tr>
<td>GSI_ID</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>DV</td>
<td>GSI_PAR_IndivPointId</td>
</tr>
<tr>
<td>NHOR</td>
<td>GSI_PAR_LastPointId</td>
</tr>
<tr>
<td>DHOR</td>
<td>GSI_PAR_PointIdIncr</td>
</tr>
<tr>
<td>NMGT</td>
<td>GSI_PAR_StationId</td>
</tr>
<tr>
<td>DHGT</td>
<td>GSI_PAR_BackSideId</td>
</tr>
<tr>
<td>NSLOPE</td>
<td>GSI_PAR_CodeDescr</td>
</tr>
<tr>
<td>DSLOPE</td>
<td>GSI_PAR_PointCodeDescr</td>
</tr>
<tr>
<td>INDIV</td>
<td>GSI_PAR_PointCode</td>
</tr>
<tr>
<td>PTLA</td>
<td>GSI_PAR_PointId</td>
</tr>
<tr>
<td>STEP</td>
<td>GSI_PAR_BackSideId</td>
</tr>
<tr>
<td>SPTNR</td>
<td>GSI_PAR_PrevCodeDescr</td>
</tr>
<tr>
<td>SHZ</td>
<td>GSI_PAR_PrevPointCode</td>
</tr>
<tr>
<td>CD_DSC</td>
<td>GSI_PAR_PointDescr</td>
</tr>
<tr>
<td>PTCD_DSC</td>
<td>GSI_PAR_BackSideDescr</td>
</tr>
<tr>
<td>PV_CD</td>
<td>GSI_PAR_PrevCode</td>
</tr>
<tr>
<td>PV_PTCD</td>
<td>GSI_PAR_PrevPointCode</td>
</tr>
<tr>
<td>ACT_PTID</td>
<td>GSI_PAR_PrevPointId</td>
</tr>
<tr>
<td>BACKID</td>
<td>GSI_PAR_PrevPointId</td>
</tr>
<tr>
<td>PTNR</td>
<td>GSI_PAR_RunningPointId</td>
</tr>
<tr>
<td>FNR</td>
<td>GSI_PAR_SerialNr (undefined)</td>
</tr>
<tr>
<td>TYPE</td>
<td>GSI_PAR_InstrType (undefined)</td>
</tr>
<tr>
<td>TIME_1</td>
<td>GSI_PAR_Date</td>
</tr>
<tr>
<td>TIME_2</td>
<td>GSI_PAR_Time</td>
</tr>
<tr>
<td>HZ</td>
<td>GSI_PAR_AngleHz</td>
</tr>
<tr>
<td>V</td>
<td>GSI_PAR_AngleV</td>
</tr>
<tr>
<td>SLOPE</td>
<td>GSI_PAR_DistSlope</td>
</tr>
<tr>
<td>HOR</td>
<td>GSI_PAR_DistHoriz</td>
</tr>
<tr>
<td>HGT</td>
<td>GSI_PAR_ElevDiff</td>
</tr>
<tr>
<td>PPMM</td>
<td>GSI_PAR_PpmMm</td>
</tr>
<tr>
<td>SIGMA</td>
<td>GSI_PAR_NS</td>
</tr>
</tbody>
</table>

Meas
<table>
<thead>
<tr>
<th>Meas</th>
<th>GSI_ID_MM</th>
<th>GSI_PAR_AddConst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meas</td>
<td>GSI_ID_PPM</td>
<td>GSI_PAR_PpmTotal</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_1</td>
<td>GSI_PAR_Info1</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_2</td>
<td>GSI_PAR_Info2</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_3</td>
<td>GSI_PAR_Info3</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_4</td>
<td>GSI_PAR_Info4</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_5</td>
<td>GSI_PAR_Info5</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_6</td>
<td>GSI_PAR_Info6</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_7</td>
<td>GSI_PAR_Info7</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_8</td>
<td>GSI_PAR_Info8</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_9</td>
<td>GSI_PAR_Info9</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_E</td>
<td>GSI_PAR_TargetEast</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_N</td>
<td>GSI_PAR_TargetNorth</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_H</td>
<td>GSI_PAR_TargetElev</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_E0</td>
<td>GSI_PAR_StationEast</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_N0</td>
<td>GSI_PAR_StationNorth</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_H0</td>
<td>GSI_PAR_StationElev</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HR</td>
<td>GSI_PAR_ReflHeight</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HI</td>
<td>GSI_PAR_InstrHeight</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE</td>
<td>GSI_PAR_Attrib1</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_1</td>
<td>GSI_PAR_Attrib2</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_2</td>
<td>GSI_PAR_Attrib3</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_3</td>
<td>GSI_PAR_Attrib4</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_4</td>
<td>GSI_PAR_Attrib5</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_5</td>
<td>GSI_PAR_Attrib6</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_6</td>
<td>GSI_PAR_Attrib7</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_7</td>
<td>GSI_PAR_Attrib8</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_8</td>
<td>GSI_PAR_Attrib9</td>
</tr>
</tbody>
</table>

GSI_ID_APPDATA0 are for the purpose of exchanging data between applications and between application and MDlg. They cannot be recorded. Both can be of the form GSI_ASCII or GSI_DOUBLE.
### Special Ids

<table>
<thead>
<tr>
<th>GSI_ID_NONE</th>
<th>GSI_PAR_NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_EMPTY</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_UNKNOWN</td>
<td></td>
</tr>
</tbody>
</table>

The set of GSI_PAR-ids is not complete in this table. There exist several more Ids, which can be used for displaying.

### 6.4.5 Data Structures for GSI Functions

**GSI_Widg_Entry_Type: Dialog entry information**

*Description*  
This data structure is used to store information about the entries (data fields) of the WI dialog.

```plaintext
TYPE GSI_Widg_Entry_Type  
iId AS Integer  
```

The identifier of the dialog entry. For possible value see WI constants.
iDataType AS Integer
The type of the date stored in dValue or sValue. For possible value see table below.

<table>
<thead>
<tr>
<th>Meaning</th>
<th>AS iDataType</th>
<th>Meaning</th>
<th>AS iDataType</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII data (stored in sValue)</td>
<td>GSI_ASCII</td>
<td>signed ASCII data (stored in sValue)</td>
<td>GSI_ASCII_SIGN</td>
</tr>
<tr>
<td>double data (stored in dValue)</td>
<td>GSI_DOUBLE</td>
<td>double data (stored in dValue)</td>
<td>GSI_DOUBLE</td>
</tr>
</tbody>
</table>

lValid AS Logical
TRUE if the value is valid.

dValue AS Double
Data if value is of type Double.

sValue AS String18
Data if value is of type String.

END GSI_WiDlg_Entry_Type

Wi_List: An array of GSI_WiDlg_Entry_Type
Description
This array consists of GSI_MAX_REC_WI elements of the type GSI_WiDlg_Entry_Type.

GSI_Rec_Id_List: An array of integers (indicating WI-identifications)
Description
This array consists of GSI_MAX_REC_WI elements of the type Integer. It is used to define the recorded values (recmask).

GSI_Point_Coord_Type: Point co-ordinate data
Description
This data structure is used to store a point name and its co-ordinates.

TYPE GSI_Point_Coord_Type
sPtNr AS String18
point number
dEast AS Double
east co-ordinate
dNorth AS Double
north co-ordinate
dHeight AS Double
height co-ordinate
lPtNrValid AS Logical
TRUE if point number is valid
lEValid AS Logical
TRUE if east co-ordinate is valid
lNValid AS Logical
TRUE if north co-ordinate is valid
6.4.6 GSI_GetRunningNr

**Description**
Fetches the running point number and the increment.

**Declaration**

```basic
GSI_GetRunningNr( sPntId AS String20, sPntIncr AS String20 )
```

**Remarks**
Fetches the running point number and increment for it.

**Parameters**

- `sPntId` out the running point number
- `sPntIncr` out the increment for the running point number

**See Also**
GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr, GSI_IsRunningNr

**Return-Codes**

- `RC_OK` successful

**Example**

```basic
DIM sPntId AS String20
DIM sPntInc AS String20

GSI_GetRunningNr( sPntId, sPntInc )
```
6.4.7 GSI_SetRunningNr

**Description**
Sets the running point number and increment.

**Declaration**

```vba
GSI_SetRunningNr(
    BYVAL sPntId AS String20,
    BYVAL sPntIncr AS String20
)
```

**Remarks**
Sets the running point number and the increment for it. The running point number mode is switched on.

**Parameters**

- `sPntId` in: The user running point number.
- `sPntIncr` in: The increment for the user point running number.

**See Also**

- GSI_GetRunningNr, GSI_GetIndivNr,
- GSI_SetIndivNr, GSI_IsRunningNr

**Return-Codes**

- RC_OK: successful

**Example**

```vba
DIM sPntId AS String20
DIM sPntInc AS String20
GSI_SetRunningNr( sPntId, sPntInc )
```

6.4.8 GSI_GetIndivNr

**Description**
Fetched the individual point number.

**Declaration**

```vba
GSI_GetIndivNr( sPntId AS String20 )
```

**Remarks**
Fetched the individual point number.

**Parameters**

- `sPntId` out: The user-defined individual point number.

**See Also**

- GSI_GetRunningNr, GSI_SetRunningNr,
- GSI_SetIndivNr, GSI_IsRunningNr
6.4.9 GSI_SetIndivNr

Description: Sets the individual point number.

Declaration: GSI_SetIndivNr( BYVAL sPntId AS String20 )

Remarks: Sets the individual point number. After this call, the running point number mode is switched to the individual point number. This mode will be active until replaced by a running number or until the next save.

Parameters:
- sPntId in The user-defined individual point number.

See Also:
- GSI_GetRunningNr, GSI_SetRunningNr,
- GSI_GetIndivNr, GSI_IsRunningNr

Return-Codes
- RC_OK successful

Example:
- DIM sPntId AS String20
- GSI_SetIndivNr( sPntId )

6.4.10 GSI_IsRunningNr

Description: Queries if running number is being used.

Declaration: GSI_IsRunningNr( lRunningOn AS Logical )

Remarks: If the running number is active the parameter will forced to TRUE otherwise to FALSE.
Parameters

\[ l\text{RunningOn} \text{ out} \text{ information about the running point number} \]

See Also

GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr

Return-Codes

RC_OK successful

Example

\[
\text{DIM } l\text{RunningOn AS Logical} \\
\text{GSI_IsRunningNr( } l\text{RunningOn )}
\]

6.4.11 GSI_SetIvPtNrStatus

Description
Switches the individual point number mode on/off.

Declaration

\[
\text{GSI_SetIvPtNrStatus(} \\
\text{BYVAL } l\text{Switch AS Logical })
\]

Remarks
Switch the individual point number on or off. When point number is shown in the display the number will change.

Parameters

\[ l\text{Switch} \text{ in} \text{ switch for the individual point-number} \]  
\(\text{(TRUE = on, FALSE = off)}\)

See Also

GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr, GSI_IsRunningNr

Return-Codes

RC_OK successful

Example

\[
\text{GSI_SetIvPtNrStatus( } \text{FALSE )}
\]
6.4.12 GSI_IncPNumber

**Description**  
Automatically point number increment.

**Declaration**  
GSI_IncPNumber()

**Remarks**  
This function increments the running alphanumeric point number.

**Parameters**  
none

**See Also**  
GSI_GetRunningNr, GSI_SetRunningNr,  
GSI_GetIndivNr, GSI_SetIndivNr

**Return Codes**  
RC_IVRESULT  
Point number is not incremented, possible reasons could be:  
wrong alphanumeric chars in point number  
alphanumeric chars in step  
overflow on a alphanumeric char  
step is longer as the point number

**Example**  
GSI_IncPNumber()

6.4.13 GSI_Coding

**Description**  
Starts the active coding function of the TPS system.

**Declaration**  
GSI_Coding( BYVAL Caption AS _Token)

**Remarks**  
This routine starts the active coding function of the TPS system. Since there exist three possible locations, the TPS system follows a default ordering rule to invoke one of the programs. First it checks if there is an appropriate set up GeoBASIC coding program. If yes it will be executed, otherwise it examines the codelist management if a codelist is selected. If yes then the codelist will be opened, otherwise the standard coding will be activated.

**Parameters**  
Caption  
The left caption string of the dialog.
6. System Functions

Return-Codes

RC_OK successful
LDR_ GeoBASIC is already running
RECURSIV_ERR

Example

The example uses the GSI_Coding routine to open a dialog for coding.

GSI_Coding( "CODE" )

6.4.14 GSI_SelectCode

Description
This routine shows the codelist-coding dialog

Declaration
GSI_SelectCode( BYVAL Caption AS _Token)

Remarks
This routine starts the codelist-coding function of the TPS system. It will be executed only if a valid codelist is selected.

Parameters

Caption in The left caption string of the dialog.

Return-Codes

RC_OK successful
RC_ABORT Coding was aborted by pressing of the ESC-button
RC_ABORT_APPL Coding was aborted by pressing of the QUIT-button
COD_RC_LIST_ NOT_VALID No valid codelist selected

Example
See example file „meas.gbs“.

6.4.15 GSI_GetQCodeAvailable

Description
This routine returns the status for Quick-Coding.

Declaration
GSI_GetQCodeAvailable(lAvailable As Logical, lEnabled As Logical)
6.4.16 GSI_SetQCodeMode

**Description**
Sets the Quick-Coding mode.

**Declaration**
GSI_SetQCodeMode(BYVAL lEnabled As Logical)

**Remarks**
This routine enables or disables the Quick-Coding. It can be only activated if a valid codelist is selected (see GSI_GetQCodeAvailable).

**Parameters**
- lEnabled in TRUE: enable Quick-Coding

**See Also**
GSI_GetQCodeAvailable, GSI_ExecQCoding

**Return-Codes**
- RC_OK successful

**Example**
See example file „meas_od.gbs“.

6.4.17 GSI_ExecQCoding

**Description**
Executes the Quick-Coding.
Declaration

GSI_ExecQCoding(
    BYVAL lRecEnable AS Logical
    iButtonId AS Integer,
    lNewCode AS Logical)

Remarks

This routine executes the Quick-Coding. If Quick-Coding is enabled, it checks the button iButtonId and searches the corresponding code. If the selected code needs mandatory attributes, it shows the coding dialog. As successful coding is indicated by lNewCode=TRUE. The results are stored in the Theodolite data pool (see GSI_GetWiEntry).

If lRecEnable=TRUE, this routine executes the ALL-button functionality too, it measures a distance and records the results. The recording order (measurement block – code block or vice versa) depends on the system setting (see GSI_GetRecOrder).

If lRecEnable=FALSE, this routine forces no new distance measurement and there is no recording.

Parameters

lRecEnable in

TRUE: Quick-Coding including distance measurement. It records a code- and a measurement-block in the correct order.
FALSE: Quick-Coding without measurement and without recording

iButtonId inout

In: Pressed button.
Out: If a Quick-Coding was possible, iButtonId is changed to MMI_NO_KEY, otherwise it is unchanged

lNewCode out

TRUE: Quick-Coding was successful

See Also

GSI_GetQCodeAvailable, GSI_SetQCodeMode, GSI_SetRecOrder

Return-Codes

RC_OK successful

Example

See example files „meas.gbs” and „meas_od.gbs“.
6.4.18 GSI_SetRecOrder

**Description**
Sets the recording order for Quick-Coding.

**Declaration**
GSI_SetRecOrder(BYVAL lCodeFirst As Logical)

**Remarks**
This routine defines the recording order for Quick-Coding.

If lCodeFirst=TRUE, then the code-block will be recorded before the measurement block.

**Parameters**

lCodeFirst in TRUE: code-block before measurement block

**See Also**
GSI_GetRecOrder, GSI_ExecQCoding

**Return-Codes**
RC_OK successful

**Example**
See example file „meas_od.gbs“.

6.4.19 GSI_GetRecOrder

**Description**
Returns the recording order for Quick-Coding.

**Declaration**
GSI_GetRecOrder(lCodeFirst As Logical)

**Remarks**
This routine returns the recording order for Quick-Coding.

If lCodeFirst=TRUE, then the code-block will be recorded before the measurement block.

**Parameters**

lCodeFirst out TRUE: code-block before measurement block

**See Also**
GSI_SetRecOrder, GSI_ExecQCoding

**Return-Codes**
RC_OK successful

**Example**
See example file „meas_od.gbs“.
6.4.20  GSI_QuickSet

**Description**  Shows the Quickset dialog.

**Declaration**  
```plaintext
GSI_QuickSet(BYVAL sCaptionLeft AS _Token)
```

**Remarks**  This procedure shows Quickset for station setting.

**Parameters**  
- `sCaptionLeft`  in  Left caption for the Quickset dialog

**Return-Codes**
- `RC_OK`  Successful termination.

**Example**  Show the dialog:

```plaintext
GSI_QuickSet ( "BASIC" )
```

6.4.21  GSI_SetRecPath

**Description**  Defines the recording path for the measurements.

**Declaration**  
```plaintext
GSI_SetRecPath(
    BYVAL iPathInfo AS Integer,  
    BYVAL sFileName AS FileName,  
    BYVAL sFilePath AS FilePath )
```

**Remarks**  This procedure defines where the measurements will be recorded. 
If `iPathInfo` is set to `GSI_INTERFACE`, then the measurements will be sent to the RS232 line and the other parameters are not be interpreted. If `iPathInfo` is set to `GSI_EXTERNAL`, then `sFileName` defines the filename i.e. "MeasJob.GSI" and `sFilePath` defines the file-path, i.e. "A:\\GSI".

**Parameters**  
- `iPathInfo`  in  Defines where the data are recorded
- `sFileName`  in  Valid Filename (8+3 format)
- `sFilePath`  in  file-path
6. System Functions

Return-Codes

RC_OK Successful termination.

See Also  GSI_GetRecPath

Example  This example shows the actual recording path and set it to the RS232 line:

```
DIM sFile As FileName
DIM sPath As FilePath
DIM iPathInfo As Integer

GSI_GetRecPath(iPathInfo, sFile, sPath)
IF iPathInfo = GSI_EXTERNAL THEN
    MMI_PrintStr(0, 1, "RecFile-CARD: " + sFile, TRUE)
    MMI_PrintStr(0, 2, " Path: " + sPath, TRUE)
ELSE
    MMI_PrintStr(0, 1, "RecPath - serial line", TRUE)
END IF
GSI_SetRecPath( GSI_INTERFACE, sFile, sPath)
```

6.4.22  GSI_GetRecPath

Description  Returns the recording path for the measurements.

Declaration  GSI_GetRecPath(  
    iParamInfo AS Integer,  
    sFileName AS FileName,  
    sFilePath AS FilePath )

Remarks  This procedure returns where the measurements will be recorded. If iParamInfo = GSI_INTERFACE, then the measurements will be sent to the RS232 line and the other parameters are not valid. If iParamInfo = GSI_EXTERNAL, then sFileName defines the filename i.e. "MeasJob.GSI" and sFilePath defines the filepath, i.e. "A: \GSI".
Parameters

iPathInfo out Device info
sFileName out Filename (8+3 format)
sFilePath out File-path

Return-Codes

RC_OK Successful termination.

See Also

GSI_SetRecPath

Example

see GSI_SetRecPath

6.4.23 GSI_SetDataPath

Description

Set the file with the import data.

Declaration

GSI_SetDataPath(
    BYVAL iPathInfo AS Integer,
    BYVAL sFileName AS FileName,
    BYVAL sFilePath AS FilePath
)

Remarks

This procedure sets the file from which data will be imported. Only GSI_EXTERNAL is valid for the iPathInfo. sFileName defines the filename i.e. "DataJob.GSI" and sFilePath defines the file-path, i.e. "A:\GSI".

Parameters

iPathInfo in Device info (Only GSI_EXTERNAL is valid)
sFileName in Valid Filename (8+3 format)
sFilePath in File-path

Return-Codes

RC_OK Successful termination.

See Also

GSI_GetDataPath

Example

The example defines the file “A:\GSI\DataJob.GSI” as new import file.
GSI_SetDataPath(GSI_EXTERNAL, "DataJob.GSI", "A:\\GSI")

6.4.24 GSI_GetDataPath

Description Get the name of the file with the import data.

Declaration

GSI_GetDataPath(
    iPathInfo AS Integer,
    sFileName AS FileName,
    sFilePath AS FilePath
)

Remarks This procedure fetches the name and the path of the file from which data will be imported. If iPathInfo = GSI_EXTERNAL, then sFileName defines the filename i.e. "DataJob.GSI" and sFilePath defines the file-path, i.e. "A:\\GSI".

Parameters

- iPathInfo  out  Device info
- sFileName  out  Filename (8+3 format)
- sFilePath  out  File-path

Return-Codes

- RC_OK  Successful termination.

See Also GSI_SetDataPath

Example The example fetches the name and the path of the standard import data file:

```basic
DIM iPathInfo AS Integer
DIM sFileName AS FileName
DIM sFilePath AS FilePath
GSI_GetDataPath(iPathInfo, sFileName, sFilePath)
```

6.4.25 GSI_GetWiEntryText

Description Get coding text-data from the Theodolite data pool.
Declaration

\[
\text{GSI\_GetWiEntryText(}
\begin{align*}
\text{WiIdentification AS Integer,} \\
\text{WiEntryText AS String30}
\end{align*}
\)

Remarks

This routine is used to fetch coding descriptions from the Theodolite data pool, i.e. the code-description itself or the description text of the attributes. If no codelist is selected, then the standard prompts will be returned.

Texts for the following wi’s can be fetched:
- GSI\_ID\_PTCD\_DSC,
- GSI\_ID\_REM\_1, GSI\_ID\_REM\_2, GSI\_ID\_REM\_3, GSI\_ID\_REM\_4,
- GSI\_ID\_REM\_5, GSI\_ID\_REM\_6, GSI\_ID\_REM\_7, GSI\_ID\_REM\_8,
- GSI\_ID\_REM\_9, GSI\_ID\_CD\_DSC, GSI\_ID\_CODE,
- GSI\_ID\_CODE\_1, GSI\_ID\_CODE\_2, GSI\_ID\_CODE\_3,
- GSI\_ID\_CODE\_4, GSI\_ID\_CODE\_5, GSI\_ID\_CODE\_6,
- GSI\_ID\_CODE\_7, GSI\_ID\_CODE\_8.

Parameters

- WiIdentification in The identification of the Wi.
- WiEntryText out Entry-Text.

See Also

-

Example

This example gets the description-text and the value of the first coding attribute and send it out over the serial line.

\[
\text{GSI\_GetWiEntryText( GSI\_ID\_CODE\_1, sWiEntryText )}
\]

\[
\text{GSI\_GetWiEntry( GSI\_ID\_CODE\_1, WiEntry )}
\]

\[
\text{send(\"Info1:\" } + \text{sWiEntryText}
\]

\[
\text{+\":\"} + \text{WiEntry.sValue)}
\]

6.4.26 GSI\_GetWiEntry

Description

Get data from the Theodolite data pool.

Declaration

\[
\text{GSI\_GetWiEntry(}
\begin{align*}
\text{WiIdentification AS Integer,} \\
\text{WiEntry AS GSI\_WiDlg\_Entry\_Type}
\end{align*}
\)

Remarks

This routine is used to fetch data from the Theodolite data pool. All existing wi’s can be fetched (see the description of the Wi constants for possible values).
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiIdentification</td>
<td>in/out</td>
<td>The identification of the WI.</td>
</tr>
<tr>
<td>WiEntry</td>
<td>in</td>
<td>The WI entry data. See the description of GSI_WiDlg_Entry_Type for further information.</td>
</tr>
</tbody>
</table>

#### See Also

GSI_SetWiEntry

#### Example

See example GSI_SetWiEntry.

### 6.4.27 GSI_SetWiEntry

**Description**

Put data to the Theodolite data pool.

**Declaration**

```basic
GSI_SetWiEntry(
    WiIdentification AS Integer,
    WiEntry AS GSI_WiDlg_Entry_Type
)
```

**Remarks**

This routine is used to put data to the Theodolite data pool. See the description of the WI constants.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiIdentification</td>
<td>in</td>
<td>The identification of the WI.</td>
</tr>
<tr>
<td>WiEntry</td>
<td>in</td>
<td>The WI entry data. See the description of GSI_WiDlg_Entry_Type for further information.</td>
</tr>
</tbody>
</table>

#### See Also

GSI_GetWiEntry

#### Example

GSI_SetWiEntry does not set WI.iId according to the first parameter, instead it will just use the value stored in WI.iId. If that value is unequal to the first parameter value, then it comes to a conflict. Use a GSI_GetWiEntry() first, to be sure that all values of the GSI_WiDlg_Entry_Type are initialized correctly. See also the example for the definition of a measurement dialog.

Save way:

```basic
GSI_GetWiEntry ( GSI_ID_HR, Wi )
Wi.lValid = TRUE
Wi.dValue = 2.12
GSI_SetWiEntry ( GSI_ID_HR, Wi )
```
6.4.28 GSI_GetRecMask

Description
Get the definition and the format of a recording mask.

Declaration
GSI_GetRecMask(
    BYVAL iMaskNr AS Integer,
    sMaskName AS String18,
    RecWiMask AS GSI_Rec_Id_List,
    iRecFormat AS Integer,
    lEditMask AS Logical
)

Remarks
This routine fetches the definition and the format of the recording mask with the number iMaskNr. Valid formats are GSI_RECFORMAT_GSI and GSI_RECFORMAT_GSI16. A recording mask can be set with GSI_SetRecMask. If lEditMask is TRUE the elements of the recording mask can be changed in GSI_DefineRecMaskDlg. All unused elements of the recording list are set to GSI_ID_NONE. All values from 0 to 5 are valid for the mask number. Mask number 0 is predefined for the station recording mask.

Note
Only the first 16 characters of sMaskName are valid.

Parameters
iMaskNr in Number of the recording mask.
    GSI_ACTUAL_RECMASK can be used to retrieve settings of the actual mask
sMaskName out Name of the recording mask
RecWiMask out The definition of the recording mask. The elements of the array are the identification numbers of the WI’s. See the description of the WI constants.
iRecFormat out Recording format
    (GSI_RECFORMAT_GSI, GSI_RECFORMAT_GSI16)
lEditMask out Mask editable flag

See Also
GSI_SetRecMask, GSI_DefineRecMaskDlg

Example
The example uses the GSI_GetRecMask routine to fetch the definition and the format of the recording mask number 2.
6.4.29 GSI_SetRecMask

**Description** Set the definition and the format of a recording mask.

**Declaration**
```
GSI_SetRecMask(    
    BYVAL iMaskNr AS Integer,    
    BYVAL sMaskName AS String18,    
    BYVAL RecWiMask AS GSI_Rec_Id_List,    
    BYVAL iRecFormat AS Integer,    
    BYVAL lEditMask AS Logical)
```

**Remarks** This routine sets the definition and the format of the recording mask with the number iMaskNr. Valid formats are GSI_RECFORMAT_GSI and GSI_RECFORMAT_GSI16. If lEditMask is TRUE the elements of the recording mask can be changed in GSI_DefineRecMaskDlg. All unused elements should be set to GSI_ID_NONE. All values from 0 to 5 are valid for the mask number. Mask number 0 is predefined for the station recording mask.

**Note**
1) WiEntries must be unique, hence may not appear doubly.
2) Only GSI_MAX_REC_WI number of entries may be defined.
3) Only the first 16 characters of sMaskName are valid.

**Parameters**
- **iMaskNr** in Number of the recording mask. GSI_ACTUAL_RECMASK can be used to set the values of the currently active
6. System Functions

6.4.30 GSI_SetRecMaskNr

Description
Set the used recording mask.

Declaration
GSI_SetRecMaskNr(BYVAL iMaskNr AS Integer)

Parameters
iMaskNr in Number of the recording mask. Number must be in the range 1.. GSI_MAX_REC_Masks.

See Also
GSI_GetRecMaskNr
Example

The example sets the next recording mask.

```basic
DIM i AS Integer
GSI_GetRecMaskNr( i )
i = i + 1 ' take next mask
i = ((i - 1) MOD GSI_MAX_REC_MASKS) + 1
GSI_SetRecMaskNr( i )
```

6.4.31 GSI_GetRecMaskNr

**Description**

Returns the used recording mask.

**Declaration**

```basic
GSI_GetRecMaskNr(iMaskNr AS Integer)
```

**Parameters**

- `iMaskNr` out Number of the recording mask.

**See Also**

GSI_SetRecMaskNr

6.4.32 GSI_DefineRecMaskDlg

**Description**

Defines the recording mask dialog.

**Declaration**

```basic
GSI_DefineRecMaskDlg()
```

**Remarks**

Defines the contents of the recording mask. Using a dialog with list-fields, the user can select the items for the user registration mask. This routine is an interactive equivalent to the routines GSI_GetRecMask and GSI_SetRecMask.

**See Also**

GSI_GetRecMask, GSI_SetRecMask,

**Example**

```basic
GSI_DefineRecMaskDlg ()
```

6.4.33 GSI_ManCoordDlg

**Description**

Show the manual co-ordinate input dialog.
Declaration  
GSI_ManCoordDlg( 
    BYVAL sCaption AS _Token, 
    BYVAL iPointType AS Integer, 
    Point AS GSI_Point_Coord_Type, 
    BYVAL iFlags AS Integer, 
    BYVAL sHelpText AS _Token )

Remarks  
This routine shows the manual co-ordinates input dialog and allows editing, coding and recording. The type of co-ordinates (station or target) can be selected using iPointType. Recording to the current data-file (defined in GSI_ImportCoordDlg) with REC or leaving this function with CONT is only possible if the point number is valid, and at least E- and N-co-ordinates are valid. If GSI_HEIGHT_MUST is included in iFlags the Height / Elevation-co-ordinate must be valid too. Leaving using ESC or QUIT (Shift-F6) is always possible. Recording and coding sets the according values in the Theodolite data-pool too.

Parameters  
sCaption in  The maximal five-character long left part of the title bar.
iPointType in  station or target point. For the values for PointType see table below

Point Type | Meaning
--- | ---
GSI_STATION | station point number
GSI_INDIV_TG | individual target number
GSI_RUN_TG | running target
GSI_BACKSIGHT | backside number (analog target, only changed prompts)
### GSI_POINT_CODE

**PointId / CodeId**  
(analog target, only changed prompts)

<table>
<thead>
<tr>
<th>Input/Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point in</td>
<td>only point number, co-ordinates will be set to 0</td>
</tr>
<tr>
<td>Point out</td>
<td>point number and -co-ordinates. For further information see the description of GSI_Point_Coord_Type</td>
</tr>
<tr>
<td>iFlags in</td>
<td>defines functionality</td>
</tr>
</tbody>
</table>

#### Valid Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ALLOW_REC</td>
<td>allows recording and coding</td>
</tr>
<tr>
<td>GSI_HEIGHT_MUST</td>
<td>height must be entered</td>
</tr>
<tr>
<td>GSI_NE_OPTIONAL</td>
<td>only height must be entered, north &amp; east are optional</td>
</tr>
<tr>
<td>GSI_MULTI_REC</td>
<td>Allows entering and recording of more than one data-set, without leaving this routine</td>
</tr>
<tr>
<td>GSI_NO_FILE_CHANGE</td>
<td>File changing is disabled</td>
</tr>
</tbody>
</table>

Flags can be combined with ‘+-’-operator (iFlags = iFlag1 + iFlag2)

#### sHelpText in

This text is shown, when the help button SHIFT-F1 is pressed and the help functionality of the theodolite is enabled.

### See Also

GSI_ImportCoordDlg

### Example
DIM Point AS GSI_Point_Coord_Type
GSI_ManCoordDlg ("TEST", GSI_STATION, Point,
    GSI_HEIGHT_MUST+GSI_ALLOW_REC,
    "This is the Helpext" )

6.4.34 GSI_ImportCoordDlg

**Description**  
Show the co-ordinate import dialog.

**Declaration**  
GSI_ImportCoordDlg(
    BYVAL sCaption AS _Token,
    BYVAL iPointType AS Integer,
    Point AS GSI_Point_Coord_Type,
    BYVAL iFlags AS Integer,
    BYVAL iImportFile AS Integer,
    BYVAL sImportHelp AS _Token,
    BYVAL sInputHelp AS _Token,
    BYVAL sF2Button AS _Token,
    BYVAL sF4Button AS _Token
)

**Remarks**  
This routine contains three dialogues, the search-, the view- and the manual-input dialog. The type of co-ordinates (station or target) can be selected using `iPointType`. The search dialog allows selecting the data- or the measure file and editing a point-number. Depending on the pressed button, the manual co-ordinate input function (only if `GSI_ALLOW_MAN` is included in `iFlags`, see `GSI_ManCoordDlg`) or the view-co-ordinates dialog will be called.

The start of searching is always at the top of the file. With the two search keys, the user can step from one valid point to the next in both directions.

**Rules for a valid point:**
- point number found
- E- and N-coordinates (target or station) exists and are valid
- if `GSI_HEIGHT_MUST` is included in `iFlags`, a valid
height / elevation-coordinate must exist to within the file too.

If no valid point exists or no more valid points are in the desired search direction, a warning message will be displayed.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sCaption</td>
<td>in</td>
<td>The maximal five-character long left part of the title bar.</td>
</tr>
<tr>
<td>iPointType</td>
<td>in</td>
<td>station or target point. For the values for PointType see table below</td>
</tr>
<tr>
<td>Point</td>
<td>in</td>
<td>Only point number, the co-ordinates will be set to 0.</td>
</tr>
<tr>
<td>Point</td>
<td>out</td>
<td>point number and co-ordinates. For further information see the description of GSI_Point_Coord_Type.</td>
</tr>
<tr>
<td>iFlags</td>
<td>in</td>
<td>defines functionality</td>
</tr>
</tbody>
</table>

#### Valid Flags

<table>
<thead>
<tr>
<th>Valid Flags</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ALLOW_REC</td>
<td>allows recording and coding</td>
</tr>
<tr>
<td>GSI_MULTI_REC</td>
<td>Allows multiple manual coord. entering</td>
</tr>
</tbody>
</table>

#### Point Type

<table>
<thead>
<tr>
<th>Point Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_STATION</td>
<td>station point number</td>
</tr>
<tr>
<td>GSI_INDIV_TG</td>
<td>individual target number</td>
</tr>
<tr>
<td>GSI_RUN_TG</td>
<td>running target</td>
</tr>
<tr>
<td>GSI_BACKSIGHT</td>
<td>backside number (analog target, only changed prompts)</td>
</tr>
<tr>
<td>GSI_POINT_CODE</td>
<td>PointId / CodecId (analog target, only changed prompts)</td>
</tr>
</tbody>
</table>
GSI_ALLOW_MAN allows manual coord. entering
GSI_HEIGHT_MUST height must be entered
GSI_DIRECT_SEARCH direct searching without dialog
GSI_NO_VIEW no coord view if found
GSI_NE_OPTIONAL only height must be entered, north & east are optional
GSI_SEARCH_FROM_END Starts searching from end of file
GSI_NO_FILE_CHANGE Changing of file is disabled
GSI_GET_NEXT Return the next valid data-set, ignore sPtNr

Flags can be combined with ‘+’-operator (iFlags = iFlag1 + iFlag2)
iImportFile in defines the source file for importing

**Valid Import File**
- **GSI_FILE_MEAS** MEAS file
- **GSI_FILE_DATA** DATA file
- **GSI_FILE_LAST** last used file

sImportHelp in Help text for import dialog. Only visible if the help functionality of the theodolite is enabled.
sInputHelp in Help text for manual input dialog. Only visible if the help functionality of the theodolite is enabled.
sF2Button in Text for activating F2 button.
sF4Button in Text for activating F4 button

**See Also** GSI_ManCoordDlg
Example

```basic
DIM Point AS GSI_Point_Coord_Type
GSI_ImportCoordDlg( "IMP", GSI_INDIV_TG, Point, GSI_ALLOW_REC + GSI_ALLOW_MAN, GSI_FILE_DATA, "Import Help Text", "Input Help Text", "F2", "F4" )
```

### 6.4.35 GSI_SetLineSysMDlg

**Description**
Sets a line in the system measurement dialog.

**Declaration**

```basic
GSI_SetLineSysMDlg( 
  BYVAL iDlgNr AS Integer 
  BYVAL iLineNr AS Integer 
  BYVAL iSysParamId AS Integer )
```

**Remarks**
This routine sets one line in the system measurement dialog. To fetch information about a line, `GSI_GetLineSysMDlg` can be used. Unused lines should be set to `GSI_PAR_NONE`.

1) Parameters are identified by `GSI_PAR_*` values and not by `GSI_ID_*` values.
2) A line in the system measurement dialog can only be set to a system parameter not to an application parameter.

**Parameters**

- **iDlgNr** in
  The number of the system measurement dialog where the line should be set. Possible values are:
  - Value | Meaning
    - GSI_SYS_MDLG_1 | Dialog 1
    - GSI_SYS_MDLG_2 | Dialog 2
    - GSI_SYS_MDLG_3 | Dialog 3

- **iLineNr** in
  The number of the line to set.
  Valid numbers:
  - GSI_MAX_DLG_LINES

- **iSysParamId** in
  Identification of the system parameter. Refer to the chapter
“Constants for Measurement Dialog Definition”

See Also
- GSI_GetLineSysMDlg
- GSI_DefineMDlg

Example
See sample program “meas.gbs”.
This example uses GSI_SetLineSysMDlg to configure the first two lines of the first system measurement dialog.

GSI_SetLineSysMDlg( GSI_SYS_MDLG_1, 1, GSI_PAR_Date )
GSI_SetLineSysMDlg( GSI_SYS_MDLG_1, 2, GSI_PAR_Time )

6.4.36 GSI_GetLineSysMDlg

Description
Gets the definition of a line in the system measurement dialog.

Declaration
GSI_GetLineSysMDlg( 
BYVAL iDlgNr AS Integer 
BYVAL iLineNr AS Integer 
iSysParamId AS Integer )

Remarks
This routine fetches the information about the setting of one line in the system measurement dialog. To set a line in the system measurement dialog the routine GSI_SetLineMDlg can be used.
Parameters

iDlgNr in
The number of the system measurement dialog where the line should be fetched. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_SYS_MDLG_1</td>
<td>Dialog 1</td>
</tr>
<tr>
<td>GSI_SYS_MDLG_2</td>
<td>Dialog 2</td>
</tr>
<tr>
<td>GSI_SYS_MDLG_3</td>
<td>Dialog 3</td>
</tr>
</tbody>
</table>

iLineNr in
The number of the line to fetch.

iSysParamId out
Identification of the system parameter. Refer to the chapter “Constants for Measurement Dialog Definition”

See Also
GSI_SetLineSysMDlg
GSI_DefineMDlg

Example
See sample program “meas.gbs”.
This example uses GSI_GetLineSysMDlg to get information about the configuration of the first system measurement dialog’s first two lines.

```basic
DIM iParLine1 AS Integer
DIM iParLine2 AS Integer

GSI_GetLineSysMDlg( GSI_SYS_MDLG_1, 1, iParLine1)
GSI_GetLineSysMDlg( GSI_SYS_MDLG_1, 2, iParLine2)
```

6.4.37 GSI_SetMDlgNr

Description
Sets the number of the system measurement dialog.

Declaration
GSI_SetMDlgNr( BYVAL iDlgNr AS Integer)

Remarks
Sets the number of the system measurement dialog. The content of these dialogs can be changed by using GSI_DefineMDlg.
6.4.38 GSI_GetMDlgNr

Description
Returns the number of the system measurement dialog.

Declaration
GSI_GetMDlgNr(iMDlgNr AS Integer)

Remarks
Returns the number of the system measurement dialog.

Parameters

See Also
GSI_SetMDlgNr

Example
See sample program “meas_od.gbs”. This example sets the next dialog mask
GSI_GetMDlgNr(i)
i = (i + 1) MOD GSI_MAX_MDLG_MASKS
GSI_SetMDlgNr(i)

6.4.39 GSI_CreateMDlg

Description
Create and show the user definable measurement dialog.

Declaration
GSI_CreateMDlg(
    BYVAL iFixLines AS Integer
    BYVAL sCaptionLeft AS _Token
    BYVAL sCaptionRight AS _Token
    BYVAL sHelpText AS _Token
)

Remarks
This routine creates and shows the user definable measurement dialog with iFixLines fix lines, the left part of the title bar sCaptionLeft, the caption sCaptionRight, and the help text sHelpText.
Only one measurement dialog can exist at the same time. If GSI_CreateMDlg is called and there already exists a measurement dialog, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

| Note | If a graphics dialog or a text dialog exist together with a measurement dialog, all button routines (MMI_AddButton, MMI_GetButton, MMI_DeleteButton) are related to the measurement dialog. |

The shown parameters used in the dialog are defined in the user display mask (see GSI_DefineMDlg).

### Parameters

- **iFixLines** in
  The number of fix lines. (These lines are not scrolled.)

- **sCaptionLeft** in
  The part of the title bar displayed on the left border (up to five characters wide)

- **sCaptionRight** in
  The caption of the dialog.

- **sHelpText** in
  This text is shown, when the help button `SHIFT-F1` is pressed and the help functionality of the theodolite is enabled.

### See Also

- GSI_UpdateMDlg
- GSI_UpdateMeasurement

### Example

See example file „meas.gbs“ too.

This example uses the measure dialog routines GSI_CreateMDlg, GSI_UpdateMDlg and GSI_UpdateMeasurement to execute a measure process.

```plaintext
DIM ValidForRec AS Logical
DIM RetCodeForMsg AS Integer
DIM WaitTime AS Integer
DIM iButton AS Integer

WaitTime = 10 'ms

' user definition of measurement dialog
' can be placed here
```

```
6-176  TPS1100-Version 1.30
```
GSI_CreateMDlg( 1, "WIR", "Measure Dialog", "This is the Helpertext")
DO
  GSI_UpdateMeasurement( TMC_MEA_INC,
                         WaitTime, ValidForRec,
                         RetCodeForMsg, FALSE )
  GSI_UpdateMDlg (iButton)
LOOP UNTIL iButton = MMI_ESC_KEY
GSI_DeleteDialog()

6.4.40 GSI_SetLineMDlg

Description Sets one line in the user definable measurement dialog to system parameter.

Declaration GSI_SetLineMDlg(
  BYVAL iLineNr AS Integer
  BYVAL iSysParamId AS Integer )

Remarks This routine sets the configuration of a line in the user definable measurement dialog to a system parameter. This measurement dialog is initialized automatically with the actual settings of the first system measurement dialog. Modifications of the user definable dialog have no effects on the system measurement dialog and will be lost after termination of the program. An unused line should be set to GSI_PAR_NONE. To add a user definable application parameter to the dialog use GSI_SetLineMDlgPar. To add a line of text (e.g. separator line) to the dialog use GSI_SetLineMDlgText.

Parameters

iLineNr in The number of the line to set. Valid numbers:
                     1.. GSI_MAX_DLG_LINES
iSysParamId in Identification of the system parameter. Refer to the chapter "Constants for Measurement Dialog Definition"
See Also
- GSI_SetLineMDlgPar
- GSI_SetLineMDlgText
- GSI_CreateMDlg

Example
This example uses GSI_SetLineMDlg to configure the user definable measurement dialog.

GSI_SetLineMDlg( 1, GSI_PAR_ReflHeight )
GSI_SetLineMDlg( 2, GSI_PAR_Info1 )
GSI_SetLineMDlg( 3, GSI_PAR_Info2 )
...
GSI_SetLineMDlg( 10, GSI_PAR_NONE )
GSI_SetLineMDlg( 11, GSI_PAR_NONE )
GSI_SetLineMDlg( 12, GSI_PAR_NONE )

6.4.41 GSI_SetLineMDlgText

Description
Puts a text line into the user definable measurement dialog.

Declaration
GSI_SetLineMDlgText(  
BYVAL iLineNr AS Integer,  
BYVAL iParamId AS Integer,  
BYVAL sText AS _Token )

Remarks
This routine inserts a pure text line into the user definable measurement dialog. To add an user definable application parameter to the dialog use GSI_SetLineMDlgPar. To add a system parameter to the dialog use GSI_SetLineMDlg.

Parameters
- **iLineNr** in The number of the line to set. Valid numbers: 1.. GSI_MAX_DLG_LINES
- **iParamId** in Id of the system parameter.
- **sText** in Contents of the line.

See Also
- GSI_SetLineMDlg
- GSI_SetLineMDlgPar
- GSI_CreateMDlg
Example

This example uses GSI_SetLineMDlg and GSI_SetLineMDlgText to configure the user definable measurement dialog.

GSI_SetLineMDlg(1, GSI_PAR_Date)
GSI_SetLineMDlg(2, GSI_PAR_Time)
GSI_SetLineMDlgText(3, GSI_PAR_APPDATA0, "------------------")
GSI_SetLineMDlg(4, GSI_PAR_Info1)
GSI_SetLineMDlg(5, GSI_PAR_Info2)

6.4.4 GSI_SetLineMDlgPar

Description

Sets one line in the user definable measurement dialog to an application parameter.

Declaration

GSI_SetLineMDlgPar(
    BYVAL iLineNr AS Integer
    BYVAL iApplParamId AS Integer
    BYVAL sLabel AS _Token
    BYVAL lEditable AS Logical
    BYVAL iFormat AS Integer
)

Remarks

This routine sets the configuration of a line in the user definable measurement dialog to an application parameter. The style of the application parameter is also defined in this routine. Any floating point format and strings are valid formats. The starting values of every application parameter is not predefined and hence has to be set explicitly. To initialize an application parameter the routine GSI_SetWiEntry can be used. To add a line of text to the dialog use GSI_SetLineMDlgText. To add a system parameter to the dialog use GSI_SetLineMDlg.

Parameters

iLineNr in The number of the line to set.
Valid numbers:
1.. GSI_MAX_DLG_LINES
iApplParamId in Id of the application parameter.
sLabel in Description of parameter on display.
lEditAble in

Edit ability of the value in the measurement dialog.

iFormat in

Format descriptor of the application parameter. The format defines if a dimension field is available. Following values can be used:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_FFORMAT_STRING</td>
<td>string</td>
</tr>
<tr>
<td>MMI_FFORMAT_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>MMI_FFORMAT_DISTANCE</td>
<td>distance</td>
</tr>
<tr>
<td>MMI_FFORMAT_SUBDISTANCE</td>
<td>sub-distance [mm]</td>
</tr>
<tr>
<td>MMI_FFORMAT_ANGLE</td>
<td>angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_VANGLE</td>
<td>vertical angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_HZANGLE</td>
<td>horizontal angle</td>
</tr>
<tr>
<td>MMI_FFORMAT TEMPERATURE</td>
<td>temperature</td>
</tr>
</tbody>
</table>

See Also

GSI_SetLineMDlg
GSI_SetLineMDlgText
GSI_CreateMDlg

Example

See also sample file “meas.gbs”.
This example uses GSI_SetLineMDlgPar and GSI_SetWiEntry to configure the user definable measurement dialog.
DIM WI AS GSI_WIDLG_ENTRY_TYPE
WI.lValid = FALSE
WI.iDataType = GSI_ASCII
GSI_SetWiEntry(GSI_ID_APPDATA0, WI)
GSI_SetLineMDlgPar(1, GSI_PAR_AppData0, "Stat. Name: ", TRUE, MMI_FFORMAT_STRING)
WI.lValid = TRUE
WI.iDataType = GSI_DOUBLE
WI.dValue = 2.2
GSI_SetWiEntry(GSI_ID_APPDATA3, WI)
GSI_SetLineMDlgPar(8, GSI_PAR_AppData3, "Distance: ", TRUE, MMI_FFORMAT_DISTANCE)

6.4.43 GSI_UpdateMDlg

Description Updates the user definable measurement dialog.

Declaration GSI_UpdateMDlg (iButton As Integer)

Remarks This procedure updates the user definable measurement dialog
with the actual values from the Theodolite data pool and returns
pressed buttons.

Parameters

iButton out Contains pressed button identifier. For
details see MMI_GetButton
(lAllKeys = TRUE).

See Also GSI_CreateMDlg
GSI_UpdateMeasurement

Example See example GSI_CreateMDlg and example file
"meas.gbs".
### 6.4.44 GSI_DefineMDlg

**Description**
Defines the entries of the user definable measurement dialog.

**Declaration**
GSI_DefineMDlg( BYVAL sCaption AS _Token)

**Remarks**
Interactively defines the contents of the user definable measurement dialog. Using a dialog with list fields, the user can select the items for the measurement dialog. This routine is an interactive equivalent to the routines GSI_SetLineSysMDlg and GSI_GetLineSysMDlg.

**Parameters**
- **sCaption in**
The left caption of the title bar. (Up to 5 characters wide.)

**See Also**
GSI_GetDlgMask
GSI_SetDlgMask

**Example**
GSI_DefineMDlg( "DEF" )

### 6.4.45 GSI_UpdateMeasurement

**Description**
Update the measurement data.

**Declaration**
GSI_UpdateMeasurement(
  iInclinePrg AS Integer,
  iWaitTime AS Integer,
  lValidForRec AS Logical,
  iRetCodeForMsg AS Integer,
  lChkIncRangeNow AS Logical )

**Remarks**
This function updates the measurement values in the Theodolite data pool. The data are the incline program, angles, distances, time, reflector height.
Parameters

iInclinePrg  in  The manner of incline compensation. Following settings are possible:

<table>
<thead>
<tr>
<th>Incline Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_MEA_INC</td>
<td>get inclination</td>
</tr>
<tr>
<td>TMC_AUTO_INC</td>
<td>get inclination with automatism</td>
</tr>
<tr>
<td>TMC_PLANE_INC</td>
<td>get inclination always with plane</td>
</tr>
</tbody>
</table>

iWaitTime  in  The wait time for a result (in ms). This time is used for synchronising the TMC task.

lValidForRec  out  Indicates validity of the registration

iRetCodeForMsg out  Return code of the measurement

lChkIncRange  in  Now

TRUE: check incline range immediate

See Also

GSI_CreateMDlg
GSI_UpdateMDlg
GSI_DeleteDialog

Example

See example GSI_CreateMDlg and example file "meas.gbs".

6.4.46 GSI_Measure

Description  Measure and registration dialog.

Declaration  GSI_Measure ( )

Remarks  This procedure opens the measure and registration dialog.

Parameters

none
Return Codes

| RC_OK     | Success |

Example

Do a measure and registration dialog.

GSI_Measure ( )

6.4.47 GSI_ExecuteAutoDist

Description
Executes an automatic distance measurement.

Declaration
GSI_ExecuteAutoDist ( )

Remarks
This procedure starts a distance measurement on condition that “Auto Dist” is enabled and one of the distance measurement-program buttons (FNC-menu) was pressed.

Parameters

none

Return Codes

| RC_OK     | Success |

Example

See example file “meas.gbs” or “meas_od.gbs”.

6.4.48 GSI_CheckTracking

Description
Returns if distance tracking is running.

Declaration
GSI_CheckTracking ( lTracking As Logical )

Remarks
This returns if a distance tracking is running.

An automatic start of distance tracking can be started on several conditions, i.e. by Quick-Coding, GSI_ExecuteAutoDist or by pressing buttons in the FNC-menu.

Tracking can be terminated by the instrument itself due several reasons, i.e. for laser security reasons (US-configuration)
6.4.49  GSI_RecordRecMask

Description  Recording the given wi mask.

Declaration  GSI_RecordRecMask (  
\begin{verbatim}
RecList AS GSI_REC_ID_LIST,  
BYVAL eProgFunction AS Logical,  
BYVAL bCheckStdMask AS Logical,  
BYVAL bIncAndSetRunPt AS Logical)
\end{verbatim}

Remarks  This procedure records the given wi list. The target can be the memory card or the interface. The parameter for the interface depends on the GSI communication settings. Errors will shown on the display, when recording list will be stored in the memory card. Otherwise the error messages will be given on the interface.

Parameters  
\begin{itemize}
  \item RecList \hspace{1cm} \textbf{in} \hspace{1cm} recording list
  \item eProgFunction \hspace{1cm} \textbf{in} \hspace{1cm} program flag in the wi's (TRUE = ON, FALSE = OFF)
  \item bCheckStdMask \hspace{1cm} \textbf{in} \hspace{1cm} testing the standard recording mask
  \item bIncAndSetRunPt \hspace{1cm} \textbf{in} \hspace{1cm} increment the point number
\end{itemize}

Return Codes  
\begin{itemize}
  \item RC_OK \hspace{1cm} \textbf{Success}
  \item RC_IVRESULT \hspace{1cm} registration failure
\end{itemize}

See Also
Example

Record RecList.

DIM RecList AS GSI_REC_ID_LIST

' initialize RecList with adequate values
GSI_RecordRecMask ( RecList, TRUE, TRUE, TRUE )
### 6.5 CENTRAL SERVICE FUNCTIONS CSV

#### 6.5.1 Summarizing Lists of CSV Types and Procedures

**6.5.1.1 Types**

<table>
<thead>
<tr>
<th>Type name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPS_Fam_Type</td>
<td>Information about the current hardware.</td>
</tr>
<tr>
<td>Date_Time_Type</td>
<td>Date and time information.</td>
</tr>
<tr>
<td>Date_Type</td>
<td>Date information.</td>
</tr>
<tr>
<td>Time_Type</td>
<td>Time information.</td>
</tr>
</tbody>
</table>

**6.5.1.2 Procedures**

<table>
<thead>
<tr>
<th>Procedure name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_ChangeFace</td>
<td>Do an absolute positioning to the opposite.</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>Returns if an alternative user-task was running.</td>
</tr>
<tr>
<td>CSV_Delay</td>
<td>Delay routine</td>
</tr>
<tr>
<td>CSV_GetATRStatus</td>
<td>Gets the current ATR state.</td>
</tr>
<tr>
<td>CSV_GetDateTime</td>
<td>Get the date and the time of the system.</td>
</tr>
<tr>
<td>CSV_GetElapseSysTime</td>
<td>Returns the difference between a reference time and the system time.</td>
</tr>
<tr>
<td>CSV_GetGBIVersion</td>
<td>Returns the release number of the GeoBASIC interpreter</td>
</tr>
<tr>
<td>CSV_GetInstrumentFamily</td>
<td>Get information about the system.</td>
</tr>
<tr>
<td>CSV_GetInstrumentName</td>
<td>Get the LEICA specific instrument name.</td>
</tr>
<tr>
<td>CSV_GetInstrumentNo</td>
<td>Get the instrument number.</td>
</tr>
<tr>
<td>CSV_GetLaserPlummet</td>
<td>Returns the laser plummet state</td>
</tr>
<tr>
<td>CSV_GetLockStatus</td>
<td>Gets the current state of the locking facility.</td>
</tr>
<tr>
<td>CSV_GetLRStatus</td>
<td>Returns the status of the system.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>CSV_GetPrismType</td>
<td>Returns the used prism</td>
</tr>
<tr>
<td>CSV_GetSWVersion</td>
<td>Get the version of the system software.</td>
</tr>
<tr>
<td>CSV_GetSysTime</td>
<td>Returns the system time.</td>
</tr>
<tr>
<td>CSV_GetTargetType</td>
<td>Get the target type for distance measurements.</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>Returns the internal temperature of the instrument.</td>
</tr>
<tr>
<td>CSV_Laserpointer</td>
<td>Switch on / off the laser pointer.</td>
</tr>
<tr>
<td>CSV_LibCall</td>
<td>Call a GeoBASIC routine from another program.</td>
</tr>
<tr>
<td>CSV_LibCallAvailable</td>
<td>Check if GeoBASIC routine from another program is available.</td>
</tr>
<tr>
<td>CSV_LockIn</td>
<td>Starts locking (ATR)</td>
</tr>
<tr>
<td>CSV_LockOut</td>
<td>Stops locking (ATR)</td>
</tr>
<tr>
<td>CSV_MakePositioning</td>
<td>Do an absolute positioning.</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>Resets the “alternative user-task was running” flag.</td>
</tr>
<tr>
<td>CSV_SetATRStatus</td>
<td>Sets the current state of Automatic Target Recognition.</td>
</tr>
<tr>
<td>CSV_SetLaserPlummet</td>
<td>Switches the laser plummet</td>
</tr>
<tr>
<td>CSV_SetLightGuide</td>
<td>Switch on / off the light guide.</td>
</tr>
<tr>
<td>CSV_SetLockStatus</td>
<td>Sets the current state of the locking facility.</td>
</tr>
<tr>
<td>CSV_SetPrismType</td>
<td>Sets the used prism</td>
</tr>
<tr>
<td>CSV_SetTargetType</td>
<td>Set the target type for distance measurements.</td>
</tr>
<tr>
<td>CSV_SysCall</td>
<td>Call a system function.</td>
</tr>
<tr>
<td>CSV_SysCallAvailable</td>
<td>Check if system function is available.</td>
</tr>
</tbody>
</table>
6.5.2 Data Structures for the Central Service Functions

6.5.2.1 Date_Time_Type: Date and Time

Description: These data structures are used to store date and time information.

```
TYPE Date_Type
  iYear  AS Integer  year as a 4 digit number
  iMonth AS Integer  month as a 2 digit number
  iDay   AS Integer  day as a 2 digit number
END Date_Type

TYPE Time_Type
  iHour  AS Integer  hour as a 2 digit number (24 hours format)
  iMinute AS Integer  minutes as a 2 digit number
  iSecond AS Integer  seconds as a 2 digit number
END Time_Type

Date_Time_Type
  Date  AS Date_Type  date (as defined above)
  Time  AS Time_Type  time (as defined above)
END Date_Time_Type
```
6.5.2.2 TPS_Fam_Type: Information about the system

**Description**
This data structure is used to store information about the hardware. Further information about the hardware can be obtained by your local Leica representative.

```
TYPE TPS_Fam_Type
    iClass AS Integer
        The class of the system. Values:
        Id     Meaning
        TPS1101 TPS1100 accuracy 1"
        TPS1102 TPS1100 accuracy 2"
        TPS1103 TPS1100 accuracy 3"
        TPS1105 TPS1100 accuracy 5"
    lEDMBuiltIn AS Logical
        EDM built-in
    lEDMTypeII AS Logical
        EDM built-in, type II
    lEDMTypeIII AS Logical
        EDM built-in, type III
    lEDMReflectorless AS Logical
        Red Laser
    lMotorized AS Logical
        Motorised
    lATR AS Logical
        Automatic Target Recognition (ATR)
    lEGL AS Logical
        EGL Guide Light
    lLaserPlummet AS Logical
        Laser Plummet
    lAutoCollimation AS Logical
        Auto-collimation lamp
    lSimulator AS Logical
        Hardware is simulator on Windows-PC
END TPS_Fam_Type
```
6.5.3 CSV_GetDateTime

Description
Get the date and the time of the system.

Declaration
CSV_GetDateTime(
    DateAndTime AS Date_Time_Type )

Remarks
The CSV_GetDateTime routine reads the date and the time from the system's real-time clock (RTC) and returns the values in the structure Date_Time_Type. In the case of TPS_Sim the system clock will be read.

Parameters
DateAndTime out The structure for the date and the time.

Return Codes
RC_UNDEFINED The date and time is not set (not yet/not any longer).

Example
The example uses the CSV_GetDateTime routine to get the date and the time of the system and displays the values.

```
DIM DT AS Date_Time_Type
ON ERROR RESUME
CSV_GetDateTime( DT )

IF ERR = RC_OK THEN
    MMI_PrintInt( 0, 0, 5, DT.Date.iYear, TRUE )
    MMI_PrintInt( 6, 0, 3, DT.Date.iMonth, TRUE )
    MMI_PrintInt( 10, 0, 3, DT.Date.iDay, TRUE )
    MMI_PrintInt( 0, 1, 3, DT.Time.iHour, TRUE )
    MMI_PrintInt( 4, 1, 3, DT.Time.iMinute, TRUE )
    MMI_PrintInt( 8, 1, 3, DT.Time.iSecond, TRUE )
ELSEIF ERR = RC_UNDEFINED THEN
    MMI_PrintStr( 0, 0, "Date and time not set.", TRUE )
ELSE
    MMI_PrintStr( 0, 0, "Unexpected error code.", TRUE )
END IF
```
### 6.5.4 CSV_GetTemperature

**Description**
Returns the internal temperature of the instrument.

**Declaration**
```basic
CSV_GetTemperature( IntTemp AS Temperature )
```

**Remarks**
This routine returns the internal temperature.

**Parameters**
- `IntTemp` out Internal temperature

### 6.5.5 CSV_GetInstrumentName

**Description**
Get the LEICA specific instrument name.

**Declaration**
```basic
CSV_GetInstrumentName( sName AS String30 )
```

**Remarks**
The `CSV_GetInstrumentName` routine returns the name of the system in the string `sName`.

**Parameters**
- `sName` out The LEICA specific instrument name.

**Return Codes**
- `none`

**See Also**
`CSV_GetInstrumentNo`,
`CSV_GetInstrumentFamily`

**Example**
The example uses the `CSV_GetInstrumentName` routine to get the instrument name and displays it.

```basic
DIM sName AS String30

CSV_GetInstrumentName ( sName )
MMI_PrintStr ( 0, 0, sName, TRUE )
```
### 6.5.6 CSV_GetInstrumentNo

**Description**  
Get the instrument number.

**Declaration**  
`CSV_GetInstrumentNo( iSerialNo AS Integer )`

**Remarks**  
The `CSV_GetInstrumentNo` routine returns the serial number of the system.

**Parameters**

- `iSerialNo` out  
The serial number of the system.

**Return Codes**

- none

**See Also**

- CSV_GetInstrumentName, CSV_GetInstrumentFamily

**Example**  
The example uses the `CSV_GetInstrumentNo` routine to get the instrument number and displays it.

```basic
DIM iSerialNo AS Integer
CSV_GetInstrumentNo( iSerialNo )
MMI_PrintInt( 0, 1, 20, iSerialNo, TRUE )
```

### 6.5.7 CSV_GetInstrumentFamily

**Description**  
Get information about the system.

**Declaration**  
`CSV_GetInstrumentFamily( Family AS TPS_Fam_Type )`

**Remarks**  
The `CSV_GetInstrumentFamily` routine returns the class and the instrument type of the system (see description of the data structure TPS_Fam for return values).

**Parameters**

- `Family`  
The family of the instrument.

**Return Codes**

- none

**See Also**

- CSV_GetInstrumentName, CSV_GetInstrumentFamily

**Example**  
The example uses the `CSV_GetInstrumentFamily` routine to get the instrument family and displays it.

```basic
DIM Family AS TPS_Fam_Type
CSV_GetInstrumentFamily( Family )
MMI_PrintInt( 0, 1, 20, Family.lSimulator, TRUE )
```
Parameters

Family out
Contains the class and instrument type data. See description of the data structure TPS_Fam for return values.

See Also
CSV_GetInstrumentName,
CSV_GetInstrumentNo

Example
The example uses the CSV_GetInstrumentFamily routine to get information about the instrument and displays it.

```
DIM Family AS TPS_Fam_Type
CSV_GetInstrumentFamily( Family )
MMI_PrintInt( 0, 1, 10, Family.iClass, TRUE )
IF (Family.lSimulator) THEN
    MMI_PrintString( 0, 2, 10, "ON TPS_SIM", TRUE)
END IF
```

6.5.8 CSV_GetSWVersion

Description
Get the version of the system software.

Declaration
```
CSV_GetSWVersion( iRelease AS Integer, iVersion AS Integer )
```

Remarks
The CSV_GetSWVersion routine returns the Release number and the number of the system software version. These numbers can be interpreted together as software identification (Release.Version, e.g. 1.05).

```
TPS_Sim Delivers the version of the simulator.
```

Parameters

iRelease out
value of the Release number can be in the range from 0 to 99

iVersion out
value of the version number can be in the range from 0 to 99

See Also
Example  The example uses the CSV_GetSWVersion routine to get the system software version and displays it.

```
DIM iRelease AS Integer
DIM iVersion AS Integer

CSV_GetSWVersion( iRelease, iVersion )
MMI_PrintVal( 0, 0, 6, 2,
              iRelease + iVersion / 100, TRUE )
```

### 6.5.9 CSV_GetGBIVersion

**Description**  Returns the release number of the GeoBASIC interpreter.

**Declaration**

```
CSV_GetGBIVersion( 
    iRelease  as Integer, 
    iVersion as Integer, 
    iSubVersion as Integer )
```

**Remarks**  This function returns the release version of the running GeoBASIC interpreter.

**Parameters**

- `iRelease` out  Release number
- `iVersion` Out  Version Number
- `iSubVersion` out  Subversion number

**Return-Codes**

- `RC_OK`  Successful termination.
**Example**

This example shows the currently used GeoBASIC interpreter release number.

```
DIM iRel As Integer
DIM iVer As Integer
DIM iSubVer As Integer

MMI_CreateTextDialog(6, "-CSV-", "Test CSV", "no help available")
CSV_GetGBIVersion(iRel, iVer, iSubVer)
MMI_PrintStr(0, 0, "GBI: " + Str$(iRel) + "." + Str$(iVer) + "." + Str$(iSubVer), TRUE)
MMI_DeleteDialog()
```

### 6.5.10 CSV_GetElapseSysTime

**Description**

Returns the difference between a reference time and the system time.

**Declaration**

```
CSV_GetElapseSysTime (iRefTime AS Integer, iElapse AS Integer)
```

**TPS_Sim**

Use PC time base. Time resolution is one second.

**Remarks**

The routine `CSV_GetElapseSysTime` returns the difference of between a given reference time `iRefTime` and the systems time. Whenever the system starts up, the system time is reset.

**Parameters**

- `iRefTime in` The reference time.
- `iElapse out` The difference between `iRefTime` and the system time. The difference is returned in [ms].

**See Also**

- `CSV_GetSysTime`
- `CSV_GetDateTime`
Example
The example uses the routine CSV_GetElapsedTime to get a time difference.

```
DIM iElapse AS Integer
DIM iRefTime AS Integer

CSV_GetSysTime(iRefTime) 'returns reference time
' do something ..
CSV_GetElapsedTime(iRefTime, iElapse)
MMI_PrintInt ( 0, 0, 20, iElapse, TRUE )
```

### 6.5.11 CSV_GetSysTime

**Description**
Returns the system time.

**Declaration**
```
CSV_GetSysTime( iTime AS Integer )
```

**Remarks**
The routine returns the system's time. Whenever the system starts up, the system time is reset.

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTime out</td>
</tr>
</tbody>
</table>

**See Also**
CSV_GetElapsedTime, CSV_GetDateTime

**Example**
See CSV_GetElapsedTime.

### 6.5.12 CSV_GetLRStatus

**Description**
Returns the status of the system.

**Declaration**
```
CSV_GetLRStatus( iLRStatus AS Integer )
```

**Remarks**
The routine CSV_GetLRStatus returns the mode of the system. The system can either be in local or in Remote mode. For Release 1.0 this function always delivers local mode as an answer.
**Note**  This function is reserved for future purposes and has no special usage in the current implementation.

**TPS_Sim**  Always delivers LOCAL_MODE.

**Parameters**

<table>
<thead>
<tr>
<th>iLRStatus</th>
<th>The mode of the system. Possible values for the iLRStatus are:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode</strong></td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td>LOCAL_MODE</td>
<td>0</td>
</tr>
<tr>
<td>REMOTE_MODE</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example**  The example uses the routine CSV_GetLRStatus to get the mode of the system.

```
DIM iLRStatus AS Integer

CSV_GetLRStatus( iLRStatus )

MMI_PrintInt( 0, 0, 10, iLRStatus, TRUE )
```

### 6.5.13 CSV_SetGuideLight

**Description**  Set the guide light intensity.

**Declaration**  `CSV_SetGuideLight( BYVAL iLight AS Integer )`

**Remarks**  Sets the guide light intensity.

**Parameters**

<table>
<thead>
<tr>
<th>iLight</th>
<th>in</th>
<th>Guide light intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
<td><strong>Meaning</strong></td>
<td></td>
</tr>
<tr>
<td>CSV_EGL_OFF</td>
<td>Switching off</td>
<td></td>
</tr>
<tr>
<td>CSV_EGL_LOW</td>
<td>Low intensity</td>
<td></td>
</tr>
<tr>
<td>CSV_EGL_MID</td>
<td>Middle intensity</td>
<td></td>
</tr>
<tr>
<td>CSV_EGL_HIGH</td>
<td>High intensity</td>
<td></td>
</tr>
</tbody>
</table>

**Return Codes**

| RC_SYSBUSY | EDM is busy. Guide light cannot be switched. |
| RC_NOT_IMPL | Guide light Hardware is not available |
Example
Switch off the Light guide.
CSV_SetGuideLight( CSV_EGL_OFF )

6.5.14 CSV_Laserpointer

Description
Switch on / off the laser pointer.

Declaration
CSV_Laserpointer( BYVAL lLaser AS Logical )

Remarks
Switches on / off the laser pointer.

Parameters
lLaser in Switch on / off the Laser pointer (TRUE = on, FALSE = off)

Return Codes
RC_SYSBUSY EDM is busy. Laser pointer cannot be switched.
RC_NOT_IMPL Laser pointer Hardware is not available.

Example
Switch off the laser pointer.
CSV_Laserpointer( FALSE )

6.5.15 CSV_MakePositioning

Description
Do an absolute positioning.

Declaration
CSV_MakePositioning(BYVAL dHz AS Double,
                     BYVAL dV AS Double)

Remarks
Absolute positioning of the Theodolite axes to the desired angles with the currently active tolerance for positioning. If any control function is active at the point of call, it will be cancelled and the positioning will be performed. After the positioning the controller will be automatically activated for manual input for the moving device. When starting the positioning the calling application has to take care that a valid inclination plane is available for an angle measure, as it can normally not be redone during positioning. The positioning is done with the planes valid at the beginning of
it. During the process no inclination will be measured. The used positioning method can cause inexact results, especially for steep \( V > \sim 25 \text{ GON} \)

### Parameters

- \( dHz \text{ in Corrected Hz-angle [Radiant]} \)
- \( dV \text{ in Corrected V-angle [Radiant]} \)

### Return Codes

- **RC_IVPARAM**: No valid positioning angle.
- **CSV_DETENT_ERROR**: Target angle is out of the limits or a collision is occurred.
- **CSV_TIMEOUT**: Time out at positioning of one or both axes.
- **CSV_MOTOR_ERROR**: Error in subsystem.
- **CSV_ANGLE_ERROR**: Error at measuring the angle.
- **RC_FATAL**: Fatal error.
- **RC_ABORT**: System abort.

### See Also

- BAP_PosTelescope

### Example

Perform an absolute positioning.

```plaintext
CSV_MakePositioning( 0, 2*atn(1) ) ' (0, Pi/2)
```

#### 6.5.16 CSV_ChangeFace

### Description

Do an absolute positioning to the opposite.

### Declaration

```plaintext
CSV_ChangeFace()
```

### Remarks

Perform positioning into the position opposite to the current. If any control function is active at the point of call, it will be cancelled and the positioning will be performed. After the positioning the controller will be automatically activated for manual input for the moving device. When starting the positioning the calling application has to take care that a valid inclination plane is available for an angle measure, as it can normally not be redone during positioning.
The positioning is done with the planes valid at the beginning of it. During the process no inclination will be measured. The used positioning method can cause inexact results, especially for steep \( V > \sim 25 \) GON

**Parameters**

none

**Return Codes**

- RC_IVPARAM: No valid positioning angle.
- CSV_DETENT_ERROR: Target angle is out of the limits or a collision is occurred.
- CSV_TIMEOUT: Time out at positioning of one or both axes.
- CSV_MOTOR_ERROR: Error in subsystem.
- CSV_ANGLE_ERROR: Error at measuring the angle.
- RC_FATAL: Fatal error.
- RC_ABORT: System abort.

**See Also**

BAP_PosTelescope

**Example**

Perform a change of face.

```
CSV_ChangeFace()
```

### 6.5.17 CSV_SetLockStatus

**Description**

Sets the current state of the locking facility.

**Declaration**

```
CSV_SetLockStatus(BYVAL lOn AS Logical)
```

**Remarks**

It switches the locking facility on or off.

**Parameters**

- lOn: Switches on / off the locking facility (TRUE = on, FALSE = off)
### CSV_GetLockStatus

**Description**
Gets the current state of the locking facility.

**Declaration**

```basic
CSV_GetLockStatus( lOn AS Logical )
```

**Remarks**
It queries the TPS system if the locking facility is on or off.

**Parameters**

- **lOn**
  - **Out**
  - **Meaning**
    - `FALSE`: Locking is switched off.
    - `TRUE`: Locking is switched on.

**Return Codes**

- **RC_FATAL**: fatal error
- **RC_NOT_IMPL**: if ATR hardware is not available
- **RC_ABORT**: system abort

**See Also**

- CSV_GetLockStatus,
- CSV_LockIn,
- CSV_LockOut

**Example**
Perform an absolute positioning.

```basic
DIM l AS Logical
CSV_SetLockStatus( l ) ' queries locking
```
### CSV_LockIn

**Description**
Starts the locking facility.

**Declaration**
`CSV_LockIn( )`

**Remarks**
If ATR is switched on then locking to the target will be done. If no target available, then manual positioning will be started.

**Parameters**
`none`

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT_RC_NOT_ENABLED</td>
<td>Theodolite without ATR or lock status not set</td>
</tr>
<tr>
<td>AUT_RC_MOTOR_ERROR</td>
<td>Error at motor control.</td>
</tr>
<tr>
<td>AUT_RC_DETECTOR_ERROR</td>
<td>Error at ATR</td>
</tr>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>No target at the detection range</td>
</tr>
<tr>
<td>AUT_RC_BAD_ENVIRONMENT</td>
<td>Bad environment at the detection range (bad light…)</td>
</tr>
<tr>
<td>RC_NOT_IMPL</td>
<td>if ATR hardware is not available</td>
</tr>
</tbody>
</table>

**See Also**
`CSV_GetLockStatus`,
`CSV_SetLockStatus`,
`CSV_LockOut`

**Example**
This example starts locking.

```
CSV_LockIn( )
```
6.5.20  CSV_LockOut

**Description**  Stops a running locking function.

**Declaration**  CSV_LockOut ( )

**Parameters**  none

**Return Codes**
- RC_OK: no error
- RC_NOT_IMPL: if ATR hardware is not available

**See Also**  CSV_GetLockStatus, CSV_SetLockStatus, CSV_LockIn

**Example**  This example stops locking.

```
CSV_LockOut( )
```

6.5.21  CSV_SetATRStatus

**Description**  Sets the current state of Automatic Target Recognition.

**Declaration**  CSV_SetATRStatus(BYVAL lOn AS Logical )

**Remarks**  It switches the ATR facility on or off.

**Parameters**
- lOn in  Switches on / off the ATR facility
  (TRUE = on, FALSE = off)

**Return Codes**
- RC_FATAL: fatal error
- RC_ABORT: system abort
- RC_NOT_IMPL: if ATR hardware is not available

**Example**  Perform an absolute positioning.

```
CSV_SetATRStatus( TRUE ) ' switches ATR on
```
6.5.22 CSV_GetATRStatus

**Description**
Gets the current ATR state.

**Declaration**
```plaintext
CSV_GetATRStatus(lOn1 AS Logical )
```

**Remarks**
It queries the TPS system if the ATR facility is on or off.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>lOn</code></td>
<td>out</td>
</tr>
<tr>
<td>FALSE</td>
<td>ATR is switched off.</td>
</tr>
<tr>
<td>TRUE</td>
<td>ATR is switched on.</td>
</tr>
</tbody>
</table>

**Return Codes**

- **RC_FATAL**  fatal error
- **RC_ABORT**  system abort
- **RC_NOT_IMPL**  if ATR hardware is not available

**Example**
Get current ATR status.

```plaintext
DIM l AS Logical
CSV_SetATRStatus( l )
```

6.5.23 CSV_Delay

**Description**
This routine delays the execution of a program.

**Declaration**
```plaintext
CSV_Delay( BYVAL iDelay AS Integer )
```

**Remarks**
This routine delay using the operating system, that means that other Theodolite tasks can run during the delay (It is not a busy waiting).

**Note**
Avoid busy waiting using FOR - or WHILE loops.

**TPS_Sim**
Delay resolution is one second. `iDelay < 500` means no delay
6.5.24 CSV_SetTargetType

**Description**
Set the target type for distance measurements.

**Declaration**
```basic
CSV_SetTargetType(
    BYVAL iTargetType as Integer )
```

**Remarks**
This routine sets the target type for distance measurements. The target type defines if the next distance measurement happens with prism or without prism.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTargetType</td>
<td>Integer</td>
<td>Target type</td>
</tr>
</tbody>
</table>

**Valid target types**
- CSV_WITH_REFLECTOR: With reflector
- CSV_WITHOUT_REFLECTOR: Without reflector

**Return-Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Instrument don’t support this target type</td>
</tr>
</tbody>
</table>
6.5.25 CSV_GetTargetType

**Description**
Get the target type for distance measurements.

**Declaration**
CSV_GetTargetType( iTargetType as Integer )

**Remarks**
This routine fetches the target type for distance measurements. The target type defines if the next distance measurement happens with prism or without prism.

**Parameters**
- **iTargetType** out Target type

<table>
<thead>
<tr>
<th>Valid target types</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_WITH_REFLECTOR</td>
<td>With reflector</td>
</tr>
<tr>
<td>CSV_WITHOUT_REFLECTOR</td>
<td>Without reflector</td>
</tr>
</tbody>
</table>

**Return-Codes**
- **RC_OK** Successful termination.

**See**
CSV_SetTargetType, BAP_SetMeasPrg, BAP_GetMeasPrg

**Example**
The example fetches the target type.

```basic
DIM iTargetType AS Integer
CSV_GetTargetType(iTargetType)
```
### 6.5.26 CSV_SetPrismType

**Description**
Sets the used prism.

**Declaration**

```vbnet
CSV_SetPrismType( BYVAL iPrism as Integer)
```

**Remarks**
This routine sets the used prism `iPrism` (BAP_PRISM_ROUND, BAP_PRISM_TAPE, BAP_PRISM_MINI, BAP_PRISM_360, BAP_PRISM_USER1, BAP_PRISM_USER2 or BAP_PRISM_USER3). If `iPrism` is one of the user defined prisms and this prism is actually not defined then this routine will return RC_IVRESULT.

**Parameters**
- `iPrism` in Used prism

**Return-Codes**
- RC_OK Successful termination.
- RC_IVRESULT Prism not defined.

**See**
CSV_GetPrismType

**Example**
The example sets the 360 degrees prism.

```vbnet
CSV_SetPrismType(BAP_PRISM_360)
```

### 6.5.27 CSV_GetPrismType

**Description**
Returns the used prism.

**Declaration**

```vbnet
CSV_GetPrismType(iPrism as Integer)
```

**Remarks**
This routine returns the used prism `iPrism`.

**Parameters**
- `iPrism` out Used prism

**Return-Codes**
- RC_OK Successful termination.
See  CSV_SetPrismType

Example  The example returns the used prism.

    DIM iPrism AS Integer
    CSV_SetPrismType( iPrism )

6.5.28  CSV_SetLaserPlummet

Description  Switches the laser plummet.

Declaration  CSV_SetLaserPlummet( BYVAL lOn as Logical )

Remarks  This function switches the optional laser plummet. The plummet will be switched off automatically after 3 minutes.

Parameters

    lOn  in  TRUE: switch plummet on

Return-Codes

   RC_OK  Successful termination.

See  CSV_GetLaserPlummet, CSV_GetInstrumentFamily

6.5.29  CSV_GetLaserPlummet

Description  Returns the laser plummet state.

Declaration  CSV_GetLaserPlummet( lOn as Logical )

Remarks  This function returns the state of the optional laser plummet.

Parameters

    lOn  out  TRUE: plummet is switched on

Return-Codes

   RC_OK  Successful termination.
6.5.30 CSV_CheckAltUserTask

**Description**
Returns if an alternative user-task was running.

**Declaration**
CSV_CheckAltUserTask(lWasRunning AS Logical)

**Remarks**
This routine returns if an alternative user-task was running. One of these tasks can be started by pressing one of the buttons FNC, Shift-FNC, PROG, Shift-PROG, Light and Level. Functions, executed by an alternative user task, can change several system settings. The CSV_CheckAltUserTask routine notifies the running GeoBASIC application that it was interrupted by another program. With this information, the GeoBASIC program is able to respond to these changes.

After processing this information, the subroutine CSV_ResetAltUserTask must be called.

**Parameters**
lWasRunning out TRUE: a task was running

**Return-Codes**
RC_OK Successful termination.

**See**
CSV_ResetAltUserTask

**Example**
The example checks if an alternative task was running.

```geo
CSV_CheckAltUserTask( l )
IF l THEN
send("AltUserTask: was running")
ELSE
send("AltUserTask: was NOT running")
END IF
CSV_ResetAltUserTask( )
```

6.5.31 CSV_ResetAltUserTask

**Description**
Resets the “alternative user-task was running” flag.

**See**
CSV_ResetAltUserTask
Declaration  CSV_ResetAltUserTask( )
Remarks  This routine restarts the alternative user-task tracking.
Parameters  none
Return-Codes  
RC_OK  Successful termination.
See  CSV_CheckAltUserTask

6.5.32 CSV_SysCall

Description  Call a system function.
Declaration  CSV_SysCall( BYVAL CId AS CIdType)
Remarks  This routine works in two different forms depending on the parameter CId. If CId is a system function CSV_SysCall calls the function directly. In the other form the CId is a system event. In this case CSV_SysCall calls the system function (or dialog, menu, macro, application) which is defined in the current configuration to handle this event. See description of the system functions and system events in the appendix H.
Parameters  CId  in  System function or system event
Return-Codes  
RC_OK  Successful termination.
RC_IVPARAM  No function defined to handle the event
RC_NOT_IMPL  System function not available
See CSV_SysCallAvailable

Example The example calls the system function electronic level.

\[ \text{CSV.SysCall(CSV_SFNC_Libelle)} \]

### 6.5.33 CSV_SysCallAvailable

**Description** Check if system function is available.

**Declaration**

\[
\text{CSV.SysCallAvailable(}\quad \text{BYVAL CId AS CIdType}, \\
\text{}\quad \text{lAvailable AS Logical})
\]

**Remarks** This routine checks, if it is possible to call the function CId if CId is a system function or if there is a function defined and available to handle the event CId if CId is an system event. See the description of system functions and system events in appendix H.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CId</td>
<td>System function or system event.</td>
</tr>
<tr>
<td>lAvailable</td>
<td>TRUE: System function is available or function (dialog, menu, macro, application) to handle the event is defined and available.</td>
</tr>
</tbody>
</table>

**Return-Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>
See  CSV_SysCall

Example  The example checks if the red laser is available.
DIM lAvailable AS Logical

CSV_SysCallAvailable(CSV_SFNC_ToggleRedLaser,
lAvailable)

6.5.34  CSV_LibCall

Description  Call a GeoBASIC or C application routine of another program.

Declaration  CSV_LibCall( BYVAL PrgName AS String255,
BYVAL FuncName AS String255,
BYVAL CptShort AS _Token )

Remarks  This routine is used to call a GeoBASIC routine which is defined in another program. Please refer also to Appendix

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgName</td>
<td>in</td>
<td>Program name</td>
</tr>
<tr>
<td>FuncName</td>
<td>in</td>
<td>Function name</td>
</tr>
<tr>
<td>CptShort</td>
<td>In</td>
<td>Short caption for dialogs</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Return-Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See  CSV_LibCallAvailable

Example  See IAC.GBS and IAC2.GBS for an example.

6.5.35  CSV_LibCallAvailable

Description  Check if the GeoBASIC routine from another program is available.

Declaration  CSV_LibCallAvailable( 
BYVAL PrgName AS String255,
BYVAL FuncName AS String255,
lAvailable AS Logical )
Remarks  This routine checks if a GeoBASIC routine which is defined in another program is available. Usually this means that it checks if the other program is loaded and the specified entry point exists.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgName</td>
<td>in</td>
<td>Program name</td>
</tr>
<tr>
<td>FuncName</td>
<td>in</td>
<td>Function name</td>
</tr>
<tr>
<td>lAvailable</td>
<td>out</td>
<td>Routine is available</td>
</tr>
</tbody>
</table>

Return-Codes

- **RC_OK**  Successful termination.

See  CSV_LibCall

Example  See IAC.GBS and IAC2.GBS for an example.
Appendix A — GeoBASIC SYNTAX

ArrayDeclaration ::= "TYPE" "DIM" Name SubscriptList "AS" DataType "END"

DataType ::= ( DataTypeName | "STRING" "*" Length )
SubscriptList ::= ( "UpperBound { "" UpperBound } "")
UpperBound ::= IntegerConstant
Length ::= IntegerConstant

TypeDeclaration ::= "TYPE" Name

ConstantDeclaration ::= "CONST" Name [ "AS" DataType ] 
Expression

VariableDeclaration ::= "DIM" Name [ SubscriptList ]

Variable ::= VariableName { Selector }
Selector ::= ( ArraySelector | FieldSelector )
ArraySelector ::= ( "SubscriptExpression { "," SubscriptExpression } ")
FieldSelector ::= "," ElementName
SubscriptExpression ::= IntegerExpression

Expression ::= LogicalTerm { "OR" LogicalTerm }
LogicalTerm ::= LogicalFactor { "AND" LogicalFactor }
LogicalFactor ::= { "NOT" } LogicalPrimary
LogicalPrimary ::= SimpleExpression

RelationOperator ::= ( "+" | ">" | ">=" | ">=" | "<" )

AddOperator ::= ( "+" | "-" )
Term ::= Factor { MultiOperator Factor }
MultiOperator ::= ( "+" | "/" | "^" | "MOD" )
Factor ::= Primary [ "+" Factor ]
Primary ::= ( Variable | Constant | FunctionCall )

TPS1100-Version 1.30
6. SYSTEM FUNCTIONS

6.1 MMI Functions ............................................................ 6-7
   6.1.1 Summarising Lists of MMI Types and Procedures .... 6-7
   6.1.2 MMI Data Types ..................................................... 6-9
   6.1.3 MMI_CreateMenuItem .............................................. 6-10
   6.1.4 MMI_CreateGBMenu ................................................ 6-11
   6.1.5 MMI_CreateGBMenuItem ......................................... 6-13
   6.1.6 MMI_CreateGBMenuStr .......................................... 6-14
   6.1.7 MMI_CreateGBMenuItemStr .................................... 6-16
   6.1.8 MMI_DeleteGBMenu ............................................... 6-17
   6.1.9 MMI_SelectGBMenuItem ......................................... 6-17
   6.1.10 MMI_AddGBMenuButton ........................................ 6-18
   6.1.11 MMI_CreateTextDialog ......................................... 6-19
   6.1.12 MMI_CreateGraphDialog ....................................... 6-21
   6.1.13 MMI_DeleteDialog ............................................... 6-22
   6.1.14 MMI_CheckButton ................................................ 6-23
   6.1.15 MMI_GetButton .................................................... 6-24
   6.1.16 MMI_AddButton ..................................................... 6-26
   6.1.17 MMI_DeleteButton ................................................. 6-27
   6.1.18 MMI_PrintStr ...................................................... 6-28
   6.1.19 MMI_PrintTok ..................................................... 6-29
   6.1.20 MMI_PrintVal ...................................................... 6-30
   6.1.21 MMI_PrintInt ...................................................... 6-32
   6.1.22 MMI_InputStr ....................................................... 6-33
   6.1.23 MMI_InputVal ....................................................... 6-35
   6.1.24 MMI_InputInt ....................................................... 6-38
   6.1.25 MMI_InputList ...................................................... 6-40
   6.1.26 MMI_FormatVal ................................................... 6-42
   6.1.27 MMI_WriteMsg .................................................... 6-44
6. **SYSTEM FUNCTIONS**

6. System Functions ................................................................. 6-1

6.1 MMI Functions ................................................................. 6-7

6.1.1 Summarising Lists of MMI Types and Procedures .............. 6-7

6.1.2 MMI Data Types............................................................... 6-9

6.1.3 MMI_CreateMenuItem ...................................................... 6-10

6.1.4 MMI_CreateGBMenu ...................................................... 6-11

6.1.5 MMI_CreateGBMenuItem ................................................ 6-13

6.1.6 MMI_CreateGBMenuStr ................................................... 6-14

6.1.7 MMI_CreateGBMenuItemStr .......................................... 6-16

6.1.8 MMI_DeleteGBMenu ....................................................... 6-17

6.1.9 MMI_SelectGBMenuItem ................................................. 6-17

6.1.10 MMI_AddGBMenuButton .............................................. 6-18

6.1.11 MMI_CreateTextDialog ................................................ 6-19

6.1.12 MMI_CreateGraphDialog .............................................. 6-21

6.1.13 MMI_DeleteDialog ....................................................... 6-22

6.1.14 MMI_CheckButton ......................................................... 6-23

6.1.15 MMI_GetButton ............................................................ 6-24

6.1.16 MMI_AddButton ........................................................... 6-26

6.1.17 MMI_DeleteButton ....................................................... 6-27

6.1.18 MMI_PrintStr ............................................................... 6-28

6.1.19 MMI_PrintTok ............................................................. 6-29

6.1.20 MMI_PrintVal .............................................................. 6-30

6.1.21 MMI_PrintInt .............................................................. 6-32

6.1.22 MMI_InputStr .............................................................. 6-33

6.1.23 MMI_InputVal .............................................................. 6-35

6.1.24 MMI_InputInt .............................................................. 6-38

6.1.25 MMI_InputList ............................................................. 6-40

6.1.26 MMI_FormatVal ........................................................... 6-42

6.1.27 MMI_WriteMsg ............................................................ 6-44
6.1.28 MMI_WriteMsgStr ................................................................. 6-46
6.1.29 MMI_DrawLine ..................................................................... 6-48
6.1.30 MMI_DrawRect ..................................................................... 6-49
6.1.31 MMI_DrawCircle ................................................................. 6-51
6.1.32 MMI_DrawText ................................................................. 6-52
6.1.33 MMI_DrawBusyField ............................................................ 6-53
6.1.34 MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong ......... 6-54
6.1.35 MMI_StartVarBeep .............................................................. 6-54
6.1.36 MMI_SwitchVarBeep ............................................................ 6-55
6.1.37 MMI_GetVarBeepStatus ....................................................... 6-56
6.1.38 MMI_SwitchAFKey .............................................................. 6-57
6.1.39 MMI_SwitchIconsBeep .......................................................... 6-58
6.1.40 MMI_SetAngleRelation ....................................................... 6-59
6.1.41 MMI_GetAngleRelation ....................................................... 6-60
6.1.42 MMI_SetVAngleMode ......................................................... 6-60
6.1.43 MMI_GetVAngleMode ......................................................... 6-61
6.1.44 MMI_SetAngleUnit .............................................................. 6-61
6.1.45 MMI_GetAngleUnit .............................................................. 6-63
6.1.46 MMI_SetDistUnit ............................................................... 6-63
6.1.47 MMI_GetDistUnit ............................................................... 6-65
6.1.48 MMI_SetPressUnit ............................................................. 6-65
6.1.49 MMI_GetPressUnit ............................................................. 6-67
6.1.50 MMI_SetTempUnit ............................................................. 6-67
6.1.51 MMI_GetTempUnit ............................................................. 6-68
6.1.52 MMI_SetDateFormat .......................................................... 6-69
6.1.53 MMI_GetDateFormat .......................................................... 6-70
6.1.54 MMI_SetTimeFormat .......................................................... 6-70
6.1.55 MMI_GetTimeFormat .......................................................... 6-71
6.1.56 MMI_SetCoordOrder ......................................................... 6-72
6.1.57 MMI_GetCoordOrder .......................................................... 6-73
6.1.58 MMI_SetLanguage ............................................................. 6-73
6.1.59 MMI_GetLanguage ............................................................. 6-74
6.1.60 MMI_GetLangName ........................................................... 6-75
6.2 BASIC APPLICATIONS BAP ......................................................... 6-76
  6.2.1 Summarizing Lists of BAP Types and Procedures .......... 6-76
  6.2.2 BAP_SetAccessoriesDlg ...................................................... 6-77
  6.2.3 BAP_MeasDistAngle ......................................................... 6-77
  6.2.4 BAP_MeasRec ................................................................. 6-81
  6.2.5 BAP_FineAdjust .............................................................. 6-84
  6.2.6 BAP_SearchPrism ............................................................ 6-85
  6.2.7 BAP_SetManDist ............................................................. 6-86
  6.2.8 BAP_SetPpm ................................................................. 6-87
  6.2.9 BAP_SetPrism ............................................................... 6-88
  6.2.10 BAP_SetMeasPrg .......................................................... 6-88
  6.2.11 BAP_GetMeasPrg .......................................................... 6-90
  6.2.12 BAP_SetManDist .......................................................... 6-91
  6.2.13 BAP_SetHz ................................................................. 6-93

6.3 Measurement Functions TMC ............................................... 6-94
  6.3.1 Summarizing Lists of TMC Types and Procedures ........ 6-94
  6.3.2 TMC Data Structures ...................................................... 6-96
  6.3.3 TMC_DoMeasure ............................................................ 6-99
  6.3.4 TMC_GetPolar ............................................................... 6-101
  6.3.5 TMC_GetCoordinate ........................................................ 6-104
  6.3.6 TMC_GetAngle ............................................................. 6-107
  6.3.7 TMC_GetAngle_WInc ...................................................... 6-109
  6.3.8 TMC_QuickDist ............................................................ 6-110
  6.3.9 TMC_GetSimpleMea ......................................................... 6-114
  6.3.10 TMC_GetSetAngleFaceDef ............................................ 6-117
  6.3.11 TMC_GetSetHzOffset .................................................... 6-118
  6.3.12 TMC_GetSetDistPpm .................................................... 6-119
  6.3.13 TMC_SetDistPpm .......................................................... 6-122
  6.3.14 TMC_GetSetRefractiveCorr ........................................... 6-120
  6.3.15 TMC_GetSetRefractiveMethod ....................................... 6-120
  6.3.16 TMC_GetSetStation ..................................................... 6-121
  6.3.17 TMC_GetSetDistTapeMeasured ..................................... 6-121
  6.3.18 TMC_SetDistPpm .......................................................... 6-122
6.3 System Functions

6.3.19 TMC_SetDistSwitch ................................................................. 6-123
6.3.20 TMC_GetDistSwitch ............................................................... 6-123
6.3.21 TMC_SetOffsetDist ............................................................... 6-124
6.3.22 TMC_GetOffsetDist ............................................................... 6-125
6.3.23 TMC_IFOffsetDistMeasured ................................................... 6-125
6.3.24 TMC_GetFace1 ..................................................................... 6-126
6.3.25 TMC_SetAngSwitch ............................................................... 6-126
6.3.26 TMC_GetAngSwitch ............................................................... 6-127
6.3.27 TMC_SetInclineSwitch .......................................................... 6-127
6.3.28 TMC_GetInclineSwitch .......................................................... 6-128
6.3.29 TMC_GetInclineStatus ......................................................... 6-128

6.4 Functions for GSI .................................................................................... 6-130

6.4.1 Summarizing Lists of GSI Types and Procedures............... 6-130
6.4.2 Constants for WI values ............................................................ 6-132
6.4.3 Constants for Measurement Dialog Definition ..................... 6-135
6.4.4 Relationship of GSI_ID’s to GSI_PAR’s............................ 6-139
6.4.5 Data Structures for GSI Functions ............................................ 6-142
6.4.6 GSI_GetRunningNr ................................................................. 6-144
6.4.7 GSI_SetRunningNr ................................................................. 6-145
6.4.8 GSI_GetIndivNr ...................................................................... 6-145
6.4.9 GSI_SetIndivNr ...................................................................... 6-146
6.4.10 GSI_IsRunningNr ................................................................. 6-146
6.4.11 GSI_SetIVPtNrStatus ............................................................ 6-147
6.4.12 GSI_IncPNumber ................................................................. 6-148
6.4.13 GSI_Coding ......................................................................... 6-148
6.4.14 GSI_SelectCode ................................................................. 6-149
6.4.15 GSI_GetQCodeAvailable ..................................................... 6-149
6.4.16 GSI_SetQCodeMode ............................................................. 6-150
6.4.17 GSI_ExecQCoding ............................................................... 6-150
6.4.18 GSI_SetRecOrder ............................................................... 6-152
6.4.19 GSI_GetRecOrder ............................................................... 6-152
6.4.20 GSI_QuickSet ................................................................. 6-153
6.4.21 GSI_SetRecPath ................................................................. 6-153
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_GetRecPath</td>
<td>6-154</td>
</tr>
<tr>
<td>GSI_SetDataPath</td>
<td>6-155</td>
</tr>
<tr>
<td>GSI_GetDataPath</td>
<td>6-156</td>
</tr>
<tr>
<td>GSI_GetWiEntry</td>
<td>6-156</td>
</tr>
<tr>
<td>GSI_SetWiEntry</td>
<td>6-157</td>
</tr>
<tr>
<td>GSI_GetRecMask</td>
<td>6-158</td>
</tr>
<tr>
<td>GSI_SetRecMask</td>
<td>6-159</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>6-161</td>
</tr>
<tr>
<td>GSI_DefineRecMaskDlg</td>
<td>6-162</td>
</tr>
<tr>
<td>GSI_ManCoordDlg</td>
<td>6-162</td>
</tr>
<tr>
<td>GSI_ImportCoordDlg</td>
<td>6-165</td>
</tr>
<tr>
<td>GSI_SetLineSysMDlg</td>
<td>6-168</td>
</tr>
<tr>
<td>GSI_GetLineSysMDlg</td>
<td>6-169</td>
</tr>
<tr>
<td>GSI_SetMDlg</td>
<td>6-170</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>6-171</td>
</tr>
<tr>
<td>GSI_CreateMDlg</td>
<td>6-171</td>
</tr>
<tr>
<td>GSI_SetLineMDlg</td>
<td>6-173</td>
</tr>
<tr>
<td>GSI_GetLineMDlgText</td>
<td>6-174</td>
</tr>
<tr>
<td>GSI_SetLineMDlgPar</td>
<td>6-175</td>
</tr>
<tr>
<td>GSI_UpdateMDlg</td>
<td>6-177</td>
</tr>
<tr>
<td>GSI_DefineMDlg</td>
<td>6-178</td>
</tr>
<tr>
<td>GSI_UpdateMeasurement</td>
<td>6-178</td>
</tr>
<tr>
<td>GSI_Measure</td>
<td>6-179</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>6-180</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>6-180</td>
</tr>
<tr>
<td>GSI_RecordRecMask</td>
<td>6-181</td>
</tr>
<tr>
<td>CSV_GetDateTime</td>
<td>6-183</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>6-183</td>
</tr>
<tr>
<td>CSV_GetInstrumentName</td>
<td>6-185</td>
</tr>
<tr>
<td>CSV_GetDateTime</td>
<td>6-187</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>6-188</td>
</tr>
<tr>
<td>CSV_GetInstrumentName</td>
<td>6-188</td>
</tr>
</tbody>
</table>
6.5.6 CSV_GetInstrumentNo ............................................................. 6-189
6.5.7 CSV_GetInstrumentFamily ...................................................... 6-189
6.5.8 CSV_GetSWVersion ............................................................... 6-190
6.5.9 CSV_GetGBIVersion .............................................................. 6-191
6.5.10 CSV_GetElapsedSysTime ...................................................... 6-192
6.5.11 CSV_GetSysTime ................................................................. 6-193
6.5.12 CSV_GetLRSStatus ............................................................... 6-193
6.5.13 CSV_SetGuideLight .............................................................. 6-194
6.5.14 CSV_Laserpointer ................................................................. 6-195
6.5.15 CSV_MakePositioning .......................................................... 6-195
6.5.16 CSV_ChangeFace ................................................................. 6-196
6.5.17 CSV_SetLockStatus .............................................................. 6-197
6.5.18 CSV_GetLockStatus .............................................................. 6-198
6.5.19 CSV_LockIn .......................................................................... 6-199
6.5.20 CSV_LockOut ...................................................................... 6-200
6.5.21 CSV_SetATRStatus ............................................................... 6-200
6.5.22 CSV_GetATRStatus .............................................................. 6-201
6.5.23 CSV_Delay .......................................................................... 6-201
6.5.24 CSV_SetTargetType ............................................................. 6-202
6.5.25 CSV_GetTargetType ............................................................. 6-203
6.5.26 CSV_SetPrismType ............................................................... 6-204
6.5.27 CSV_GetPrismType ............................................................... 6-204
6.5.28 CSV_SetLaserPlummet ......................................................... 6-205
6.5.29 CSV_GetLaserPlummet ......................................................... 6-205
6.5.30 CSV_CheckAltUserTask ......................................................... 6-206
6.5.31 CSV_ResetAltUserTask .......................................................... 6-206
6.5.32 CSV_SysCall ........................................................................ 6-207
6.5.33 CSV_SysCallAvailable ......................................................... 6-208
6.5.34 CSV_LibCall ........................................................................ 6-209
6.5.35 CSV_LibCallAvailable .......................................................... 6-209
## 6.1 MMI FUNCTIONS

### 6.1.1 Summarising Lists of MMI Types and Procedures

#### 6.1.1.1 Types

<table>
<thead>
<tr>
<th>Type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListArray</td>
<td>List field Data structure</td>
</tr>
<tr>
<td>sLine</td>
<td>Display line</td>
</tr>
</tbody>
</table>

#### 6.1.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_AddButton</td>
<td>Add a Button to a dialog.</td>
</tr>
<tr>
<td>MMI_AddGBMenuButton</td>
<td>Adds a button to a menu</td>
</tr>
<tr>
<td>MMI_BeepAlarm</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_BeepLong</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_BeepNormal</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_CheckButton</td>
<td>Checks if a button was pressed.</td>
</tr>
<tr>
<td>MMI_CreateGBMenu</td>
<td>Creates a menu</td>
</tr>
<tr>
<td>MMI_CreateGBMenuItem</td>
<td>Creates an item to an existing menu</td>
</tr>
<tr>
<td>MMI_CreateGBMenuItem</td>
<td>Creates an item with a variable string</td>
</tr>
<tr>
<td>Str</td>
<td>Creates a menu with variable strings</td>
</tr>
<tr>
<td>MMI_CreateGraphDialog</td>
<td>Create and show a graphics dialog.</td>
</tr>
<tr>
<td>MMI_CreateMenuStr</td>
<td>Creates a menu item on the Theodolite menu.</td>
</tr>
<tr>
<td>MMI_CreateMenuItem</td>
<td>Create and show a text dialog.</td>
</tr>
<tr>
<td>MMI_DeleteButton</td>
<td>Delete a button from a dialog.</td>
</tr>
<tr>
<td>MMI_DeleteDialog</td>
<td>Deletes a dialog.</td>
</tr>
<tr>
<td>MMI_DeleteGBMenu</td>
<td>Deletes a menu</td>
</tr>
<tr>
<td>MMI_DrawBusyField</td>
<td>Shows or hides the Busy-Icon</td>
</tr>
<tr>
<td>MMI_DrawCircle</td>
<td>Draw a circle / ellipse.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>MMI_DrawLine</td>
<td>Draw a line.</td>
</tr>
<tr>
<td>MMI_DrawRect</td>
<td>Draw a rectangle.</td>
</tr>
<tr>
<td>MMI_DrawText</td>
<td>Draw / delete text.</td>
</tr>
<tr>
<td>MMI_FormatVal</td>
<td>Convert a value to a string.</td>
</tr>
<tr>
<td>MMI_GetAngleRelation</td>
<td>Request the current angle relationships.</td>
</tr>
<tr>
<td>MMI_GetAngleUnit</td>
<td>Return the currently displayed unit of angle.</td>
</tr>
<tr>
<td>MMI_GetButton</td>
<td>Get the button identifier of the pressed button.</td>
</tr>
<tr>
<td>MMI_GetCoordOrder</td>
<td>Retrieve the co-ordinate order.</td>
</tr>
<tr>
<td>MMI_GetDateFormat</td>
<td>Retrieves the date display format.</td>
</tr>
<tr>
<td>MMI_GetDistUnit</td>
<td>Return the currently displayed unit of distance.</td>
</tr>
<tr>
<td>MMI_GetLangName</td>
<td>Gets the name to a language number.</td>
</tr>
<tr>
<td>MMI_GetLanguage</td>
<td>Query the current language.</td>
</tr>
<tr>
<td>MMI_GetPressUnit</td>
<td>Return the currently displayed unit of pressure.</td>
</tr>
<tr>
<td>MMI_GetTempUnit</td>
<td>Return the currently displayed unit of temperature.</td>
</tr>
<tr>
<td>MMI_GetTimeFormat</td>
<td>This function retrieves the format used to display the time.</td>
</tr>
<tr>
<td>MMI_GetVAngleMode</td>
<td>Returns the V-Angle mode.</td>
</tr>
<tr>
<td>MMI_GetVarBeepStatus</td>
<td>Read the switch status for a variable signal beep.</td>
</tr>
<tr>
<td>MMI_InputInt</td>
<td>Get an integer input value in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputList</td>
<td>Shows a list field in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputStr</td>
<td>Get a string input in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputVal</td>
<td>Get a numerical input value in a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintInt</td>
<td>Print an integer value on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintStr</td>
<td>Print a string on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintTok</td>
<td>Print a token on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintVal</td>
<td>Print a value on a text dialog.</td>
</tr>
<tr>
<td>MMI_SelectGBMenu</td>
<td>Select a menu item.</td>
</tr>
<tr>
<td>MMI_SetAngleRelation</td>
<td>Set the angle relationship.</td>
</tr>
<tr>
<td>MMI_SetAngleUnit</td>
<td>Set the displayed unit of angle.</td>
</tr>
<tr>
<td>MMI_SetCoordOrder</td>
<td>Set the co-ordinate order.</td>
</tr>
</tbody>
</table>
procedure name | description
---|---
MMI_SetDateFormat | Set the date display format.
MMI_SetDistUnit | Set the displayed unit of distance.
MMI_SetLanguage | Set the display language.
MMI_SetPressUnit | Set the displayed unit of pressure.
MMI_SetTempUnit | Set the displayed unit of temperature.
MMI_SetTimeFormat | Set the time display format.
MMI_SetVAngleMode | Set the V-Angle mode.
MMI_StartVarBeep | Start beep sequences with configurable interrupts.
MMI_SwitchAFKey | Switch aF... key
MMI_SwitchIconsBeep | Switches measurement icons and special beeps
MMI_SwitchVarBeep | Switch a varying beep.
MMI_WriteMsg | Output to a message window. Parameter is a token.
MMI_WriteMsgStr | Output to a message window. Parameter is a string.

6.1.2  MMI Data Types

6.1.2.1  ListArray – List field data structure

**Description**  This array is used for list fields and consists of **LIST_ARRAY_MAX_ELEMENT** (200) elements of the type **STRING30**.

**Note**  Each variable of this data type reserves 6400 Bytes.

6.1.2.2  sLine – Display line

**Description**  This type is used to define a string with 29 characters, which is necessary to print variable strings on the display. The length depends on the actual display width, which is 29 for TPS1100 instruments.
6.1.3 MMI_CreateMenuItem

**Description**
Creates a system menu item on the Theodolite menu to establish the invocation of a GeoBASIC application.

**Declaration**

```pascal
MMI_CreateMenuItem(
  BYVAL sAppName AS String,
  BYVAL sFuncName AS String,
  BYVAL iMenuNum AS Integer,
  BYVAL sMenuText AS _Token )
```

**Remarks**
The `CreateMenuItem` creates a menu item in a system menu with the text `MenuText` on the chosen entry point `MenuNum` in the menu-system. By clicking the new menu item on the Theodolite, the subroutine with the name `FuncName` in the Program `AppName` will be executed. The number of applications which can be loaded at a time are limited to 25. The maximum number of entry points over all applications (C and GeoBASIC applications) is 50. All GLOBAL declared subroutines count as entry points. Be aware of the fact that the interpreter and a possible Coding function also count for the number of application. The same is true for any C-application which has been loaded onto the TPS.

**Note**
The subroutine denoted in `sFuncName` must be declared as GLOBAL.
The intended use for this procedure is during the installation phase only!

**Parameters**

- **sAppName** in The name of the program where the function or subroutine is defined.
- **sFuncName** in The name of the global function or subroutine to be called.
- **iMenuNum** in Defines in which menu the menu-entry is generated. There are three possible menus where a menu item can be added. For multiple menu items the menus can be combined with '+'-operator.
valid menus | meaning
--- | ---
MMI_MENU_PROGRAMS | Add to menu „Main menu“
MMI_MENU_PROGMENU | Add to „PROG“ - Key menu
MMI_MENU_AUTOEXEC | Add to menu „Autoexec“

sMenuText | The text of the menu-entry which should be displayed on the Theodolite.

Return-Codes

| RC_OK | Successful termination.

**Note**
Since this procedure will be called during installation phase you do not have the possibility to do any error handling. Only the loader will report an error which may be caused by an erroneous call.

**Example**
The example uses the MMI_CreateMenuItem routine to create a menu entry named "START THE PROGRAM" under the main menu. The function "Main" in the GeoBASIC program "ExampleProgram" will be called when this menu item is selected.

```vbnet
MMI_CreateMenuItem( "ExampleProgram", "Main", MMI_MENU_PROGRAMS, "START THE PROGRAM" )
```

### 6.1.4 MMI_CreateGBMenu

**Description**
Creates a menu.

**Declaration**
```vbnet
MMI_CreateGBMenu( 
    BYVAL sMenuName AS _Token, 
    iMenuId AS Integer 
)
```

**Remarks**
This routine creates an empty menu and the caption sMenuName. The function MMI_CreateGBMenuItem adds items to a menu.
Note: Before terminating a GeoBASIC program, all menus must be deleted.

The GeoBASIC menus system has the following limitations:

- The maximal number of menus for a GeoBASIC program is 5.
- The maximal number of items / menu is 49.
- The maximal number of items over all menus plus menus is 254.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sMenuName</td>
<td>The caption of the menu.</td>
</tr>
<tr>
<td>iMenuId</td>
<td>Returned menu identifier. It is the handle for using this menu.</td>
</tr>
</tbody>
</table>

Return-Codes

- RC_OK: Successful termination.
- MMI_NOMORE_MENUS: No more menus available.

See Also

- MMI_CreateGBMenuItem
- MMI_DeleteGBMenu
- MMI_SelectGBMenuItem
- MMI_AddGBMenuButton

Example

The example creates a menu with a button. For a complete example see sample program MENU.GBS

```plaintext
CONST MHELP = "Help for measurement type...."

DIM iMenu AS Integer ' menu identifier
DIM iSelection AS Integer ' selected item
DIM iButton AS Integer ' used button

'Create main menu
MMI_CreateGBMenu("MEASUREMENT TYPE", iMenu)
```
'Create menu items - all items use
  the same help text
MMI_CreateGBMenuItem(iMenu,
  "Polygon", MHELP)
MMI_CreateGBMenuItem(iMenu,
  "Border point", MHELP)
MMI_CreateGBMenuItem(iMenu,
  "Situation point", MHELP)

'Create the button supported in this menu
MMI_AddGBMenuButton(iMenu, MMI_F5_KEY, "EXIT ")

' show and execute menu
MMI_SelectGBMenuItem(iMenu, "TEST",
  iSelection, iButton)
SELECT CASE iSelection
  CASE 1 ' Polygon
  ...   CASE ELSE
    MMI_BeepAlarm()
  END SELECT
MMI_DeleteGBMenu(iMenu)

6.1.5 MMI_CreateGBMenuItem

Description  Creates an item in an existing menu.

Declaration  MMI_CreateGBMenuItem(
  BYVAL iMenuId AS Integer,
  BYVAL sMenuItemName AS _Token,
  BYVAL sHelpText AS _Token )

Remarks  This function adds one menu item to an existing menu iMenuId. This item will be displayed as the last item.

Parameters
  iMenuId  in  Menu identifier
  sMenuItemName  in  Displayed text
  sHelpText  in  Help text; only visible if the help functionality of theodolite is enabled

Return-Codes
6. System Functions

RC_OK  Successful termination.
BAS_MENU_ID_INVALID  Bad iMenuId
BAS_MENU_TABLE_FULL  No more free menu items

See Also  MMI_CreateGBMenu, MMI_DeleteGBMenu,
           MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example  see MMI_CreateGBMenu

6.1.6  MMI_CreateGBMenuStr

Description  Creates a menu with variable strings as menu name and menu items.

Declaration  

    MMI_CreateGBMenuStr( 
        BYVAL sMenuName AS sLine, 
        iMenuId AS Integer )

Remarks  This routine creates an empty menu and the caption sMenuName. sMenuName need not be constant, it can be generated during the execution of the program. The function MMI_CreateGBMenuItemStr adds items to this kind of menu.

Note  Before terminating a GeoBASIC program, all menus must be deleted.

    The GeoBASIC menus system has the following limitations:
    
    The maximal number of menus for a GeoBASIC program is 5.
    The maximal number of items / menu is 49.
    The maximal number of items over all menus plus menus is 254.

Parameters

    sMenuName in  The caption of the menu.
Returned menu identifier. It is the handle for using this menu.

Return-Codes

RC_OK
Successful termination.

MMI_NOMORE_MENUS
No more menus available

See Also

MMI_CreateGBMenuItemStr, MMI_DeleteGBMenu,
MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example

The example creates a menu with a button. The menu name is a composition with a constant string and the instrument name. The menu item names are extended with the current language name.

```
CONST MHELP = "Help for measurement type...."

DIM iMenu AS Integer ' menu identifier
DIM iSelection AS Integer ' selected item
DIM iButton AS Integer ' used button
DIM sMenuName AS sLine ' menu name
DIM sMenuItemName1 AS sLine ' menu item 1 name
DIM sMenuItemName2 AS sLine ' menu item 2 name
DIM iLangNr AS Integer ' language number
DIM sLangName AS String20 ' language name
DIM sInstrumentName AS String30 ' instrument name

' generate menu name
CSV_GetInstrumentName(sInstrumentName)
sMenuName = "Programs on " + sInstrumentName

' Create menu
MMI_CreateGBMenuStr(sMenuName, iMenu)

' generate menu item names
MMI_GetLanguage(iLangNr, sLangName)
sMenuItemName1 = "Polygon in " + sLangName
sMenuItemName2 = "Border point in " + sLangName

' Create menu items - all items use
' the same help text
MMI_CreateGBMenuItemStr(iMenu, sMenuItemName1, MHELP)
MMI_CreateGBMenuItemStr(iMenu, sMenuItemName2, MHELP)
```
'Create the button supported in this menu
MMI_AddGBMenuButton(iMenu, MMI_F5_KEY, "EXIT ")

' show and execute menu
MMI_SelectGBMenuItem(iMenu, "TEST", iSelection, iButton)
SELECT CASE iSelection
CASE 1 ' Polygon
...
CASE ELSE
    MMI_BeepAlarm()
END SELECT
MMI_DeleteGBMenu(iMenu)

6.1.7 MMI_CreateGBMenuItemStr

Description  Creates an item with a variable string in an existing menu.

Declaration  MMI_CreateGBMenuItemStr(
    BYVAL iMenuId AS Integer,
    BYVAL sMenuItemName AS sLine,
    BYVAL sHelpText AS _Token )

Remarks  This routine adds one menu item to an existing menu iMenuId. This item will be displayed as the last item. The menu must be created with MMI_CreateGBMenuStr. sMenuItemName need not be constant, it can be generated during the execution of the program.

Parameters  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMenuId</td>
<td>in Integer</td>
<td>Menu identifier</td>
</tr>
<tr>
<td>sMenuItemName</td>
<td>in sLine</td>
<td>Displayed text</td>
</tr>
<tr>
<td>sHelpText</td>
<td>in _Token</td>
<td>Help text; only visible if the help functionality of the theodolite is enabled</td>
</tr>
</tbody>
</table>

Return-Codes  

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_MENU_ID_INVALID</td>
<td>Bad iMenuId</td>
</tr>
</tbody>
</table>
6.1.8 MMI_DeleteGBMenu

Description
Deletes a menu.

Declaration
MMI_DeleteGBMenu( BYVAL iMenuId AS Integer )

Remarks
This function deletes the menu iMenuId.

Parameters
iMenuId in Menu identifier

Return-Codes
RC_OK Successful termination.
BAS_MENU_ID_INVALID Bad iMenuId

See Also
MMI_CreateGBMenu, MMI_CreateGBMenuItem, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example
see MMI_CreateGBMenu

6.1.9 MMI_SelectGBMenuItem

Description
Select a menu item.

Declaration
MMI_SelectGBMenuItem( BYVAL iMenuId AS Integer, BYVAL sCaptionLeft AS _Token, iSelItem AS Integer, iButtonId AS Integer )

Remarks
This function shows and executes a menu iMenuId and returns the selected item iSelItem or pressed button iButtonId.
Parameters

- **iMenuId** in: Menu identifier
- **sCaptionLeft** in: The maximal five-character long part of the title bar displayed left of the menu title, with a separation symbol.
- **iSelItem** in/out: Selected item
- **iButtonId** out: Pressed button

Return-Codes

- **RC_OK**: Successful termination.
- **BAS_MENU_ID_INVALID**: Bad iMenuId

See Also

- **MMI_CreateGBMenu**, **MMI_CreateGBMenuItem**, **MMI_DeleteGBMenu**, **MMI_AddGBMenuButton**

Example

- see **MMI_CreateGBMenu**

### 6.1.10 MMI_AddGBMenuButton

Description

Adds a button to a menu.

Declaration

```plaintext
MMI_AddGBMenuButton( 
    BYVAL iMenuId AS Integer, 
    BYVAL iButtonId AS Integer, 
    BYVAL sCaption AS _Token )
```

Remarks

This function adds a button with the identifier `iButtonId` to the menu `iMenuId` and shows the caption `sCaption`.
### Parameters

- **iMenuId in**
  - Menu identifier
  - Identifier of the button to be added.
  - Valid buttons are MMI_F1_KEY..MMI_F6_KEY and MMI_SHF2_KEY..MMI_SHF6_KEY.

- **iButtonId in**
  - Text placed onto the button (max. 5 characters)

- **sCaption in**
  - Text placed onto the button (max. 5 characters)

### Return-Codes

- **RC_OK**
  - Successful termination.
- **BAS_MENU_ID_INVALID**
  - Bad iMenuId

### See Also

- MMI_CreateGBMenu, MMI_CreateGBMenuItem, MMI_DeleteGBMenu, MMI_SelectGBMenuItem

### Example

see MMI_CreateGBMenu

---

### 6.1.11 MMI_CreateTextDialog

#### Description

Create and show a text dialog.

#### Declaration

```basic
MMI_CreateTextDialog(
    BYVAL iLines AS Integer,
    BYVAL sCaptionLeft AS _Token,
    BYVAL sCaptionRight AS _Token,
    BYVAL sHelpText AS _Token )
```

#### Remarks

The routine creates and shows a dialog with **iLines** lines, the left part of the title bar **sCaptionLeft**, the caption **sCaptionRight** and the help text **sHelpText**. Only one text dialog can exist at the same time. If **MMI_CreateTextDialog** is called while already a text dialog or a measurement dialog exists, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.
Only a text dialog or a measurement dialog is valid at a time. They cannot be defined at the same time. A graphic dialog overrides a text or measurement dialog but does not delete the definition of it.

On the dialog field strings, numerical values and list fields can be displayed or edited using the routines MMI_PrintStr, MMI_PrintVal, MMI_PrintInt, MMI_InputStr, MMI_InputVal, MMI_InputInt and MMI_InputList.

**Parameters**

- `iLines` in: The number of lines of the dialog. There are up to 12 lines possible. If the dialog has more than 6 lines, a scrollbar on the right side appear and it is possible to scroll up and down with the cursor keys.

- `sCaptionLeft` in: The maximal five-character long part of the title bar displayed left of the `CaptionRight`, with a separation symbol.

- `sCaptionRight` in: The caption of the dialog.

- `sHelpText` in: This text is shown, when the help button `SHIFT-F1` is pressed and the help functionality of the theodolite is enabled.

**Return-Codes**

- `RC_OK`: Successful termination.

**See Also**

- MMI_DeleteDialog, MMI_CreateGraphDialog, GSI_CreateMDlg, MMI_PrintVal, MMI_PrintStr, MMI_PrintTok, MMI_PrintInt, MMI_InputVal, MMI_InputStr, MMI_InputInt, MMI_InputList
Example   The example uses the MMI_CreateTextDialog routine to create and display a text dialog.

Define a help text containing the inverse written word "Help"

CONST Helptext = MMI_INVERSE_ON + "Help" + MMI_INVERSE_OFF + " Test"

MMI_CreateTextDialog(5, "TEXT", "DIALOG CAPTION", Helptext)

6.1.12 MMI_CreateGraphDialog

Description   Create and show a graphics dialog.

Declaration   MMI_CreateGraphDialog(
    BYVAL sCaptionLeft AS _Token,
    BYVAL sCaptionRight AS _Token,
    BYVAL sHelptext AS _Token )

Remarks   The routine creates and shows a graphics dialog filled with the left part of the title bar sCaptionLeft, the caption sCaptionRight and the help text sHelptext for later use of MMI graphics functions. The size of the field is the whole dialog display area = 232 x 48 pixels. Only one graphics dialog can exist at the same time. If CreateGraphDialog is called while already a graphics dialog exists, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

Note   Only a text dialog or a measurement dialog is valid at a time. They cannot be defined at the same time. A graphic dialog overrides a text or measurement dialog but does not delete the definition of it.
Parameters

- **sCaptionLeft** in
  - The maximal five-character long part of the title bar displayed left of the `sCaptionRight`, with a separation symbol.

- **sCaptionRight** in
  - The caption of the dialog.

- **sHelpText** in
  - This text is shown, when the help button Shift-F1 is pressed and the help functionality of the theodolite is enabled.

Return-Codes

- **RC_OK**
  - Successful termination.

See Also

- MMI_DeleteDialog, MMI_CreateTextDialog, GSI_CreateMDlg, MMI Graphic Functions

Example

The example uses the MMI_CreateGraphDialog routine to create and display a graphic dialog field.

```
MMI_CreateGraphDialog( "GRAPH",
                      "DIALOG CAPTION",
                      "This is a help text")
```

6.1.13 MMI_DeleteDialog

Description

Deletes a dialog.

Declaration

`MMI_DeleteDialog()`

Remarks

The routine deletes the currently active dialog. It makes no distinction between graphic, measure and text dialog. By deleting the dialog all user defined buttons added with `MMI_AddButton` are deleted as well.

Return-Codes

- **RC_OK**
  - Successful termination.

- **BAS_NO_DLG_EXIST**
  - No dialog exists for this operation.

See Also

- MMI_CreateTextDialog, MMI_CreateGraphDialog, GSI_CreateMDlg
Example
The example uses the MMI_DeleteDialog routine to delete a
text, measure or graphic dialog.

MMI_DeleteDialog();

6.1.14 MMI_CheckButton

Description
Checks if a button was pressed.

Declaration
MMI_CheckButton( lKeyPressed AS Logical )

Remarks
The routine MMI_CheckButton checks the keyboard buffer for
pressed buttons. If a button was pressed, the routine returns
KeyPressed = TRUE, otherwise KeyPressed = FALSE is
returned.

Note
The routine MMI_CheckButton does not wait until a
button was pressed. It only checks the keyboard buffer.

Parameters

KeyPressed In

KeyPressed = TRUE is returned,
if a valid button was pressed.
Otherwise the value of
KeyPressed is FALSE.

Return-Codes

RC_OK
Successful termination.

BAS_NO_DLG_EXIST
No dialog exists for this operation.

See Also
MMI_AddButton
MMI_GetButton
Example
The example uses the MMI_CheckButton routine to wait until a (valid) key was pressed.

```basic
DIM lKeyPressed AS Logical
DO
  MMI_CheckButton( lKeyPressed )
LOOP UNTIL lKeyPressed
'do something ..
```

### 6.1.15 MMI_GetButton

**Description**
Get the button identifier of the pressed button.

**Declaration**

```basic
MMI_GetButton( iButtonId AS Integer,
               BYVAL lAllKeys AS Logical )
```

**Remarks**
Waits until a valid key is pressed and returns the button Identifier `iButtonId` of the pressed button.
If `lAllKeys = FALSE`, the keys ESC, ENTER, ON/OFF or any assigned button (added with `MMI_AddButton`) terminates this function and the `iButtonId` of the pressed button is returned. If `lAllKeys = TRUE`, additional keys i.e. the cursor keys terminates this routine too. For details see table below.

| Note | This function relates to the currently active dialog. |

**Parameters**

- `iButtonId` Out: The identifier of the pressed button. For values of `iButtonId` see the table below.
- `lAllKeys` In: Determines which keys exit the routine. If `lAllKeys = TRUE` any valid pressed key exit the routine, otherwise only normal ones.
### Button pressed | iButtonId returned
---|---
Assigned (using MMI_AddButton) | MMI_F1_KEY, MMI_F6_KEY, MMI_SHF2_KEY, MMI_SHF6_KEY
unassigned | MMI_UNASS_KEY
Assigned "CODE" | MMI_CODE_KEY
unassigned "CODE" | MMI_UNASS_KEY
"ENTER" within dialog, focus on a field | MMI_UNASS_KEY
"ENTER" within dialog, no focus | MMI_UNASS_KEY
"ENTER" after editing | MMI_EDIT_ENTER_KEY
"ESC" within dialog | MMI_ESC_KEY
"ESC" after editing | MMI_EDIT_ESC_KEY
"SHIFT" | MMI_UNASS_KEY
"0".."9", focus on spin/list-field | MMI_NUM0_KEY, MMI_NUM9_KEY
"0".."9", no focus | MMI_NUM0_KEY, MMI_NUM9_KEY
"CE" | MMI_UNASS_KEY
**Cursor Keys** | MMI_UP_KEY, MMI_DOWN_KEY, MMI_RIGHT_KEY, MMI_LEFT_KEY

### Return-Codes
- **RC_OK**: Successful termination.
- **BAS_NO_DLG_EXIST**: No dialog exists for this operation.

**See Also**: MMI_AddButton, MMI_CheckButton
Example

The example uses the MMI_GetButton routine to react to a pressed button. To make a function key valid for MMI_GetButton it must be added to the dialog (with MMI_AddButton).

```basic
DIM iActionButton AS Integer
DIM iPressedButton AS Integer

iActionButton = MMI_F2_KEY

MMI_GetButton ( iPressedButton, TRUE )
IF iPressedButton = iActionButton THEN
    'any actions
END IF
```

6.1.16 MMI_AddButton

Description
Add a button to a dialog.

Declaration
```basic
MMI_AddButton( BYVAL iButtonId AS Integer,
    BYVAL sCaption AS _Token )
```

Remarks
The routine MMI_AddButton adds the button with the Identifier iButtonId to the actual dialog and places the text sCaption onto the button. These added buttons are valid for the routines MMI_CheckButton and MMI_GetButton and the input routines (MMI_InputStr, MMI_InputVal, MMI_InputInt and MMI_InputList) which means the according button identifier can be returned from these routines.

Note
Either a text dialog or a measurement dialog can be defined at a time. Additionally a graphics dialog can override one of these above. Then the functionality applies to the graphics dialog.

The added buttons can be deleted with the routine MMI_DeleteButton while the dialog exists. Closing the dialog with MMI_DeleteDialog deletes all buttons attached to this dialog.
Parameters

ibuttonid in Identifier of the button to be added. See for the values that can be used for the ibuttonid under the routine description MMI_GetButton. Only MMI_F1_Key..MMI_F5_KEY, MMI_SHF2_KEY..MMI_SHF6_KEY and MMI_CODE_KEY are available for the AddButton routine.

sCaption in The text placed onto the button, left alignment (max. 5 characters).

Return-Codes

RC_OK Successful termination.
BAS_NO_DLG_EXIST No dialog exists for this operation.
MMI_BUTTON_ID_EXISTS This button has been defined already.

See Also MMI_GetButton, MMI_CheckButton, MMI_DeleteButton

Example The example uses the MMI_AddButton routine to add the F2-KEY with the caption "EXIT" to the dialog.

MMI_AddButton( MMI_F2_KEY, "$EXIT" )
Parameters

iButtonId in  Identifier of the button to be deleted. See for the values that can be used for iButtonId under the routine description MMI_GetButton.

Return-Codes

RC_OK  Successful termination.
BAS_NO_DLG_EXIST  No dialog exists for this operation.
MMI_ILLEGAL_BUTTON_ID  This button has not been defined by MMI_AddButton.

See Also  MMI_AddButton

Example

The example uses the MMI_DeleteButton routine to delete the F2-KEY from the dialog.

MMI_DeleteButton( MMI_F2_KEY )

6.1.18 MMI_PrintStr

Description  Print a string on a text dialog.

Declaration  

MMI_PrintStr( BYVAL iColumn AS Integer,
BYVAL iLine AS Integer,
BYVAL sText AS String30,
BYVAL lValid AS Logical )

Remarks  The text string sText is placed on position iColumn and iLine on the text dialog. If lValid is not TRUE, then the symbols for invalid values are displayed. Too long text strings are truncated, illegal co-ordinates are adjusted.

Note  A text dialog must already exist. Only display length number of character will be displayed, hence 29.

Parameters

iColumn in  The horizontal position (0..28)
6.1.19 MMI_PrintTok

Description
Print a string on a text dialog.

Declaration
MMI_PrintTok(  BYVAL iColumn AS Integer,  
                BYVAL iLine AS Integer,  
                BYVAL sText AS _Token )

Remarks
The text token sText is placed on position iColumn and  
iLine on the text dialog. Too long text strings are truncated,  
illegal co-ordinates are adjusted. This routine may be used instead  
of MMI_PrintStr to support internationalisation of multiple  
language applications.

Note  A text dialog must already exist.

Parameters
iColumn  in  The horizontal position (0..28)
iLine  in  The vertical position (0..number of lines  
defined with MMI_CreateTextDialog)
sText in The text string to display

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
<tr>
<td>TXT_UNDEF_TOKEN</td>
<td>The given token could not be found in the database. Most probably an old version is loaded either on TPS or simulator.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>No text token database is loaded with the currently set language.</td>
</tr>
</tbody>
</table>

See Also MMI_PrintStr

Example The example uses the MMI_PrintTok routine to print the text string „Hello World” in the first line on row 2 of the actual text dialog:

MMI_PrintTok( 2, 0, "Hello World" )

6.1.20 MMI_PrintVal

Description Print a value on a text dialog.

Declaration

```
MMI_PrintVal( BYVAL iColumn AS Integer, 
              BYVAL iLine AS Integer, 
              BYVAL iLen AS Integer, 
              BYVAL iDecimals AS Integer, 
              BYVAL dVal AS Double, 
              BYVAL lValid AS Logical, 
              BYVAL iMode AS Integer )
```

Remarks This routine can be used to display double values (or values with equal type, e.g. dimension). If lValid = TRUE the value dVal is placed on position iColumn and iLine on the text dialog, otherwise the symbols for invalid values "-----" are displayed. Too long value strings are truncated, illegal co-ordinates are adjusted. If iMode = MMI_DIM_ON, a dimension field is automatically displayed when the type of dVal has units.
If the \textit{dVal} cannot be displayed in \textit{iLen} characters, then "xxx" will be displayed instead.

**Note**  A text dialog must already exist.

**Parameters**

\begin{itemize}
\item \textit{iColumn} \textbf{in}  The horizontal position (0..28).
\item \textit{iLine} \textbf{in}  The vertical position (0..number of lines defined with \texttt{CreateTextDialog}).
\item \textit{iLen} \textbf{in}  The length of the value consisting of a sign, the characters before and after the comma and the comma itself. The dimension field is not included.
\item \textit{iDecimals} \textbf{in}  The number of decimals. If \textit{iDecimals} = -1 then the number of decimals set by the system is taken.
\item \textit{dVal} \textbf{in}  The value to display. Use this routine to display double (and equal to double) values with the correct units. For integer values a separate routine (\texttt{MMI_PrintInt}) exists.
\item \textit{lValid} \textbf{in}  Determines if the value should be shown as valid. If \textit{lValid} = \texttt{TRUE} the value \textit{dVal} is displayed, otherwise the symbols for invalid values are displayed.
\item \textit{iMode} \textbf{in}  Determines the display of the dimension. If \textit{iMode} = \texttt{MMI_DIM_ON} a dimension field is automatically displayed when the type \textit{dVal} has units. Otherwise use \texttt{MMI_DEFAULT_MODE}.
\end{itemize}

**Return-Codes**

\begin{itemize}
\item \texttt{RC_OK}  Successful termination.
\item \texttt{BAS_NO_DLG_EXIST}  No dialog exists for this operation.
\end{itemize}

**See Also**

\texttt{MMI_PrintInt, MMI_InputVal}

**Example**

The example uses the \texttt{MMI_PrintVal} routine to print the value of \textit{TestVal} as distance (with corresponding dimension) in the first line on row 2 of the currently open text dialog.
DIM TestVal AS Distance
TestVal = 287.47

MMI_PrintVal( 2, 0, 10, 2, TestVal, TRUE, 
               MMI_DIM_ON )

6.1.21 MMI_PrintInt

Description
Print an integer value on a text dialog.

Declaration
MMI_PrintInt( BYVAL iColumn AS Integer,
               BYVAL iLine AS Integer,
               BYVAL iLen AS Integer,
               BYVAL iVal AS Integer,
               BYVAL lValid AS Logical )

Remarks
This routine can be used to display integer values. Too long value strings are truncated, illegal co-ordinates are adjusted. If lValid = TRUE the value iVal is placed on position iColumn and iLine on the text dialog, otherwise the symbols for invalid values are displayed. If the iVal can not be displayed in iLen characters, then "xxx" will be displayed instead.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn</td>
<td>The horizontal position (0..28).</td>
</tr>
<tr>
<td>iLine</td>
<td>The vertical position (0..number of lines defined with MMI_CreateTextDialog).</td>
</tr>
<tr>
<td>iLen</td>
<td>The length of the value plus the sign.</td>
</tr>
<tr>
<td>iVal</td>
<td>The value to display. Use this routine to display integer values. For double values a separate routine (MMI_PrintVal) exists.</td>
</tr>
<tr>
<td>lValid</td>
<td>Determines if the value should be shown as valid. If lValid = TRUE the value iVal is displayed, otherwise the symbols for invalid values are displayed.</td>
</tr>
</tbody>
</table>

Note: A text dialog must already exist.
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

See Also

MMI_PrintVal
MMI_InputInt

Example

The example uses the MMI_PrintInt routine to print the value of TestVal in the first line on row 2 of the currently open text dialog.

```GeoBASIC
DIM TestVal AS Integer
TestVal = 1000
MMI_PrintInt( 2, 0, 5, TestVal, TRUE )
```

6.1.22 MMI_InputStr

Description

Get a string input in a text dialog.

Declaration

```GeoBASIC
MMI_InputStr( BYVAL iColumn AS Integer, 
  BYVAL iLine AS Integer, 
  BYVAL iLen AS Integer, 
  BYVAL iMode AS Integer, 
  sText AS String30, 
  lValid AS Logical, 
  iButtonId AS Integer )
```

Remarks

If lValid = TRUE the text string sText is placed on position iColumn and iLine on the text dialog, otherwise the symbols for invalid values are displayed. Illegal co-ordinates are adjusted. If the length of the string exceeds the given length iLen the string is truncated at position iLen. After the edit process the string is returned and the text is placed right aligned on the display. If the length iLen <= 0 or no part of the field is in the dialog area the Text is not edited and the routine exits.

The string can be edited by pressing αEDIT or a numerical key. If iMode = MMI_DEFAULT_MODE the keys ESC, ENTER,
ON/OFF or any user defined button (added with MMI_AddButton) terminates the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputStr too. For details see MMI_GetButton.

**Note** A text dialog must already exist.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn in</td>
<td>The horizontal position (0..28).</td>
</tr>
<tr>
<td>iLine in</td>
<td>The vertical position (0..number of lines defined with MMI_CreateTextDialog).</td>
</tr>
<tr>
<td>iLen in</td>
<td>The length of the input field.</td>
</tr>
<tr>
<td>iMode in</td>
<td>Defines the editing mode.</td>
</tr>
<tr>
<td>sText inout</td>
<td>The text string to edit.</td>
</tr>
<tr>
<td>lValid inout</td>
<td>Determines if the value should be shown as valid.</td>
</tr>
<tr>
<td></td>
<td>If lValid=TRUE the string sText is displayed, otherwise the symbols for invalid values are displayed.</td>
</tr>
<tr>
<td>iButtonId out</td>
<td>The identifier of the pressed valid button to exit the edit process.</td>
</tr>
</tbody>
</table>

### Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

### See Also

- MMI_PrintStr

---

6-34 TPS1100-Version 2.10
Example

The example uses the `MMI_InputStr` routine to get the text string `sInputString` in the first line on row 2 of the actual text dialog.

```plaintext
DIM sInputString AS String30
DIM iButton AS Integer
DIM lValid AS Logical

sInputString = "The input text"
lValid = TRUE
MMI_InputStr( 2, 0, 20, MMI_DEFAULT_MODE,
               sInputString, lValid,iButton )
```

6.1.23 **MMI_InputVal**

**Description**

Get a numerical input for double values in a text dialog.

**Declaration**

```plaintext
MMI_InputVal( BYVAL iColumn AS Integer,
              BYVAL iLine AS Integer,
              BYVAL iLen AS Integer,
              BYVAL iDecimals AS Integer,
              BYVAL dMin AS Double,
              BYVAL dMax AS Double,
              BYVAL iMode AS Integer,
              dVal AS Double,
              lValid AS Logical,
              iButtonId AS Integer )
```

**Remarks**

If `lValid = TRUE` then the value `dVal` is placed on position `iColumn` and `iLine` on the text dialog, otherwise the symbols for invalid values are displayed. Illegal co-ordinates are adjusted. If `iMode = MMI_DIM_ON`, a dimension field is automatically displayed when the type of `dVal` has units. If the length `iLen` <= 0 or no part of the field is in the dialog area the value is not edited and the routine exits.

The value within the bounds `dMin` and `dMax` can be edited by pressing EDIT or the numerical block keys. If `iMode = MMI_DEFAULT_MODE` the keys ESC, ENTER, ON/OFF or any user defined button (added with `MMI_AddButton`) terminates the routine.
the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputVal too. For details see MMI_GetButton.

**Note** A text dialog must already exist.

### Parameters

- **iColumn in** The horizontal position (0..28).
- **iLine in** The vertical position (0..number of lines defined with MMI_CreateTextDialog).
- **iLen in** The length of the value inclusive decimals, sign and the comma, exclusive the dimension field.
- **iDecimals in** The number of decimals. If iDecimals = -1 the number of decimals set by the system is taken.
- **dMin in** The lower and upper bounds.
- **dMax**
- **iMode in** Defines the editing mode.
  - MMI_DEFAULT_MODE defines normal editing
  - MMI_SPECIALKEYS_ON allows editing with full cursor control
  - MMI_DIM_ON shows a dimension field if dVal has units.
  - Modes can be added, i.e. MMI_SPECIALKEYS_ON + MMI_DIM_ON

- **dVal inout** The value to edit. Use this routine to edit double (and equal to double) values. For integer values a separate routine (MMI_InputInt) exists.
lValid inout Determines if the value should be shown as valid. If lValid=TRUE the value dVal is displayed, otherwise the symbols for invalid values are displayed.

iButtonId out The identifier of the pressed valid button to exit the edit process.

Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

See Also

- MMI_InputInt
- MMI_PrintVal

Example

See example file „cursor.gbs“ too.

The example uses the MMI_InputVal routine to get the distance of TestVal with default decimal places. Input field is placed in the second line on row 2 of the actual text dialog. The entered values must lie in the range 0..1000.

```plaintext
CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM TestVal AS Distance
DIM iButton AS Integer
DIM lValid AS Logical

lValid = FALSE

MMI_InputVal( 2, 1, 8, -1, 0, 1000, MODE, TestVal, lValid, iButton )
```
6.1.24 MMI_InputInt

**Description**
Get an integer input value in a text dialog.

**Declaration**

```
MMI_InputInt( BYVAL iColumn AS Integer, 
BYVAL iLine AS Integer, 
BYVAL iLen AS Integer, 
BYVAL iMin AS Integer, 
BYVAL iMax AS Integer, 
BYVAL iMode AS Integer, 
iVal AS Integer, 
lValid AS Logical, 
iButtonId AS Integer )
```

**Remarks**
If lValid = TRUE then the integer value iVal is placed on position iColumn and iLine on the text dialog. Illegal co-ordinates are adjusted. If the length iLen ≤ 0 or no part of the field is in the dialog area the value is not edited and the routine exits.

The integer value within the bounds iMin and iMax can be edited by pressing EDIT or the numerical block keys. If iMode = MMI_DEFAULT_MODE the keys ESC, ENTER, ON/OFF or any user defined button (added with MMI_AddButton) terminates the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputInt too.

**Note**
A text dialog must already exist.

**Parameters**

- **iColumn** in
  The horizontal position (0..28).
- **iLine** in
  The vertical position (0..number of lines defined with MMI_CreateTextDialog).
- **iLen** in
  The length of the value plus the sign.
- **iMin** in
  The lower and upper bounds.
- **iMax**
iMode in Defines the editing mode.

MMI_DEFAULT_MODE defines normal editing

MMI_SPECIALKEYS_ON allows editing with full cursor control

iVal inout The value to display. Use this routine to edit integer values. For double values a separate routine (MMI_InputVal) exists.

lValid inout Determines if the value should be shown as valid. If lValid=TRUE the value iVal is displayed, otherwise the symbols for invalid values are displayed.

iButtonId out The identifier of the pressed valid button to exit the edit process.

Return-Codes

RC_OK Successful termination.

BAS_NO_DLG_EXIST No dialog exists for this operation.

See Also

MMI_PrintInt, MMI_InputVal

Example

See example file "cursor.gbs" too.

The example uses the MMI_InputInt routine to get the value of iTestVal in the second line on row 2 of the actual text dialog. The entered values must lie in the range 0..1000.

CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM iTestVal AS Integer
DIM iButton AS Integer
DIM lValid AS Logical

lValid = FALSE
MMI_InputInt( 2,1,5,0,1000,
    MODE,iTestVal,lValid,iButton )
6.1.25 MMI_InputList

Description
Shows a list field in a text dialog.

Declaration
```
MMI_InputList( BYVAL iColumn AS Integer,
              BYVAL iLine AS Integer,
              BYVAL iLen AS Integer,
              BYVAL iElements AS Integer,
              BYVAL iMode AS Integer,
              List AS ListArray,
              iIndex AS Integer,
              lValid AS Logical,
              iButtonId AS Integer )
```

Remarks
If lValid = TRUE then a list field is placed on position iColumn and iLine on the text dialog. Too long list elements are truncated, illegal co-ordinates are adjusted. The ListArray is an array of String30 with LIST_ARRAY_MAX_ELEMENT Elements. Only the first iElements are displayed. The value of iIndex defines which element is shown first.

The list can be edited by pressing F6 (LIST). With the cursor keys UP and DOWN a field element can be selected. If the list elements are numbered (begins with a number), then the elements can be selected directly by pressing numerical buttons. If iMode = MMI_DEFAULT_MODE the keys ESC, ENTER, ON/OFF or any user defined button (added with MMI_AddButton) terminates the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputList too.

Note
A text dialog must already exist.

Parameters
- iColumn in: The horizontal position (0..28).
- iLine in: The vertical position (0..number of lines defined with MMI_CreateTextDialog).
- iLen in: The displayed length of the list elements.
### GeoBASIC Reference Manual

#### 6. System Functions

**iElements in**

The number of list elements. The maximum number is limited to LIST_ARRAY_MAX_ELEMENT.

**iMode in**

Defines the editing mode.

- MMI_DEFAULT_MODE defines normal editing
- MMI_SPECIALKEYS_ON allows editing with full cursor control

**List in**

The array of the list elements.

**iIndex in**

Index (number of the line) of the first shown and selected field respectively. Possible value for iIndex are in the range of 1 up to Elements.

**lValid inout**

Determines if the value should be shown as valid. If lValid=TRUE the value is displayed, otherwise the symbols for invalid values are displayed.

**iButtonId out**

The identifier of the pressed valid button to exit the list process.

**Return-Codes**

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

**Example**

See example file "cursor.gbs" too.

The example uses the MMI_InputList routine to get the value of the selected list element (the selected line) of a list field displayed in the second line on row 2 of the actual text dialog. The first displayed line is the line with the number Index.
CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM iLen AS Integer
DIM iElements AS Integer
DIM List AS ListArray
DIM iIndex AS Integer
DIM iButton AS Integer
DIM lValid AS Logical

'initialize the variables
iLen = 10 'displayed length of the list
iElements = 7 'number of available fields
iIndex = 3 'number of the first shown list element
lValid = TRUE

List(1) = "1 Line No.: 1"
List(2) = "2 Line No.: 2"
List(3) = "3 Line No.: 3"
List(4) = "4 Line No.: 4"
List(5) = "5 Line No.: 5"
List(6) = "6 Line No.: 6"
List(7) = "7 Line No.: 7"

InputList( 5, 1, iLen, iElements, MODE,
            List, iIndex, lValid, iButton )

6.1.26 MMI_FormatVal

**Description**  Convert a value to a string and use TPS system formatting rules.

**Declaration**  

```
MMI_FormatVal( BYVAL iType AS Integer,
               BYVAL iLen AS Integer,
               BYVAL iDecimals AS Integer,
               BYVAL dVal AS Double,
               BYVAL lValid AS Logical,
               BYVAL iMode AS Integer,
               sValStr AS String30 )
```

**Remarks**  If lValid = TRUE then this routine converts a double value (or values with equal type, e.g. dimension) to a text string, otherwise the symbols for invalid values are returned. The returned string...
sValStr contains the value string in the same kind as it would be displayed on the Theodolite: the value is placed right aligned with the number iDecimals of decimals. If iMode = MMI_DIM_ON, a dimension field is appended to the output string when the type iType allows it. If the dVal can not be displayed in iLen characters, then "xxx" will be returned instead.

This routine is useful, if numeric values should be written on files (see chapter file handling for further information).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTypen</td>
<td>The type of the numerical field. The type defines if a dimension field is available. Following values for the type can be used:</td>
</tr>
<tr>
<td>Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>MMI_FFORMAT_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>MMI_FFORMAT_DISTANCE</td>
<td>distance</td>
</tr>
<tr>
<td>MMI_FFORMAT_SUBDISTANCE</td>
<td>sub-distance [mm]</td>
</tr>
<tr>
<td>MMI_FFORMAT_ANGLE</td>
<td>angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_VANGLE</td>
<td>vertical angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_HZANGLE</td>
<td>horizontal angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_TEMPERATURE</td>
<td>temperature</td>
</tr>
<tr>
<td>MMI_FFORMAT_TIME</td>
<td>time 12h/24h-format</td>
</tr>
<tr>
<td>MMI_FFORMAT_DATE</td>
<td>date</td>
</tr>
<tr>
<td>MMI_FFORMAT_DATE_TIME</td>
<td>date/time</td>
</tr>
</tbody>
</table>

iLen in The length of the value consisting of a sign, the characters before and after the comma and the comma itself. The dimension field is not included.

iDecimals in The number of decimals. If iDecimals = -1 the number of decimals set by the system is taken.
dVal  in  The value to convert. Use this routine to convert double (and equal to double) values.

iMode  in  If iMode = MMI_DIM_ON a dimension string is automatically added to sValStr when the type dVal has units. Otherwise use MMI_DEFAULT_MODE.

sValStr  out  sValStr contains the string representation of the value dVal.

Return-Codes

RC_OK  Successful termination.

RC_IVRESULT  The result is not valid due to an illegal input value.

See Also  sFormatVal

Example  The example uses the MMI_FormatVal routine to convert the value dTestVal as distance (with corresponding dimension).

```
DIM dTestVal AS Distance
DIM sVString AS String30

dTestVal = 287.47

MMI_FormatVal( MMI_FFORMAT_DISTANCE, 10, -1, dTestVal, TRUE, MMI_DIM_ON, sVString )
```

6.1.27 MMI_WriteMsg

Description  Output to a message window.

Declaration  MMI_WriteMsg( BYVAL sText AS _Token,  
                          BYVAL sCaption AS _Token,  
                          BYVAL iMsgType AS Integer,  
                          iRetKey AS Integer )

Remarks  The function opens a message window on the display, which shows the text specified by sText. Lines that are too long to fit into the window are split automatically.
sText may contain a carriage return (character code 10) which
breaks a line explicitly. The predefined constants
MMI_INVERSE_ON and MMI_INVERSE_OFF can be used for
inverse text.
Text lines, that exceed the size of the window, are not displayed.
A title text, which will be printed on the first line of the message
box, can be set with sCaption, which may not be longer than
one line and contain neither font attributes nor type information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>Text-token to be displayed on the window (on the Theodolite).</td>
</tr>
<tr>
<td>sCaption</td>
<td>Text-token that will be displayed as title of the window.</td>
</tr>
</tbody>
</table>
| iMsgType   | Defines the type of the message window to be displayed, with the corresponding
text on the buttons; possible types:
   MMI_MB_OK
   MMI_MB_ABORT
   MMI_MB_OK_ABORT
   MMI_MB_ABORT_RETRY_CONT
   MMI_MB_YES_NO_ABORT
   MMI_MB_YES_NO
   MMI_MB_RETRY_ABORT
   MMI_MB_ABORT_CONT
   MMI_MB_ABORT_RETRY_IGNORE
   MMI_MB_ABORT_IGNORE
| iRetKey    | Returns the button pressed, i.e.
iRetKey:                               |
|            | MMI_MB_RET_OK
            | MMI_MB_RET_ABORT
            | MMI_MB_RET_OK_ABORT
            | MMI_MB_RET_RETRY_CONT
            | MMI_MB_RET_YES
            | MMI_MB_RET_NO
            | MMI_MB_RETugar
            | MMI_MB_RET_ABORT_IGNORE

TPS1100-Version 2.10  6-45
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

Example

The example uses the MMI_WriteMsg routine to display a message box with the title text "Warning" and the text "timeout" and shows the buttons "Retry", "Abort" returning the button-id in iRetKey.

```
MMI_WriteMsg( "Warning", "timeout", 
              MMI_MB_RETRY_ABORT, iMBRetKey )
```

6.1.28 MMI_WriteMsgStr

Description

Output to a message window.

Declaration

```
MMI_WriteMsgStr( BYVAL sText AS String255, 
                 BYVAL sCaption AS _Token, 
                 BYVAL iMsgType AS Integer, 
                 iRetKey AS Integer )
```

Remarks

The function opens a message window on the display, which shows the text specified by sText. Lines, which are too long to fit into the window, are split automatically. sText may contain a carriage return (character code 10) which breaks a line explicitly. The predefined constants MMI_INVERSE_ON and MMI_INVERSE_OFF can be used for inverse text. Text lines, that exceed the size of the window, are not displayed. A title text, which will be printed on the first line of the message box, can be set with sCaption, which may not be longer than one line and contain neither font attributes nor type information.

Note

This routine is different to MMI_WriteMsg in such a way that sText may be computed. But, of course, sText will not be entered into the text token data base.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sText</td>
<td>Text string to be displayed in a message box.</td>
</tr>
</tbody>
</table>
sCaption in
Text-token that will be displayed as title of the window.

iMsgType in
Defines the type of the message window to be displayed, with the corresponding text on the buttons; possible types:
- MMI_MB_OK
- MMI_MB_ABORT
- MMI_MB_OK_ABORT
- MMI_MB_ABORT_RETRY_CONT
- MMI_MB_YES_NO_ABORT
- MMI_MB_YES_NO
- MMI_MB_RETRY_ABORT
- MMI_MB_ABORT_CONT
- MMI(mb_ABORT_RETRY_IGNORE
- MMI_MB_ABORT_IGNORE

iRetKey out
Returns the button pressed, i.e.
- iRetKey:
- MMI_MB_RET_OK
- MMI_MB_RET_ABORT
- MMI_MB_RET_RETRY
- MMI_MB_RET_CONT
- MMI_MB_RET_YES
- MMI_MB_RET_NO
- MMI MB RET IGNORE

Return-Codes
- RC_OK Successful termination.
- BAS_NO_DLG_EXIST No dialog exists for this operation.

See Also
- MMI_WriteMsg
Example

The example uses the `MMI_WriteMsgStr` routine to display a message box with the title text "Warning" and the text:

```
MessageStr
time out in 10 seconds
```

and shows the buttons "Retry", "Abort" returning the button-id in `iRetKey`.

```pascal
CONST iTimeOut AS Integer = 10
DIM sMessage As String255
DIM iMBRetKey AS Integer

sMessage = "MessageStr\d010time out in " + Str$(iTimeOut) + "seconds"
MMI_WriteMsgStr( "Warning", sMessage,
                  MMI_MB_RETRY_ABORT, iMBRetKey )
```

6.1.29  MMI_DrawLine

Description  Draw a line.

Declaration  

```pascal
MMI_DrawLine( BYVAL iX1 AS Integer,
               BYVAL iY1 AS Integer,
               BYVAL iX2 AS Integer,
               BYVAL iY2 AS Integer,
               BYVAL iPen AS Integer )
```

Remarks  The function draws a line within the graphic field using the line-style `iPen`.

```
Note  A graphics dialog has to be set up before.
```

Parameters

- `iX1` in  x-co-ordinate of the beginning of the line [pixel]
- `iY1` in  y-co-ordinate of the beginning of the line [pixel]
- `iX2` in  x-co-ordinate of the end of the line [pixel]
- `iY2` in  y-co-ordinate of the end of the line [pixel]
iPen in  Line-style; possible values:
  MMI_PEN_WHITE
  MMI_PEN_BLACK
  MMI_PEN_DASHED

Return-Codes

RC_OK          Successful termination.
BAS_NO_DLG_EXIST No graphics dialog exists for this operation.

See Also
  MMI_CreateGraphDialog, MMI_DrawRect,
  MMI_DrawCircle, MMI_DrawText

Example
  The example uses the MMI_DrawLine routine to draw a line
  with the specified attributes.

  MMI_DrawLine( 10, 10, 100, 50, MMI_PEN_BLACK )

6.1.30 MMI_DrawRect

Description
  Draw a rectangle.

Declaration
  MMI_DrawRect( BYVAL iX1 AS Integer,
                 BYVAL iY1 AS Integer,
                 BYVAL iX2 AS Integer,
                 BYVAL iY2 AS Integer,
                 BYVAL iBrush AS Integer,
                 BYVAL iPen AS Integer )

Remarks
  This function draws a rectangle in the graphic field using the fill-
  style iBrush and the line-style iPen.

  *Note* A graphics dialog has to be set up before.
Parameters

\[
iX1 \quad \text{in} \quad \text{x-co-ordinate at the upper left-hand corner of the rectangle [pixel]}
\]
\[
iY1 \quad \text{in} \quad \text{y-co-ordinate at the upper left-hand corner of the rectangle [pixel]}
\]
\[
iX2 \quad \text{in} \quad \text{x-co-ordinate at the bottom right-hand corner of the rectangle [pixel]}
\]
\[
iY2 \quad \text{in} \quad \text{y-co-ordinate at the bottom right-hand corner of the rectangle [pixel]}
\]
\[
iBrush \quad \text{in} \quad \text{Fill-style for the rectangle; possible values:}
\]
\[
\text{MMI_BRUSH_WHITE}
\]
\[
\text{MMI_BRUSH_BLACK}
\]
\[
\text{MMI_NO_BRUSH}
\]
\[
iPen \quad \text{in} \quad \text{Line-style:}
\]
\[
\text{MMI_PEN_WHITE}
\]
\[
\text{MMI_PEN_BLACK}
\]
\[
\text{MMI_PEN_DASHED}
\]

Return-Codes

\[
\text{RC_OK} \quad \text{Successful termination.}
\]
\[
\text{BAS_NO_DLG_EXIST} \quad \text{No graphics dialog exists for this operation.}
\]

See Also

\[
\text{MMI_CreateGraphDialog, MMI_DrawLine, MMI_DrawCircle, MMI_DrawText}
\]

Example

The example uses the MMI_DrawRect routine to draw a rectangle with the specified attributes.

\[
\text{MMI_DrawRect( 10, 10, 100, 50, MMI_NO_BRUSH, MMI_PEN_BLACK )}
\]
6.1.31  MMI_DrawCircle

**Description**  Draw a circle / ellipse.

**Declaration**

```basic
MMI_DrawCircle( BYVAL iX AS Integer,
                 BYVAL iY AS Integer,
                 BYVAL iRx AS Integer,
                 BYVAL iRy AS Integer,
                 BYVAL iBrush AS Integer,
                 BYVAL iPen AS Integer )
```

**Remarks**

This function draws a circle in the graphic field, using the radius `iRx`, the fill-style `iBrush`, and the line-style `iPen`, as long as `iRx = iRy`. Otherwise, an ellipse is drawn, where `iRx` and `iRy` are the lengths of the perpendicular radii.

**Note**

A graphics dialog has to be set up before.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iX</td>
<td>x-co-ordinate at the centre of the circle/ellipse [pixel]</td>
</tr>
<tr>
<td>iY</td>
<td>y-co-ordinate at the centre of the circle/ellipse [pixel]</td>
</tr>
<tr>
<td>iRx</td>
<td>Radius of the circle, horizontal radius [pixel]</td>
</tr>
<tr>
<td>iRy</td>
<td>Radius of the circle, vertical radius [pixel]</td>
</tr>
<tr>
<td>iBrush</td>
<td>Fill-style for the rectangle; possible values:</td>
</tr>
<tr>
<td>iPen</td>
<td>Line-style; possible values:</td>
</tr>
</tbody>
</table>

- MMI_BRUSH_WHITE
- MMI_BRUSH_BLACK
- MMI_NO_BRUSH
- MMI_PEN_WHITE
- MMI_PEN_BLACK
- MMI_PEN_DASHED
6. System Functions

Return-Codes

- **RC_OK**  Successful termination.
- **BAS_NO_DLG_EXIST**  No graphics dialog exists for this operation.

See Also

- MMI_CreateGraphDialog, MMI_DrawLine,
- MMI_DrawRect, MMI_DrawText

Example

Draw a circle with a radius of 10.

```plaintext
MMI_DrawCircle( 80, 25, 10, 10,
    MMI_BRUSH_BLACK,
    MMI_PEN_BLACK )
```

### 6.1.32 MMI_DrawText

**Description**

Draw / delete text.

**Declaration**

```plaintext
MMI_DrawText( BYVAL iX AS Integer,  
    BYVAL iY AS Integer,  
    BYVAL sText AS String20,  
    BYVAL iAttr AS Integer,  
    BYVAL iPen AS Integer )
```

**Remarks**

This function either draws (iPen = MMI_PEN_BLACK) or deletes (iPen = MMI_PEN_WHITE) a text string in graphic field. The co-ordinates (iX, iY) correspond to the upper left-hand corner of the first character. The character size is 6 x 8 pixel.

**Parameters**

- **iX**  in  x-co-ordinate at the upper left-hand corner of the first character [pixel]
- **iY**  in  y-co-ordinate at the upper left-hand corner of the first character [pixel]
- **sText**  in  Pointer to the text string
- **iAttr**  in  Text attribute
  - **MMI_TXT_NORMAL**  normal text
  - **MMI_TXT_INVERSE**  inverted text

---

**Note**  A graphics dialog has to be set up before.
iPen in MMI_PEN_BLACK draw text
MMI_PEN_WHITE delete text

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No graphics dialog exists for this operation.</td>
</tr>
</tbody>
</table>

See Also

MMI_CreateGraphDialog, MMI_DrawLine, MMI_DrawRect, MMI_DrawCircle

Example

Print a text at position 10, 10.

```basic
DIM sOutput AS String20
sOutput = "distance"
MMI_DrawText(10, 10, sOutput, MMI_TXT_NORMAL, MMI_PEN_BLACK)
```

6.1.33 MMI_DrawBusyField

Description

Shows or hides the Busy-Icon.

Declaration

```basic
MMI_DrawBusyField(
    BYVAL lVisible as Logical)
```

Remarks

This function controls the Busy-Icon (Hourglass).

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lVisible</td>
<td>TRUE: Icon is visible</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>
Example
The example shows and hides the Busy-Icon

```geo
MMI_DrawBusyField(TRUE) ' show icon
'time consuming function....
MMI_DrawBusyField(FALSE) ' hide icon
```

6.1.34 MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong

**Description**
Create an alert beep.

**Declaration**
```
MMI_BeepAlarm()
MMI_BeepNormal()
MMI_BeepLong()
```

**Remarks**
The functions create one or a sequence of alert beeps with configurable volume, if the boxes are turned on.

Any previously set continuous signal beep will be finished.

**Return-Codes**
```
RC_OK Successful termination.
```

**See Also**
```
MMI_StartVarBeep
MMI_SwitchVarBeep
MMI_GetVarBeepStatus
```

**Example**
The example uses the MMI_BeepNormal to sound a signal beep.

```geo
MMI_BeepNormal()
```

6.1.35 MMI_StartVarBeep

**Description**
Start beep sequences with configurable interrupts.

**Declaration**
```
MMI_StartVarBeep( BYVAL iRate AS Integer )
```

**Remarks**
The function creates sequences of beeps with configurable interrupts.
If previously a continuous signal beep has been set, the new rate will be established.

**Parameters**

- `iRate` in frequency in [%]; 0 is very slow, 100 is very fast

**Return-Codes**

- `RC_OK` Successful termination.

**See Also**

- `MMI_BeepAlarm`
- `MMI_BeepNormal`
- `MMI_BeepLong`
- `MMI_SwitchVarBeep`
- `MMI_GetVarBeepStatus`

**Example**

The example uses the `MMI_StartVarBeep` to create a very fast sequence of signal beeps.

```
MMI_StartVarBeep( 100 )
```

### 6.1.36 MMI_SwitchVarBeep

**Description**

Switch a varying beep.

**Declaration**

```
MMI_SwitchVarBeep( BYVAL lOn AS Logical )
```

**Remarks**

The function allows the general switching (on/off) of a signal beep. A continuous signal beep will be switched off immediately.

**Parameters**

- `lOn` in switches the beep on or off

  - `lOn` meaning
    - `FALSE` the beep is switched off generally
    - `TRUE` beep is on; the functions `MMI_BeepNormal` etc. will only work if the beep is switched on.

**Return-Codes**

- `RC_OK` Successful termination.
See Also

MMI_BeepNormal,
MMI_BeepLong,
MMI_BeepAlarm,
MMI_StartVarBeep,
MMI_GetVarBeepStatus

Example

The example uses the MMI_SwitchVarBeep to switch off the beep.

MMI_SwitchVarBeep( TRUE )

6.1.37  MMI_GetVarBeepStatus

Description
Read the switch status for a variable signal beep.

Declaration
MMI_GetVarBeepStatus( lOn AS Logical )

Remarks
The function retrieves the state of the general signal beep switch.

Parameters

<table>
<thead>
<tr>
<th>lOn</th>
<th>out</th>
<th>state of the switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>off</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

Return-Codes

RC_OK       Successful termination.

See Also

MMI_BeepNormal,
MMI_BeepLong,
MMI_BeepAlarm,
MMI_StartVarBeep,
MMI_SwitchVarBeep
Example: The example uses the `MMI_GetVarBeepStatus` to revert the beep status (i.e. switch on when it is off and vice versa).

```plaintext
DIM lOn AS Logical

MMI_GetVarBeepStatus(lOn)
MMI_SwitchVarBeep( NOT lOn )
```

### 6.1.38 MMI_SwitchAFKey

**Description**: Switch the aF… key on or off.

**Declaration**: `MMI_SwitchAFKEY( BYVAL lOn AS Logical )`

**Remarks**: The function allows the switching (on/off) off the aF… key. Normally it is enabled, but during tracking distances it is disabled.

**Parameters**

<table>
<thead>
<tr>
<th>lOn</th>
<th>in</th>
<th>switches the beep on or off</th>
</tr>
</thead>
<tbody>
<tr>
<td>lOn</td>
<td>meaning</td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td>Key is switched off generally</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td>Key is active</td>
<td></td>
</tr>
</tbody>
</table>

**Return-Codes**

- **RC_OK**: Successful termination.

**See Also**: `BAP_MeasRec`, `BAP_MeasDistAng`

**Example**: The example uses the `MMI_SwitchAFKey` to disable the aF… key.

```plaintext
MMI_SwitchAFKey( FALSE )
```
6.1.39 MMI_SwitchIconsBeep

Description  Switches measurement icons and special beeps on or off.

Declaration  MMI_SwitchIconsBeep( BYVAL lOn AS Logical )

Remarks  The function allows the switching (on/off) of the measurement icons and special beeps (sector and lost lock).

Parameters

<table>
<thead>
<tr>
<th>lOn</th>
<th>in</th>
<th>switches the icons and beep on or off</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>FALSE no measurement icons and no special beep</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>TRUE the measurement icons will be updated and the beeps are enabled. This is the normal state during a measurement dialog with continuous measurements.</td>
</tr>
</tbody>
</table>

Return-Codes

| RC_OK | Successful termination. |

See Also  BAP_MeasRec
           BAP_MeasDistAng

Example  The example uses the MMI_SwitchIconsBeep to disable the icons and beeps.

MMI_SwitchIconsBeep( FALSE )
6.1.40 MMI_SetAngleRelation

Description  Set the angle relationship.

Declaration  
```plaintext
MMI_SetAngleRelation(  
    BYVAL iVertRel AS Integer,  
    BYVAL iHorzRel AS Integer)
```

Remarks  This function sets the relationship of the vertical and horizontal angles. Fields already displayed are not updated.

Parameters  
- **iVertRel** in  Relationship of the vertical angle; valid values:
  - MMI_VANGLE_IN_PERCENT
  - MMI_VANGLE_REL_HORIZON
  - MMI_VANGLE_REL_ZENIT

- **iHorzRel** in  Relationship of the horizontal angle; valid values:
  - MMI_HANGLE_CLOCKWISE
  - MMI_HANGLE_ANTICLOCKWISE
  - MMI_HANGLE_CLOCKWISE_SOUTH
  - MMI_HANGLE_BEARING

Return Codes  
- **RC_OK**  Successful termination.
- **RC_IVPARAM**  The function has been called with an invalid parameter

See Also  
- MMI_GetAngleRelation

Example  
Set the angle relations (with internal default values).
```plaintext
MMI_SetAngleRelation(  
    MMI_VANGLE_IN_PERCENT,  
    MMI_HANGLE_CLOCKWISE)
```
6.1.41 MMI_GetAngleRelation

**Description**  
Request the current angle relationships.

**Declaration**  
`MMI_GetAngleRelation(iVertRel AS Integer, iHorzRel AS Integer)`

**Remarks**  
This function returns the current vertical- and horizontal- angle relationships.

**Parameters**  
- `iVertRel`  
  Relationship of the vertical angle
- `iHorzRel`  
  Relationship of the horizontal angle

**Return Codes**  
none

**See Also**  
`MMI_SetAngleRelation`

**Example**  
Get the angle relations.
```basic
DIM iVertRel AS Integer
DIM iHorzRel AS Integer
MMI_GetAngleRelation( iVertRel, iHorzRel )
```

6.1.42 MMI_SetVAngleMode

**Description**  
Set the V-Angle mode.

**Declaration**  
`MMI_SetVAngleMode(BYVAL lAngleFree AS Logical)`

**Remarks**  
This function sets the vertical angle mode. Normally (`lAngleFree=FALSE`), the vertical angle is fix if there is a valid distance available. If `lAngleFree=TRUE`, the vertical angle will be updated including all corresponding values (slope distance, vertical distance, coordinates etc).
Parameters
lAngleFree in TRUE: V-Angle is free (running)

Return Codes
RC_OK Successful termination.

See Also
MMI_GetVAngleMode

Example
See example file „meas.gbs“.

6.1.43 MMI_GetVAngleMode

Description
Returns the V-Angle mode.

Declaration
MMI_GetVAngleMode(lAngleFree AS Logical)

Remarks
This function returns the vertical angle mode.

Parameters
lAngleFree in TRUE: V-Angle is free (running)

Return Codes
RC_OK Successful termination.

See Also
MMI_SetVAngleMode

Example
See example file „meas.gbs“.

6.1.44 MMI_SetAngleUnit

Description
Set the displayed unit of angle.

Declaration
MMI_SetAngleUnit(BYVAL iUnit AS Integer, BYVAL iDigits AS Integer)

Remarks
This function sets the displayed unit of angle. Existing display fields are not updated. If iDigits is greater than the maximal number it will be reset to it without notifying the user. A negative value of iDigits is not allowed.

Note
The maximal number of decimal digits depends on the Theodolite class.
### Parameters

<table>
<thead>
<tr>
<th>iUnit</th>
<th>Specified unit of angle; possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td>MMI_ANGLE_GON</td>
<td>400 Gon</td>
</tr>
<tr>
<td>MMI_ANGLE_DEC</td>
<td>360 Decimal</td>
</tr>
<tr>
<td>MMI_ANGLE_SEXADEC</td>
<td>360 Sexadecimal</td>
</tr>
<tr>
<td>MMI_ANGLE_MIL</td>
<td>6400 Mil</td>
</tr>
<tr>
<td>MMI_ANGLE_PERCENT</td>
<td>-300 ≤ x ≤ 300; only for vertical angles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>iDigits</th>
<th>Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle unit</td>
<td>places</td>
</tr>
<tr>
<td>MMI_ANGLE_GON</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_DEC</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_SEXADEC</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_MIL</td>
<td>0-3</td>
</tr>
<tr>
<td>MMI_ANGLE_PERCENT</td>
<td>don’t care</td>
</tr>
</tbody>
</table>

### Return Codes

- **RC_OK**  Successful termination.
- **RC_IVPARAM**  The function has been called with an invalid parameter

### See Also

- **MMI_GetAngleUnit**

### Example

Set the angle unit.

```plaintext
MMI_SetAngleUnit( MMI_ANGLE_GON, 3 )
```
6.1.45 MMI_GetAngleUnit

Description
Return the currently displayed unit of angle.

Declaration
MMI_GetAngleUnit(iUnit AS Integer, iDigits AS Integer)

Remarks
This function returns the current unit of angle.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iUnit</td>
<td>out</td>
<td>Specified unit of angle</td>
</tr>
<tr>
<td>iDigits</td>
<td>out</td>
<td>Number of decimal places</td>
</tr>
</tbody>
</table>

Return Codes

RC_OK
Successful termination.

See Also
MMI_SetAngleUnit

Example
Get the angle unit.

```basic
DIM iUnit AS Integer
DIM iDigits AS Integer
MMI_GetAngleUnit( iUnit, iDigits )
```

6.1.46 MMI_SetDistUnit

Description
Set the displayed unit of distance.

Declaration
MMI_SetDistUnit(BYVAL iUnit AS Integer, BYVAL iDigits AS Integer)

Remarks
This function sets the display unit for distance. Fields already displayed are not updated. If iDigits is greater than the maximal number it will be reset to it without notifying the user. A negative value of iDigits is not allowed.

Note
The maximal number of decimal digits depends on the Theodolite class.
Parameters

iUnit in
Specified unit of distance; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_DIST_METER</td>
<td>Meter</td>
</tr>
<tr>
<td>MMI_DIST_FOOT</td>
<td>normal foot</td>
</tr>
<tr>
<td>MMI_DIST_FOOT_INCH</td>
<td>normal foot / inch / 1/8inch</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT</td>
<td>US-foot</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT_INCH</td>
<td>US-foot / inch / 1/8inch</td>
</tr>
<tr>
<td>MMI_DIST_MM</td>
<td>Millimetre</td>
</tr>
<tr>
<td>MMI_DIST_INCH</td>
<td>inches</td>
</tr>
</tbody>
</table>

iDigits in
Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_DIST_METER</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_DIST_FOOT</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_DIST_FOOT_INCH</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_DIST_US_FOOT_INCH</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_DIST_MM</td>
<td>0</td>
</tr>
<tr>
<td>MMI_DIST_INCH</td>
<td>0-3</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>RC_OK</th>
<th>Successful termination.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

See Also

MMI_GetDistUnit

Example

Set the distance unit.

MMI_SetDistUnit( MMI_DIST_METER, 4 )
### 6.1.47 MMI_GetDistUnit

**Description**
Return the currently displayed unit of distance.

**Declaration**
```basic
MMI_GetDistUnit(iUnit AS Integer, iDigits AS Integer)
```

**Remarks**
This function returns the current unit of distance.

**Parameters**
- `iUnit` out
  Specified unit of distance
- `iDigits` out
  Number of decimal places.

**Return Codes**
- `RC_OK` Successful termination.

**See Also**
MMI_SetDistUnit

**Example**
Get the distance unit.
```basic
DIM iUnit AS Integer
DIM iDigits AS Integer
MMI_GetDistUnit(iUnit, iDigits)
```

### 6.1.48 MMI_SetPressUnit

**Description**
Set the displayed unit of pressure.

**Declaration**
```basic
MMI_SetPressUnit(iUnit AS Integer, iDigits AS Integer)
```

**Remarks**
This function sets the display unit for pressure. Fields already displayed are not updated. If `iDigits` is greater than 1 it will be reset to it without notifying the user. A negative value of `iDigits` is not allowed.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>iUnit</td>
<td>Specified unit of pressure; possible values:</td>
<td>MMI_PRESS_MBAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_MMHG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_INCHHG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_HPA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_PSI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>iDigits</td>
<td>Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:</td>
<td>angle unit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_MBAR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_MMHG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_INCHHG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_HPA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MMI_PRESS_PSI</td>
</tr>
</tbody>
</table>

### Return Codes

- **RC_OK**: Successful termination.
- **RC_IVPARAM**: The function has been called with an invalid parameter

### See Also

- **MMI_GetPressUnit**

### Example

Set the pressure unit.

```plaintext
MMI_SetPressUnit(MMI_PRESS_MBAR, 1)
```
### 6.1.49 MMI_GetPressUnit

**Description**  
Return the currently displayed unit of pressure.

**Declaration**  
```plaintext
MMI_GetPressUnit(iUnit AS Integer,
                  iDigits AS Integer)
```

**Remarks**  
This function returns the current unit of pressure.

**Parameters**

- **iUnit** out  
  Specified unit of pressure
- **iDigits** out  
  Number of decimal places.

**Return Codes**

- **RC_OK**  
  Successful termination.

**See Also**  
MMI_SetPressUnit

**Example**  
Get the pressure unit.
```plaintext
DIM iUnit AS Integer
DIM iDigits AS Integer
MMI_GetPressUnit( iUnit, iDigits )
```

### 6.1.50 MMI_SetTempUnit

**Description**  
Set the displayed unit of temperature.

**Declaration**  
```plaintext
MMI_SetTempUnit(BYVAL iUnit AS Integer,
                 BYVAL iDigits AS Integer)
```

**Remarks**  
This function sets the display unit for temperature. Fields already displayed are not updated. If **iDigits** is greater than 1 it will be reset to it without notifying the user. A negative value of **iDigits** is not allowed.
Parameters

iUnit in  Specified unit of temperature; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TEMP_C</td>
<td>Celsius</td>
</tr>
<tr>
<td>MMI_TEMP_F</td>
<td>Fahrenheit</td>
</tr>
</tbody>
</table>

iDigits in  Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TEMP_C</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_TEMP_F</td>
<td>0-1</td>
</tr>
</tbody>
</table>

Return Codes

- RC_OK  Successful termination.
- RC_IVPARAM  The function has been called with an invalid parameter

See Also  MMI_GetTempUnit

Example

Set the temperature unit.

```
MMI_SetTempUnit( MMI_TEMP_C, 1 )
```

6.1.51  MMI_GetTempUnit

Description  Return the currently displayed unit of temperature.

Declaration  

```
MMI_GetTempUnit(iUnit AS Integer, 
iDigits AS Integer)
```

Remarks  This function returns the current unit of temperature.

Parameters

- iUnit out  Specified unit of temperature
- iDigits out  Number of decimal places.
6. System Functions

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

**See Also**

MMI_SetTempUnit

**Example**

Get the temperature unit.

```basic
DIM iUnit AS Integer
DIM iDigits AS Integer

MMI_GetTempUnit( iUnit, iDigits )
```

### 6.1.52 MMI_SetDateFormat

**Description**

Set the date display format.

**Declaration**

`MMI_SetDateFormat(BYVAL iFormat AS Integer)`

**Remarks**

This function sets the format in which the date is to be displayed. Existing fields remain unchanged.

**Parameters**

- `iFormat` in
  - Specified date format; possible values:
    - `MMI_DATE_EU` European: DD.MM.YY
    - `MMI_DATE_US` US: MM/DD/YY
    - `MMI_DATE_JP` Japanese: YY/MM/DD

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

**See Also**

MMI_GetDateFormat
Example
Set the date format (internal default value).

```
MMI_SetDateFormat( MMI_DATE_EU )
```

### 6.1.53 MMI_GetDateFormat

**Description**
Retrieves the date display format.

**Declaration**

```
MMI_GetDateFormat(iFormat AS Integer)
```

**Remarks**
This function retrieves the format used to display the date.

**Parameters**

```
iFormat out Specified date format
```

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

**See Also**

MMI_SetDateFormat

**Example**
Get the date format.

```
DIM iFormat AS Integer
MMI_GetDateFormat( iFormat )
```

### 6.1.54 MMI_SetTimeFormat

**Description**
Set the time display format.

**Declaration**

```
MMI_SetTimeFormat(BYVAL iFormat AS Integer)
```

**Remarks**
This function sets the format in which the time is to be displayed. Existing fields remain unchanged.

**Parameters**

```
iFormat in Specified time format; possible values:
  value  meaning
  MMI_TIME_12H  12 hour display
  MMI_TIME_24H  24 hour display
```

---

TPS1100-Version 2.10
Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

See Also

- MMI_GetTimeFormat

Example

Set the time format (internal default value).

```plaintext
MMI_SetTimeFormat( MMI_TIME_12H )
```

### 6.1.55 MMI_GetTimeFormat

Description

Retrieves the time display format.

Declaration

```plaintext
MMI_GetTimeFormat(iFormat AS Integer)
```

Remarks

This function retrieves the format used to display the time.

Parameters

- `iFormat` out
  - Specified time format

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

See Also

- MMI_SetTimeFormat

Example

Get the time format.

```plaintext
DIM iFormat AS Integer
MMI_GetTimeFormat( iFormat )
```
6.1.56  MMI_SetCoordOrder

**Description**  Set the co-ordinate order.

**Declaration**  

```vbnet
MMI_SetCoordOrder(BYVAL iOrder AS Integer)
```

**Remarks**  This function sets the order of co-ordinates. The fields already displayed are not changed.

**Parameters**

- `iOrder` in

  Specifies the co-ordinate order; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_COORD_N_E</td>
<td>Order North East</td>
</tr>
<tr>
<td>MMI_COORD_E_N</td>
<td>Order East North</td>
</tr>
</tbody>
</table>

**Return Codes**

- RC_OK  Successful termination.
- RC_IVPARAM  The function has been called with an invalid parameter

**See Also**  MMI_GetCoordOrder

**Example**  Set the co-ordinate order (internal default value).

```vbnet
MMI_SetCoordOrder( MMI_COORD_N_E )
```
## 6.1.57 MMI_GetCoordOrder

**Description**
Retrieve the co-ordinate order.

**Declaration**
```
MMI_GetCoordOrder(iOrder AS Integer)
```

**Remarks**
This function retrieves the order in which co-ordinates are displayed.

**Parameters**
- `iOrder` out
  Specified co-ordinate order

**Return Codes**
- `RC_OK` Successful termination.

**See Also**
- `MMI_SetCoordOrder`

**Example**
Get the co-ordinate order.

```plaintext
DIM iOrder AS Integer
MMI_GetCoordOrder( iOrder )
```

## 6.1.58 MMI_SetLanguage

**Description**
Set the display language.

**Declaration**
```
MMI_SetLanguage( 
    BYVAL iLanguageNr AS Integer )
```

**Remarks**
This function sets the current language. All displayed text are immediately shown in the new language.

**Parameters**
- `iLanguageNr` in
  Specifies the language number; possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_REF_LANGUAGE</td>
<td>Reference language (English) = 1</td>
</tr>
<tr>
<td>MMI_MAX_LANGUAGE</td>
<td>Language numbers</td>
</tr>
</tbody>
</table>

```
Return Codes

- **RC_OK**: Successful termination.
- **RC_INVALIDPARAM**: The function has been called with an invalid parameter.
- **TXT_UNDEF_LANG**: The given language is not defined.

See Also
- `MMI_GetLanguage`

Example
- Set the language for the display (internal default value).
- ```
MMI_SetLanguage( MMI_REF_LANGUAGE )
```
# 6.1.60 MMI_GetLangName

**Description**
Gets the name to a language number.

**Declaration**
```vbnet
MMI_GetLangName( ByVal iLangNr AS Integer, sLangName AS String20 )
```

**Remarks**
This routine delivers the name associated with the number `iLangNr`.

**Parameters**
- `iLangNr` in Language number
- `sLangName` out Language description

**Return Codes**
- **RC_OK** Successful termination.
- **RC_IVPARAM** `iLangNr` is invalid

**See Also**
- MMI_SetLanguage
- MMI_GetLanguage

**Example**
Get the name of a language.
```vbnet
DIM sLangName AS String20
MMI_GetLangName( 2, sLangName )
```
### 6.2 BASIC APPLICATIONS BAP

#### 6.2.1 Summarizing Lists of BAP Types and Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SetAccessoriesDlg</td>
<td>Sets the used accessories</td>
</tr>
<tr>
<td>BAP_FineAdjust</td>
<td>Automatic target positioning</td>
</tr>
<tr>
<td>BAP_GetMeasPrg</td>
<td>Get the current distance measure program.</td>
</tr>
<tr>
<td>BAP_MeasDistAngle</td>
<td>Measures distance and angles.</td>
</tr>
<tr>
<td>BAP_MeasRec</td>
<td>Measures and record distance and angles.</td>
</tr>
<tr>
<td>BAP_PosTelescope</td>
<td>Positioning of the Telescope.</td>
</tr>
<tr>
<td>BAP_SearchPrism</td>
<td>Searches the prism.</td>
</tr>
<tr>
<td>BAP_SetHz</td>
<td>Sets the horizontal angle to 0 or another given value.</td>
</tr>
<tr>
<td>BAP_SetManDist</td>
<td>Set the distance manually.</td>
</tr>
<tr>
<td>BAP_SetMeasPrg</td>
<td>Set the distance measure program.</td>
</tr>
<tr>
<td>BAP_SetPpm</td>
<td>Sets the ppm for distance measurements.</td>
</tr>
<tr>
<td>BAP_SetPrism</td>
<td>Sets the current prism type and constant.</td>
</tr>
</tbody>
</table>
6.2.2 BAP_SetAccessoriesDlg

Description  Sets the used accessories.

Declaration  BAP_SetAccessoriesDlg()

Remarks  This function displays the accessories dialog.

Parameters

Return-Codes

RC_OK   Successful termination.

Example  The example displays the accessories dialog

BAP_SetAccessoriesDlg()

6.2.3 BAP_MeasDistAngle

Description  Measures distance and angles.

Declaration  BAP_MeasDistAngle( iDistMode AS Integer,
                        dHz AS Angle,
                        dV AS Angle,
                        dDist AS Distance,
                        BYVAL lDisplayOn AS Logical,
                        BYVAL sCaptionLeft AS _Token )

Remarks  Measures distance and angles and updates the data pool after correct measurements. It controls the special beep (Sector or Lost Lock) and switches measurement icons and disables the aF... key during tracking.
### Parameters

<table>
<thead>
<tr>
<th>iDistMode</th>
<th>Distance measuring modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode as Input</strong></td>
<td>** Meaning**</td>
</tr>
<tr>
<td>BAP_NO_MEAS</td>
<td>No new measurement, get last one</td>
</tr>
<tr>
<td>BAP_NO_DIST</td>
<td>No distance measurement, get only angles</td>
</tr>
<tr>
<td>BAP_DEF_DIST</td>
<td>Measure distance and angles using default measurement program</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Measure distance and angles using the tracking measurement program</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Measure distance and angles using the fast tracking measurement program</td>
</tr>
<tr>
<td>BAP_STOP_TRK</td>
<td>Stop tracking, no measurement. No valid results returned.</td>
</tr>
<tr>
<td>BAP_CLEAR_DIST</td>
<td>Clear distance (Theodolite data-pool), no measurement. No valid results returned.</td>
</tr>
<tr>
<td>BAP_RED_TRK_DIST</td>
<td>Measure distance and angles using the tracking with red laser measurement program</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Mode returned</strong></th>
<th>** Meaning**</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_DEF_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
</tbody>
</table>

**All other modes** Returns BAP_DEF_DIST.

\[ dHz, dV \text{ out } \] Angles [rad], depends on
iDistMode

dDist out Distance [m], depends on iDistMode

sCaptionLeft in Left caption for the distance measurement display.

lDisplayOn in TRUE: shows the distance measurement display during distance measurement.

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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>Warning, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC submodule already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>An error occurred during distance measurement</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>Error, wrong setting of PPM or MM on EDM</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>TMC_SIGNAL_ERROR</td>
<td>Error, no signal on EDM (only in signal mode)</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Error, measurement aborted</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

**See Also**
- BAP_MeasRec

**Example**

See example file „meas.qbs“.

The example uses the BAP_MeasDistAngle routine to measure a distance and angles.

```basic
DIM iDistMode AS Integer
DIM dHz AS Angle
DIM dV AS Angle
DIM dDist AS Distance

iDistMode = BAP_DEF_DIST
BAP_MeasDistAngle(iDistMode, dHz, dV, dDist, TRUE, "TEST")
```

---

6-80 TPS1100-Version 2.10
6.2.4 BAP_MeasRec

Description
Measures distance and angles records.

Declaration
BAP_MeasRec( iDistMode AS Integer, 
BYVAL lDisplayOn AS Logical, 
BYVAL sCaptionLeft AS _Token )

Remarks
Measures distance and angles and updates the Theodolite data pool after correct measurements and records values according to the predefined record mask. After recording, a running point number will be incremented.

It controls the special beep (Sector or Lost Lock), switches Measurement icons and disables aF... Key during tracking.

Parameters

<table>
<thead>
<tr>
<th>iDistMode</th>
<th>Distance measuring modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode as Input</strong></td>
<td><strong>Meaning</strong></td>
</tr>
<tr>
<td>BAP_NO_MEAS</td>
<td>No new measurement before recording</td>
</tr>
<tr>
<td>BAP_NO_DIST</td>
<td>No distance measurement before recording (only new angles)</td>
</tr>
<tr>
<td>BAP_DEF_DIST</td>
<td>Use default distance measurement program and record values</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Use the tracking measurement program and record values</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Use the fast tracking measurement program and record values</td>
</tr>
<tr>
<td>BAP_STOP_TRK</td>
<td>Stop tracking, no measurement and no recording</td>
</tr>
<tr>
<td>BAP_CLEAR_DIST</td>
<td>Clear distance (Theodolite data pool), no measurement and no recording.</td>
</tr>
<tr>
<td>BAP_RED_TRK_DIST</td>
<td>Use the tracking with red laser measurement program and record values</td>
</tr>
</tbody>
</table>
### Mode returned

<table>
<thead>
<tr>
<th>Mode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_DEF_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>All other modes</td>
<td>Returns BAP_DEF_DIST.</td>
</tr>
</tbody>
</table>

### Return Codes

- **RC_OK**: Successful termination.
- **WIR_NO_MEDIUM**: No storage medium is available.
- **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
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>No target detected</td>
</tr>
<tr>
<td>AUT_RC_TIMEOUT</td>
<td>Position not reached</td>
</tr>
<tr>
<td>BAP_CHANGE_ALL_TO_DIST</td>
<td>No prism has been found during distance measurement with ATR, command changed from &quot;All&quot; to &quot;Dist&quot;</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ACCURACY_GUARANTEE</td>
<td>Info, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>Warning, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC sub-module already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>An error occurred during distance measurement.</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>Error, wrong setting of PPM or MM on EDM</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>TMC_SIGNAL_ERROR</td>
<td>Error, no signal on EDM (only in signal mode)</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Error, measurement aborted</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

**See Also**

BAP_MeasDistAngle, GSI_SetRecMask
Example
See example file "meas.gbs".
The example uses the BAP_MeasMeasRec routine to record actual distance and angles (no new measurement).
DIM iDistMode AS Integer

iDistMode = BAP_NO_MEAS ' no measurement
BAP_MeasRec(iDistMode, FALSE, "")

6.2.5 BAP_FineAdjust

Description
Automatic target positioning.

Declaration
BAP_FineAdjust(
    BYVAL dSearchHz AS Angle,
    BYVAL dSearchV AS Angle
)

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 dSearchV in V- direction and by parameter dSearchHz in Hz - direction. If no target is found, the instrument turns back to the initial start position. The ATR mode must be enabled for this functionality, see CSV_SetATRStatus and CSV_GetATRStatus.

Parameters
- dSearchHz in Search range Hz
- dSearchV in Search range V

Return Codes
- RC_OK Successful termination.
- AUT_RC_TIMEOUT Timeout while positioning of one or both axes. The position fault lies above 100[cc].
- 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_      Inadequate environment conditions.
ENVIRONMENT

AUT_RC_DEV_ERROR During the determination of the angle  
deviation error detected, repeat fine positioning

AUT_RC_NOT_      ATR mode not enabled, enable ATR  
ENABLED          mode

AUT_RC_         ATR error, at repeated occur call  
DETECTOR_ERROR  service

See Also CSV_SetATRStatus, CSV_GetATRStatus

Example The example see sample TRACKING.GBS.

6.2.6 BAP_SearchPrism

Description Searches the prism.

Declaration BAP_SearchPrism(  
            BYVAL lShowMessages As Logical )

Remarks This procedure searches the prism. The searching area depends on  
the defined searching area and on the setting of the additional  
working area.  
This routine works only in ATR instruments and needs at least  
Firmware-Release 2.00

Parameters

lShowMessages in TRUE: show error-messages if  
there are problems to find the prism

Return Codes

RC_OK         Successful termination.

AUT_RC_TIMEOUT Timeout while positioning of one or  
both axes. The position fault lies above 100[cc].
6. System Functions

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

See Also
CSV_SetATRStatus, CSV_GetATRStatus

6.2.7 BAP_SetManDist

Description
Set the distance manually.

Declaration
BAP_SetManDist(
    BYVAL sCaptionLeft AS _Token,
    BYVAL dDistance AS Double,
    iButtonId AS Integer )

Remarks
The BAP_SetManDist routine starts a dialog with the caption sCaption where the user can enter a horizontal distance. The distance will be stored into the Theodolite data pool.

Parameters

sCaptionLeft in left caption string of the dialog
dDistance in initial value for the distance. A negative value will be displayed as "----"
iButtonId out identifier of the pressed valid button to exit the dialog
Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC sub-module already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

See Also

TMC_IfDistTapeMeasured, TMC_SetHandDist, TMC_GetPolar, TMC_GetCoordinate

Example

The example uses the BAP_SetManDist routine to enter a distance.

```basi
DIM iButton AS Integer
DIM dInitDist AS Distance

dInitDist = 15.0 'initial value
BAP_SetManDist( "BASIC", dInitDist, iButton )
```

6.2.8 BAP_SetPpm

Description
Sets the PPM for distance measurements.

Declaration
BAP_SetPpm()  

Remarks
The BAP_SetPpm routine opens a dialog which the user can complete in order to calculate the PPM (parts per million) correction to be used to reduce the distance measured by the EDM.

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>
RC_SET_INCOMPL  Parameter set-up for subsystem incomplete.

See Also  BAP_SetManDist, BAP_SetPrism

Example  The example uses the BAP_SetPpm routine to open the PPM dialog.

BAP_SetPpm()

6.2.9  BAP_SetPrism

Description  Sets the current prism type and constant.

Declaration  BAP_SetPrism()

Remarks  The BAP_SetPrism routine opens a dialog which the user can complete in order to choose one of five prism types/constants. Two types are LEICA defaults, whereas the other three can be named and the constant values given/changed by the user. The prism constants are always given and displayed in millimetres, regardless of the distance units in use at the time.

Return Codes

RC_OK  Successful termination.

See Also  BAP_SetManDist, BAP_SetPpm

Example  The example uses the BAP_SetPrism routine to open the Prism dialog.

BAP_SetPrism()

6.2.10  BAP_SetMeasPrg

Description  Set the distance measure program.

Declaration  BAP_SetMeasPrg( BYVAL iMeasPrg AS Integer )
Remarks
The `BAP_SetMeasPrg` routine sets the program for the distance measurement.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMeasPrg</td>
<td>in</td>
<td>Distance measure program</td>
</tr>
</tbody>
</table>

Valid measure programs

<table>
<thead>
<tr>
<th>Measure Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SINGLE_REF_</td>
<td>Single measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>STANDARD</td>
<td></td>
</tr>
<tr>
<td>BAP_SINGLE_REF_</td>
<td>Single measurement, with reflector, fast</td>
</tr>
<tr>
<td>FAST</td>
<td></td>
</tr>
<tr>
<td>BAP_SINGLE_REF_</td>
<td>Single measurement, with reflector and red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_SINGLE_RLESS_</td>
<td>Single measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_CONT_REF_</td>
<td>Continuous measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>STANDARD</td>
<td></td>
</tr>
<tr>
<td>BAP_CONT_REF_</td>
<td>Continuous measurement, with reflector, fast</td>
</tr>
<tr>
<td>FAST</td>
<td></td>
</tr>
<tr>
<td>BAP_CONT_RLESS_</td>
<td>Continuous measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_AVG_REF_</td>
<td>Average measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>STANDARD</td>
<td></td>
</tr>
<tr>
<td>BAP_AVG_REF_</td>
<td>Average measurement, with reflector and red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
<tr>
<td>BAP_AVG_RLESS_</td>
<td>Average measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>VISIBLE</td>
<td></td>
</tr>
</tbody>
</table>

See Also
BAP_GetMeasPrg
Example
The example uses the BAP_SetMeasPrg routine to set the distance measurement program on single measurement without reflector.

BAP_SetMeasPrg(BAP_SINGLE_RLESS_VISIBLE)

6.2.11 BAP_GetMeasPrg

Description
Get the current distance measure program.

Declaration
BAP_GetMeasPrg( iMeasPrg AS Integer )

Remarks
The BAP_GetMeasPrg routine fetches the current program for the distance measurement.

Parameters

iMeasPrg out Distance measure program

Valid measure programs

Meaning

BAP_SINGLE_REF_STANDARD
Single measurement, with reflector, standard speed

BAP_SINGLE_REF_FAST
Single measurement, with reflector, fast

BAP_SINGLE_REF_VISIBLE
Single measurement, with reflector and red laser

BAP_SINGLE_RLESS_VISIBLE
Single measurement, reflectorless, with red laser

BAP_CONT_REF_STANDARD
Continuous measurement, with reflector, standard speed

BAP_CONT_REF_FAST
Continuous measurement, with reflector, fast

BAP_CONT_RLESS_VISIBLE
Continuous measurement, reflectorless, with red laser

BAP_AVG_REF_STANDARD
Average measurement, with reflector, standard speed

BAP_AVG_REF_VISIBLE
Average measurement, with reflector and red laser

BAP_AVG_RLESS_VISIBLE
Average measurement, reflectorless, with red laser
See Also  
BAP_SetMeasPrg

Example  
The example uses the BAP_GetMeasPrg routine to fetch the current distance measurement program.
DIM iMeasPrg AS Integer
BAP_GetMeasPrg(iMeasPrg)

6.2.12 BAP_PosTelescope

Description  
Positioning of the Telescope.

Declaration  
BAP_PosTelescope(
BYVAL eMode AS Integer,
BYVAL eDspMode AS Integer,
BYVAL dHz AS Double,
BYVAL dV AS Double,
BYVAL dHzTolerance AS Double,
BYVAL dVTolerance AS Double)

Remarks  
This procedure positions the telescope according to the specified mode and angles.

Parameters  

eMode  
Positioning mode.
BAP_POSIT  
positioning on Hz and V angle
BAP_POSIT_HZ  
positioning on Hz angle
BAP_POSIT_V  
positioning on V angle
BAP_CHANGE_FACE  
change face
**eDspMode**

Controls the context and layout of the display during manual positioning. This parameter has no effect on motorised Theodolites.

- **BAP_POS_NOMSG** No message will be displayed
- **BAP_POS_MSG** Only a message will be displayed
- **BAP_POS_DLG** Positioning will be guided with a dialog if it is a non motorised Theodolite

**dHz, dV**

Target position

**dHzTolerance, dVTolerance**

In case of manual positioning, the tolerances define the upper and lower boundaries of the target position. For successful termination of the positioning, the final target position must be within these boundaries. If the tolerance is lower than the default accuracy of the Theodolite, the tolerance will be the default accuracy.

**Return Codes**

- **RC_OK** Positioning successful
- **RC_ABORT** Abnormal termination (No positioning possible, ESC-Key)

**See Also**

- CSV_MakePositioning
- CSV_ChangeFace

**Example**

Position the telescope.

```
BAP_PosTelescope(BAP_CHANGE_FACE, BAP_POS_DLG, 0, 0, .5, .5 )
```
### 6.2.13 BAP_SetHz

**Description**  
Sets the horizontal angle to 0 or another given value.

**Declaration**  
BAP_SetHz( BYVAL sCaptionLeft AS _Token )

**Remarks**  
This procedure offers a dialogue which the user can complete in order to influence the angular offset provided by the TMC subsystem for the horizontal angle encoder. A button is provided for setting the angle to zero, directly, or the user may prefer to input another given value. Furthermore, the angle beep (at the quarter circle positions from 0°) can be turned on and off.

**Note**  
If the instrument is in Lock mode, then the instrument tries to lock first before it sets the angle to 0.

**Parameters**
- sCaptionLeft  
  Left caption text for dialog

**See Also**

**Return Codes**
- RC_OK  
  Horizontal angular offset correct.

**Example**  
Set the horizontal angle.

BAP_SetHz("BASIC")
6.3 MEASUREMENT FUNCTIONS TMC

This section contains the lower level measurement procedures.

6.3.1 Summarizing Lists of TMC Types and Procedures

6.3.1.1 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_ANG_SWITCH_Type</td>
<td>Angle measurement switches</td>
</tr>
<tr>
<td>TMC_Angle_Type</td>
<td>Data structure for measuring angles.</td>
</tr>
<tr>
<td>TMC_Coordinate_Type</td>
<td>Data structure for the co-ordinates (tracking and fixed co-ordinates).</td>
</tr>
<tr>
<td>TMC_Dist_SWITCHES_Type</td>
<td>Distance measurement switches</td>
</tr>
<tr>
<td>TMC_Distance_Type</td>
<td>Data structure for the distance measurement.</td>
</tr>
<tr>
<td>TMC_HZ_V_Ang_Type</td>
<td>Horizontal and vertical angle.</td>
</tr>
<tr>
<td>TMC_Incline_Type</td>
<td>Data structure for the inclination measurement.</td>
</tr>
<tr>
<td>TMC_OFFSET_DIST_Type</td>
<td>Target offset</td>
</tr>
<tr>
<td>TMC_PPM_CORR_Type</td>
<td>Correction for distance measurement</td>
</tr>
<tr>
<td>TMC_REFRACTION_Type</td>
<td>Refraction correction for distance measurement</td>
</tr>
<tr>
<td>TMC_STATION_Type</td>
<td>Station co-ordinates</td>
</tr>
</tbody>
</table>

6.3.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_DoMeasure</td>
<td>Start a measure program.</td>
</tr>
<tr>
<td>TMC_Get/SetAngleFaceDef</td>
<td>Gets and sets the current face definition.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TMC_Get/SetRefractiveCorr</td>
<td>Gets and sets the refractive correction for measuring the distance.</td>
</tr>
<tr>
<td>TMC_Get/SetRefractiveMethod</td>
<td>Gets and sets the method of refractive correction for measuring the distance.</td>
</tr>
<tr>
<td>TMC_Get/SetDistPpm</td>
<td>Gets and sets the correction values for distance measurements.</td>
</tr>
<tr>
<td>TMC_Get/SetHeight</td>
<td>Gets and sets the current height of the reflector.</td>
</tr>
<tr>
<td>TMC_Get/SetHzOffset</td>
<td>Gets and sets the current horizontal offset.</td>
</tr>
<tr>
<td>TMC_Get/SetStation</td>
<td>Gets and sets station co-ordinates.</td>
</tr>
<tr>
<td>TMC_GetAngle</td>
<td>Measure angles.</td>
</tr>
<tr>
<td>TMC_GetAngle_Winc</td>
<td>Measure angles with inclination control</td>
</tr>
<tr>
<td>TMC_GetAngSwitch</td>
<td>Returns the angle measurement correction switches</td>
</tr>
<tr>
<td>TMC_GetCoordinate</td>
<td>Calculate and read co-ordinates.</td>
</tr>
<tr>
<td>TMC_GetDistSwitch</td>
<td>Returns the distance measurement correction switches</td>
</tr>
<tr>
<td>TMC_GetFace1</td>
<td>Get face information of current telescope position</td>
</tr>
<tr>
<td>TMC_GetInclineStatus</td>
<td>Returns the inclination compensator status.</td>
</tr>
<tr>
<td>TMC_GetInclineSwitch</td>
<td>Returns the compensator switch</td>
</tr>
<tr>
<td>TMC_GetOffsetDist</td>
<td>Returns the distance measurement offset</td>
</tr>
<tr>
<td>TMC_GetPolar</td>
<td>Calculate and read polar co-ordinates.</td>
</tr>
<tr>
<td>TMC_GetSimpleMea</td>
<td>Gets the results of distance and angle measurement</td>
</tr>
<tr>
<td>TMC_IfDistTapeMeasured</td>
<td>Gets information about manual measurement.</td>
</tr>
<tr>
<td>TMC_IfOffsetDistMeasured</td>
<td>Returns the EDM measurement mode</td>
</tr>
<tr>
<td>TMC_QuickDist</td>
<td>Measure slope distance and angles</td>
</tr>
<tr>
<td>TMC_SetAngSwitch</td>
<td>Defines the angle measurement correction switches</td>
</tr>
<tr>
<td>TMC_SetDistSwitch</td>
<td>Defines the distance measurement correction switches</td>
</tr>
<tr>
<td>TMC_SetHandDist</td>
<td>Sets distance manually.</td>
</tr>
</tbody>
</table>
### 6.3.2 TMC Data Structures

#### 6.3.2.1 TMC_INCLINE - Data structure for the inclination measurement

```plaintext
type TMC_Incline_Type
dCrossIncline AS Double cross inclination
dLengthIncline AS Double alongside inclination
dAccuracyIncline AS Double accuracy of measuring
InclineTime AS Integer time of measuring
end TMC_Incline_Type
```

#### 6.3.2.2 TMC_ANGLE - Data structure for measuring angles

```plaintext
type TMC_Angle_Type
dHz AS Double horizontal angle
dV AS Double vertical angle
dAngleAccuracy AS Double accuracy of angle
iAngleTime AS Integer time of measurement
Incline AS TMC_Incline_Type inclination belonging to the measurement
iFace AS Integer information about position of the telescope
end TMC_Angle_Type
```
6.3.2.3 TMC_DISTANCE - Data structure for the distance measurement

TYPE TMC_Distance_Type
  Angle AS TMC_Angle_Type set of angles belonging to distance
  dSlopeDist AS Double slope distance
  dSlopeDistAccuracy AS Double accuracy of distance
  dHorizDist AS Double horizontal distance
  dHeightDiff AS Double difference in altitude
  AngleCont AS TMC_Angle_Type set of angles, measured continuously
  dSlopeDistCont AS Double slope distance, measured continuously
  dHeightDiffCont AS Double distance in altitude, measured continuously
END TMC_Distance_Type

6.3.2.4 TMC_COORDINATE - Data structure for the coordinates
(tracking and fixed co-ordinates)

TYPE TMC_Coordinate_Type
  dE AS Double east co-ordinate
  dN AS Double north co-ordinate
  dH AS Double height co-ordinate
  iCoordTime AS Integer time of measurement
  dE_Cont AS Double east coordinate, measured continuously
  dN_Cont AS Double north co-ordinate, measured continuously
  dH_Cont AS Double height co-ordinate, measured continuously
  iCoordContTime AS Integer time of continuous measurement
END TMC_Coordinate_Type

6.3.2.5 TMC_HZ_V_Ang - Horizontal and vertical angle

TYPE TMC_HZ_V_Ang_Type
  dHz AS Double horizontal angle
  dV AS Double vertical angle
END TMC_HZ_V_Ang_Type
6.3.2.6 TMC_PPM_CORR - Correction for distance measurement

TYPE TMC_PPM_CORR_Type
    dPpmI AS Double individual
dPpmA AS Double atmospheric
dPpmR AS Double height relative
dPpmP AS Double projection contortion
END TMC_PPM_CORR_Type

6.3.2.7 TMC_STATION - Station coordinates

TYPE TMC_STATION_Type
    dE0 AS Double easting co-ordinate
dN0 AS Double northing co-ordinate
dH0 AS Double height co-ordinate
dHi AS Double instrument height
END TMC_STATION_Type

6.3.2.8 TMC_REFRACTION- Refraction correction for distance measurement

TYPE TMC_REFRACTION_Type
    bOnOff AS Logical TRUE if refraction is valid
dEarthRadius AS Double earth radius
dRefractiveScale AS Double refraction coefficient
END TMC_REFRACTION_Type

6.3.2.9 TMC_DIST_SWITCH_Type- Distance measurement switches

TYPE TMC_DIST_SWITCHES_Type
    lAxisDifferCorr AS Logical ' EDM to optical axis correction
    lProjectScaleCorr AS Logical ' Projection scale correction
    lHgtReductionCorr AS Logical ' Height reduction correction
END TMC_DIST_SWITCHES_Type
6.3.2.10 TMC_ANGLE_SWITCH_Type – Angle measurement switches

TYPE TMC_ANG_SWITCH_Type
  lInclineCorr AS Logical ' Inclination correction
  lStandAxisCorr AS Logical ' Standing axis correction
  lCollimationCorr AS Logical ' Collimation error correction
  lTiltAxisCorr AS Logical ' Tilting axis correction
END TMC_ANG_SWITCH_Type

6.3.2.11 TMC_OFFSET_DIST_Type – Target offset

TYPE TMC_OFFSET_DIST_Type
  dLengthVal AS Distance ' Target - Offset Length
  dCrossVal AS Distance ' Target - Offset Cross
  dHeightVal AS Distance ' Target - Offset Height
END TMC_OFFSET_DIST_Type

6.3.3 TMC_DoMeasure

Description  Start a measure program.
Declaration  TMC_DoMeasure( BYVAL iCommand AS Integer )
Remarks  With this function a measure program is started. The commands start a distance measurement and/or a test mode. In addition an angle- and an inclination-measure are done (not at measurement).

The tracking measure program performs e.g. as follows: Start the measure program with TMC_DoMeasure(TMC_TRK_DIST). The electronic distance measuring device (EDM) begins to run. Now the co-ordinates can be read, e.g. with TMC_GetCoordinate(). Tracking can be stopped with TMC_DoMeasure(TMC_STOP). With TMC_DoMeasure(TMC_CLEAR) the function will be stopped and the distance cleared.
After calling a measure program, the last valid distance results will be cleared (as after TMC_STOP).

**Parameters**

- `iCommand in` start a measure program; possible values:
  - `TMC_STOP` switch off EDM and finish program
  - `TMC_DEF_DIST` do default distance measure
  - `TMC_TRK_DIST` do tracking distance measure
  - `TMC_RTRK_DIST` do fast tracking distance measure
  - `TMC_CLEAR` clear distance and switch off EDM
  - `TMC_SIGNAL` start signal measurement (test mode)
  - `TMC_RED_TRK_DIST` do tracking distance measure with red laser

**See Also**

- `TMC_GetPolar`
- `TMC_GetCoordinate`

**Return Codes**

- `RC_OK` measure program started
- `RC_IVPARAM` The function has been called with an invalid parameter
- `TMC_BUSY` Measurement system is busy
Example

Start a distance measure, do something, stop it and clear results.

The following variable has to be defined:

```plaintext
TMC_DoMeasure (TMC_DEF_DIST) ' ... do a measure
TMC_DoMeasure (TMC_CLEAR)
```

6.3.4 TMC_GetPolar

Description

Calculate and read polar co-ordinates.

Declaration

```plaintext
TMC_GetPolar(  
    BYVAL iWaitTime AS Integer,  
    Polar AS TMC_Distance_Type,  
    iReturnCode AS Integer )
```

Remarks

The function corrects and takes in calculation a measured distance. Angle and possibly inclination are being calculated. The result is a point in polar co-ordinates.

Simple and multiple measures (distance tracking, altitude tracking) are supported. The horizontal and the inclined distance with the difference in altitude are read. The delay (iWaitTime) just works on the distance measure, not on the measure of the angle. As long as no new measure program is started, the results can be read. Additional to the normal return codes iReturnCode delivers also informational return codes which will not interrupt program execution.

Note

The measure program must have been started (see TMC_DoMeasure).

Parameters

iWaitTime in delay time [ms] until a result is available

=0 returns results with an already measured distance.
>0 waits maximal the time iWaitTime for a result. If iWaitTime is chosen big enough (e.g. 60000, which is surely longer than the time-out period of the device), the system will wait for a result or until an error occurs.

<0 Performs an automatic target acquisition (if possible) and then tries to measuring in a until a valid result or an irrecoverable error occurs. The value itself of iWaitTime is ignored.

Polar out point in polar co-ordinates
iReturnCode out see Additional Codes below

See Also TMC_GetCoordinates

Additional Codes in iReturnCode

RC_OK measurement and values are OK
TMC_ACCURACY_GUARANTEE Accuracy is not guaranteed, because the results 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.
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.

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>measurement and values are OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy. Repeat measurement.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>

Example  Start a distance measure, perform measure.

```basic
DIM iRetCode AS Integer
DIM iWaitTime AS Integer
DIM Polar AS TMC_Distance_Type
DIM lError AS Logical
DIM lDone AS Logical
```
'start distance measurement
ON ERROR RESUME ' to get valid angles
TMC_DoMeasure( TMC_DEF_DIST )

iWaitTime = -1
lDone = FALSE
lError = FALSE

DO 'display measured values
  TMC_GetPolar( iWaitTime, Polar, iRetCode )
  SELECT CASE iRetCode
  CASE RC_OK
    'display all data
    'e.g. set lDone here
  CASE else
    'handle error
    lError = TRUE
  END SELECT
  LOOP UNTIL lError OR lDone

'stop distance measurement
TMC_DoMeasure( TMC_CLEAR )

6.3.5 TMC_GetCoordinate

Description Calculate and read co-ordinates.

Declaration TMC_GetCoordinate(
  BYVAL iWaitTime AS Integer,
  Coordinate AS TMC_COORDINATE_Type,
  iReturnCode AS Integer )

Remarks The function calculates and output co-ordinates. Angle and possibly inclination are being measured. The co-ordinates are being corrected. The result is a point in Cartesian co-ordinates. The system calculates co-ordinates and tracking co-ordinates.

Simple and multiple measurements (distance-, altitude- and co-ordinate-tracking) are supported. The delay (iWaitTime) just works on the distance measure, not on the measuring of the angle.
As far as no new measure program is started, the results can be read. Additional to the normal return codes iReturnCode delivers also informational return codes which will not interrupt program execution.

**Note**  The measure program must have been started (see TMC_DoMeasure).

**Parameters**

- **iWaitTime**  
  delay time [ms] until a result is available  
  - 0 returns already measured values  
  - >0 waits the maximal time iWaitTime for a result

- **Coordinate**  
  point in Cartesian co-ordinates (output)

- **iReturnCode**  
  return code, see Additional Codes

**See Also**  
TMC_GetPolar

**Additional Codes in iReturnCode**

- **RC_OK**  
  measurement and values are OK

- **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.

- **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.

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>measurement and values are OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>

Example
Start a distance measure, perform measurement.

```basic
DIM iretCode AS Integer
DIM iWaitTime AS Integer
DIM Coord AS TMC_COORDINATE_Type
DIM lError AS Logical
DIM lDone AS Logical

ON ERROR RESUME NEXT ' to get valid angle data
TMC_DoMeasure( TMC_DEF_DIST )
lDone = FALSE
lError = FALSE
```
DO ' display measured values
    TMC_GetCoordinate( 5, Coord, iRetCode )
SELECT CASE iRetCode
    CASE RC_OK
        'display all data
        'e.g. set lDone
    CASE ANGLE_OK
        ' display coordinate
    CASE ELSE
        'handle error
        lError = TRUE
END SELECT
LOOP UNTIL lError OR lDone
TMC_DoMeasure( TMC_CLEAR )

6.3.6 TMC_GetAngle

**Description**
Measure angles.

**Declaration**

```plaintext
TMC_GetAngle( Angles AS TMC_ANGLE_Type, iReturnCode AS Integer )
```

**Remarks**

The function measures the horizontal and vertical angle and the possibly belonging inclination, if the inclination compensation is on. If the compensation is off and no valid inclination is present, there may be a delay if the inclination can't be measured immediately. The correction values for the inclination can be calculated with several methods.

As long as no new measure program is started, the results can be read. Additional to the normal return codes `iReturnCode` delivers also informational return codes which will not interrupt program execution.

**Parameters**

- **Angles** `out` result of measuring the angle
- **iReturnCode** `out` return code, see Additional Codes

**See Also**

TMC_DoMeasure
Additional Codes in iReturnCode

RC_OK
Execution successful.

TMC_NO_FULL_CORRECTION
The results are not corrected by all active sensors. Angle data are available.
This message is to be considered 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 considered as info.

Return Codes

RC_OK
angle OK

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.

Example
Read the currently valid angle.
DIM Angles AS TMC_ANGLE_Type
DIM RetCode AS Integer
TMC_GetAngle( Angles, RetCode )
6.3.7 TMC_GetAngle_WInc

**Description**  
Measure angles with inclination control.

**Declaration**  
```vbs
TMC_GetAngle_WInc(
    iIncProg AS Integer,
    Angle AS TMC_ANGLE,
    iReturnCode AS Integer )
```

**Remarks**  
The function measures the horizontal and vertical angle and in dependence of the configuration, the inclination.

As far as no new measure program is started, the results can be read. Additional to the normal return codes `iReturnCode` delivers also informational return codes, which will not interrupt program execution.

**Parameters**
- `iIncProg` in
  The manner of incline compensation.
  Following settings are possible:
  
<table>
<thead>
<tr>
<th>Incline Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_MEA_INC</td>
<td>get inclination (apriori sigma)</td>
</tr>
<tr>
<td>TMC_AUTO_INC</td>
<td>get inclination with automatism (sensor/plane)</td>
</tr>
<tr>
<td>TMC_PLANE_INC</td>
<td>get inclination always with plane</td>
</tr>
</tbody>
</table>

- `Angle` out
  result of measuring the angle

- `iReturnCode` out
  return code, see Additional Codes

**See Also**  
TMC_DoMeasure, TMC_GetAngle

**Additional Codes in `iReturnCode`**
- **RC_OK**  
  Execution successful.
- **TMC_NO_FULL_CORRECTION**  
  The results are not corrected by all active sensors. Angle data are available.
  This message is to be considers as warning.
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.

Return Codes
RC_OK angle OK
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.

Example
Read the currently valid angle.
DIM Angles AS TMC_Angle
DIM iRetCode AS Integer
TMC_GetAngle_WInc(TMC_AUTO_INC, Angles, iRetCode)

6.3.8 TMC_QuickDist

Description Measure slope distance and angles.

Declaration TMC_QuickDist ( 
Angle AS TMC_HZ_V_ANG_type, 
Dist AS Distance, 
iReturnCode AS Integer )

Remarks The function measures the horizontal and vertical angle and in dependence of the configuration, the inclination.
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 active with the standard tracking measuring program already, the measuring program will not be 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). if 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_WInc(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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>out</td>
<td>measured Hz- and V-angle</td>
</tr>
<tr>
<td>Distance</td>
<td>out</td>
<td>measured slope-distance</td>
</tr>
<tr>
<td>iReturnCode</td>
<td>out</td>
<td>return code, see Additional Codes</td>
</tr>
</tbody>
</table>

See Also TMC_DoMeasure, TMC_GetAngle
### Additional Codes in iReturnCode

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Execution successful.</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. Angle data are available. This message is to be considered as warning.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>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 perform a forced incline measurement or switch off the incline. This message is to be considered as info.</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. Angle data are not available. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Angle measuring data are valid, but no distance data available.  (Possible reasons are: –time out period to short –target out of view) This message is to be considered as warning.</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>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 considered as warning.</td>
</tr>
</tbody>
</table>
### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>angle OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy. Angle data are not available. Repeat measurement.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>

### Example

Fast tracking with QuickDist. See example program TRACKING for more details.

```geo
DIM iRetCode AS Integer
DIM HzV AS TMC_HZ_V_ANG_Type
DIM dDist AS Distance

TMC_DoMeasure( TMC_CLEAR ) ' clear distances
```
' measurement loop
DO
  ' get measurement values
  TMC_QuickDist( HzV, dDist, iRetCode )
  IF iRetCode = RC_OK OR
     iRetCode = TMC_NO_FULL_CORRECTION OR
     iRetCode = TMC_ACCURACY_GUARANTEE THEN
    ' Angles and distance are valid
    ... 
  ELSE
    ' only Angles are valid
    ...
  END IF
LOOP UNTIL ....

' terminate
TMC_DoMeasure( TMC_CLEAR ) ' stop measurement

6.3.9 TMC_GetSimpleMea

Description  Gets the results of distance and angle measurement.

Declaration  TMC_GetSimpleMea( 
  Angles       AS TMC_HZ_V_ANG_Type,
  dSlopeDist   AS Double,
  iReturnCode  AS Integer )

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

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angles out</td>
<td>result of measuring: the angles</td>
</tr>
<tr>
<td>dSlopeDist out</td>
<td>slope distance [m]</td>
</tr>
<tr>
<td>iReturnCode out</td>
<td>return code, see Additional Codes</td>
</tr>
</tbody>
</table>

## See Also

TMC_DoMeasure

## Additional Codes in iReturnCode

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Angle OK</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. Angle and distance data are available. This message is to be considered as warning.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>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 considered as info.</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Angle values okay, but no valid distance. Perform a distance measurement.</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>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.</td>
</tr>
</tbody>
</table>
TMC_ANGLE_ACCURACY

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.

Return Codes

RC_OK Angle OK
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 aborted.

Example

This example measures the slope distance and angles.

```geo
DIM Angle AS Double
DIM dSlope AS Double
DIM RetCode AS Integer
TMC_GetSimpleMea( Angle, dSlope, RetCode )
```
6.3.10 TMC_Get/SetAngleFaceDef

Description
Gets and sets the current face definition.

Declaration
TMC_GetAngleFaceDef( eFaceDef AS Integer )
TMC_SetAngleFaceDef(
    byVal eFaceDef AS Integer )

Remarks
Note
No distance may exist for setting the face definition. Call TMC_DoMeasure(TMC_CLEAR) before this function.

Parameters

eFaceDef out/in TMC_FACE_NORMAL or TMC_FACE_TURN

See Also
-

Return Codes
RC_OK Completed successfully.
TMC_BUSY measurement system is busy (no valid results) or a distance exists

Example
The example reads the current definition and sets the opposite one.

DIM face AS TMC_FACE_DEF
TMC_GetAngleFaceDef(face)
IF (face = TMC_FACE_NORMAL) THEN
    TMC_SetAngleFaceDef(TMC_FACE_TURN)
ELSE
    TMC_SetAngleFaceDef(TMC_FACE_NORMAL)
END IF
6.3.11 TMC_Get/SetHzOffset

Description
Gets and sets the current horizontal offset.

Declaration
TMC_GetHzOffset( dHzOffset AS Double )
TMC_SetHzOffset( byVal dHzOffset AS Double )

Remarks
Note
No distance may exist for setting the Hz-offset. Call TMC_DoMeasure (TMC_CLEAR) before this function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dHzOffset</td>
<td>out/in</td>
<td>Horizontal offset in radiant.</td>
</tr>
</tbody>
</table>

See Also
-

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Completed successfully.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Measurement system is busy (no valid results) or a distance exists</td>
</tr>
</tbody>
</table>

Example
The example reads the current offsets and sets it to an increased value.

```
DIM off AS Double
TMC_GetHzOffset( off )
TMC_SetHzOffset( off + 1.0 )
```
6.3.12 TMC_Get/SetDistPpm

**Description**
Gets and sets the correction values for distance measurements.

**Declaration**

```
TMC_GetDistPpm( PpmCorr AS TMC_PPM_CORR_Type)  
TMC_SetDistPpm( PpmCorr AS TMC_PPM_CORR_Type)  
```

**Parameters**

- PpmCorr <out/in> Correction value for distance measurement.

**Return Codes**

- RC_OK: Completed successfully.
- TMC_BUSY: TMC is in use and cannot be changed.

**Example**

```
```

6.3.13 TMC_Get/SetHeight

**Description**
Gets and sets the current height of the reflector.

**Declaration**

```
TMC_GetHeight ( Height AS Double )  
TMC_SetHeight ( byVal Height AS Double )  
```

**Parameters**

- Height <out/in> Height of reflector in Meters.

**Return Codes**

- RC_OK: Completed successfully.
- TMC_BUSY: Measurement system is busy (no valid results)

**Example**

```
The example sets the reflector's height to the value of 1.0 m.
TMC_SetHeight( 1.0 )
```
6.3.14 TMC_Get/SetRefractiveCorr

Description: Gets and sets the refractive correction for measuring the distance.

Declaration:

```plaintext
TMC_GetRefractiveCorr (Refraction AS TMC_REFRACTION_Type)
TMC_SetRefractiveCorr (Refraction AS TMC_REFRACTION_Type)
```

Parameters:

- **Refraction** out/in: Refraction correction value(s).

Return Codes:

- **RC_OK**: Completed successfully.
- **TMC_BUSY**: Measurement system is busy (no valid results).

Example:

- 

6.3.15 TMC_Get/SetRefractiveMethod

Description: Gets and sets the method of refractive correction for measuring the distance.

Declaration:

```plaintext
TMC_GetRefractiveMethod (Method AS Integer)
TMC_SetRefractiveMethod (byVal Method AS Integer)
```

Parameters:

- **Method** out/in: Method of refraction calculation:
  - 1: method 1
  - 2: method 2
  - else: undefined

Return Codes:

- **RC_OK**: Completed successfully.
- **TMC_BUSY**: Measurement system is busy (no valid results)
6.3.16 TMC_Get/SetStation

Description
Gets and sets station co-ordinates.

Declaration
TMC_GetStation ( 
    Station AS TMC_STATION>Type )

TMC_SetStation ( 
    Station AS TMC_STATION_Type )

Remarks
Note
No distance may exist for setting a new station. Call TMC_DoMeasure(TMC_CLEAR) before this function.

Parameters
Station out/in Station co-ordinates.

Return Codes
RC_OK Completed successfully.
TMC_BUSY measurement system is busy (no valid results) or a distance exists.

Example
-

6.3.17 TMC_IfDistTapeMeasured

Description
Gets information about manual measurement.

Declaration
TMC_IfDistTapeMeasured ( 
    bTapeMeasured AS Logical )

Parameters
bTapeMeasured out TRUE: if measurement has been done by hand.
FALSE: if measurement has been done with EDM or if invalid.

Return Codes
RC_OK Completed successfully.

Example
-
### 6.3.18 TMC_SetHandDist

**Description**
Sets distance manually.

**Declaration**
```plaintext
TMC_SetHandDist(
    byVal dSlopeDistance AS Double,
    byVal dHgtOffset AS Double)
```

**Parameters**
- `dSlopeDistance` in slope distance [m]
- `dHgtOffset` in Height to measured point. [m]

**See Also**
- 

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Execution successful.</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. This message is to be considered as warning.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>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 considered as info.</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. At repeated occur call service.</td>
</tr>
</tbody>
</table>

### 6.3.19 TMC_SetDistSwitch

**Description**
Defines the distance measurement correction switches.

**Declaration**

```basic
TMC_SetDistSwitch(
    Switches AS TMC_DIST_SWITCH_Type
)
```

**Remarks**
This procedure sets the distance measurement correction switches.

**Parameters**

- **Switches**
  - **in**
  - Distance switches

**Return-Codes**

- **RC_OK**
  - Successful termination.

**See Also**
TMC_GetDistSwitch

### 6.3.20 TMC_GetDistSwitch

**Description**
Returns the distance measurement correction switches.

**Declaration**

```basic
TMC_GetDistSwitch(
    Switches AS TMC_DIST_SWITCH_Type
)
```

**Remarks**
This procedure returns the distance measurement correction switches.

**Parameters**

- **Switches**
  - **out**
  - Distance switches
6.3.21 TMC_SetOffsetDist

Description
Defines the distance measurement offset.

Declaration
TMC_SetOffsetDist(
    Offsets AS TMC_OFFSET_DIST_Type
)

Remarks
This procedure defines the offset to the prism pole. The dLengthVal defines the offset away from the prism pole, positive means in the line from instrument to prism. dCrossVal means right from the prism pole and dHeightVal means higher than prism pole.

Remarks
Note: No distance may exist for offset setting. Call TMC_DoMeasure(TMC_CLEAR) before this function.

Parameters
Offsets in Target point offset

Return-Codes
RC_OK Successful termination.
TMC_BUSY measurement system is busy (no valid results) or a distance exists.

See Also
TMC_GetOffsetDist, BAP_Offset, TMC_IfOffsetDistMeasured
6.3.22 TMC_GetOffsetDist

Description
Returns the distance measurement offset.

Declaration
TMC_GetOffsetDist(Offsets AS TMC_OFFSET_DIST_Type)

Remarks
This procedure returns the actual offset to the prism pole. The dLengthVal defines the offset away from the prism pole, positive means in the line from instrument to prism. dCrossVal means right from the prism pole and dHeightVal means higher than prism pole.

Parameters

Offsets out Target point offset

Return-Codes
RC_OK Successful termination.

See Also
TMC_SetOffsetDist, BAP_Offset, TMC_IfOffsetDistMeasured

6.3.23 TMC_IfOffsetDistMeasured

Description
Returns the EDM measurement mode.

Declaration
TMC_IfOffsetDistMeasured(lOffset AS Logical)

Remarks
This function returns TRUE if an offset is defined.

Parameters

lOffset out Offset is valid

Return-Codes
RC_OK Successful termination.

See Also
TMC_SetOffsetDist, TMC_GetOffsetDist, BAP_Offset
### 6.3.24 TMC_GetFace1

**Description**
Get face information of current telescope position.

**Declaration**

```
TMC_GetFace1( lFace1 AS Logical )
```

**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_GetFace1 call). Note that the instrument automatically turns into an inactive measurement state after a predefined timeout.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lFace1</td>
<td>out TRUE: Face I</td>
</tr>
<tr>
<td></td>
<td>FALSE: Face II</td>
</tr>
</tbody>
</table>

**Return-Codes**

- **RC_OK**
  Successful termination.

### 6.3.25 TMC_SetAngSwitch

**Description**
Defines the angle measurement correction switches.

**Declaration**

```
TMC_SetAngSwitch( Switches AS TMC_ANG_SWITCH_Type )
```

**Remarks**
This procedure sets the angle measurement correction switches.

**Note**
No distance may exist for setting the angle switches. Call TMC_DoMeasure( TMC_CLEAR ) before this function.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switches</td>
<td>in angular switches</td>
</tr>
</tbody>
</table>

**Return-Codes**

- **RC_OK**
  Successful termination.
- **TMC_BUSY**
  A distance exists

**See Also**

TMC_GetAngSwitch
Example
Change switches

```
DIM AngSwitches AS TMC_ANG_SWITCH_Type

TMC_DoMeasure( TMC_CLEAR ) ' clear distances
TMC_GetAngSwitch( AngSwitches )
AngSwitches.lInclineCorr = TRUE
AngSwitches.lCollimationCorr = FALSE
TMC_SetAngSwitch( AngSwitches )
```

6.3.26 TMC_GetAngSwitch

**Description**
Returns the angle measurement correction switches.

**Declaration**
```
TMC_GetAngSwitch(
    Switches AS TMC_ANG_SWITCH_Type )
```

**Remarks**
This procedure returns the actual angle measurement correction switches.

**Parameters**
- `Switches` in Angular switches

**Return-Codes**
- `RC_OK` Successful termination.

**See Also**
- `TMC_SetAngSwitch`

6.3.27 TMC_SetInclineSwitch

**Description**
Defines the compensator switch.

**Declaration**
```
TMC_SetAngSwitches( lOn AS Logical )
```

**Remarks**
This procedure enables or disables the dual axis compensator correction.

**Note**
No distance may exist for a switch setting. Call `TMC_DoMeasure(TMC_CLEAR)` before this function.

**Parameters**
- `lOn` in Switch
### 6.3.28 TMC_GetInclineSwitch

**Description**
Returns the compensator switch.

**Declaration**
```
TMC_GetInclineSwitches( lOn AS Logical )
```

**Remarks**
This procedure returns the dual axis compensator correction state.

**Parameters**
- `lOn` out Switch

**Return-Codes**
- RC_OK Successful termination.
- TMC_BUSY A distance exists

**See Also**
TMC_SetInclineSwitch

### 6.3.29 TMC_GetInclineStatus

**Description**
Returns the inclination compensator status.

**Declaration**
```
TMC_GetInclineStatus( iStatus AS Integer )
```

**Remarks**
This procedure returns status of the inclination sensor.

**Parameters**
- `iStatus` out
  - TMC_INC_OFF Incline-sensor is switched off
  - TMC_INC_OK Inclination is ok, recording is allowed
  - TMC_INC_TILT Incline-sensor is out of working area
  - TMC_INC_OLD Incline-values are not yet updated
TMC_INC_FAIL  Inclination -
measurement fails

Return-Codes

RC_OK  Successful termination.

See Also  TMC_SetInclineSwitch

Example  See example file „meas.gbs“.
### 6.4 FUNCTIONS FOR GSI

#### 6.4.1 Summarizing Lists of GSI Types and Procedures

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi_List</td>
<td>Array of GSI_WiDlg_Entry_Type.</td>
</tr>
<tr>
<td>GSI_Point_Coord_Type</td>
<td>Point co-ordinate data.</td>
</tr>
<tr>
<td>GSI_Rec_Id_List</td>
<td>Record mask array of integers (indicating WI-identifications).</td>
</tr>
<tr>
<td>GSI_WiDlg_Entry_Type</td>
<td>Dialog entry information.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedure Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_Coding</td>
<td>Starts the active coding function of the TPS system.</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>Returns if distance tracking is running.</td>
</tr>
<tr>
<td>GSI_CreateMDlg</td>
<td>Creates and shows the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_DefineMDlg</td>
<td>Defines the entries of the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_DefineRecMaskDlg</td>
<td>Defines the recording mask dialog.</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>Executes an automatic distance measurement.</td>
</tr>
<tr>
<td>GSI_ExecuteQCoding</td>
<td>Executes the Quick-Coding.</td>
</tr>
<tr>
<td>GSI_GetDataPath</td>
<td>Get the name of the file with the import data.</td>
</tr>
<tr>
<td>GSI_GetIndivNr</td>
<td>Fetches the individual point number.</td>
</tr>
<tr>
<td>GSI_GetLineSysMDlg</td>
<td>Gets the definition of a line in the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>Returns the number of the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>This routine returns the status for Quick-..</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_GetRecMask</td>
<td>Coding.</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>Get the definition and the format of a recording mask.</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>Returns the used recording mask.</td>
</tr>
<tr>
<td>GSI_GetRecPath</td>
<td>Returns the recording order for Quick-Coding.</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>Returns the recording path.</td>
</tr>
<tr>
<td>GSI_GetWiEntry</td>
<td>Fetches the running point number and the increment.</td>
</tr>
<tr>
<td>GSI_ImportCoordDlg</td>
<td>Get data from the Theodolite data pool.</td>
</tr>
<tr>
<td>GSI_IncPNumber</td>
<td>Show the co-ordinate import dialog.</td>
</tr>
<tr>
<td>GSI_IsRunningNr</td>
<td>Automatically point number increment.</td>
</tr>
<tr>
<td>GSI_ManCoordDlg</td>
<td>Queries if running number is being used.</td>
</tr>
<tr>
<td>GSI_Measure</td>
<td>Show the manual co-ordinate input dialog.</td>
</tr>
<tr>
<td>GSI_QuickSet</td>
<td>Entry point for measure and registration dialog (measure and registration).</td>
</tr>
<tr>
<td>GSI_RecordRecMask</td>
<td>Show the Quickset dialog.</td>
</tr>
<tr>
<td>GSI_SelectCode</td>
<td>Recording the given wi mask.</td>
</tr>
<tr>
<td>GSI_SetDataPath</td>
<td>This routine shows the codelist-coding dialog.</td>
</tr>
<tr>
<td>GSI_SetIndivNr</td>
<td>Set the file with the import data.</td>
</tr>
<tr>
<td>GSI_SetIVPtNrStatus</td>
<td>Sets the individual point number.</td>
</tr>
<tr>
<td>GSI_SetLineMDlg</td>
<td>Switches the individual point number mode on/off.</td>
</tr>
<tr>
<td>GSI_SetLineMDlgPar</td>
<td>Sets one line in the user definable measurement dialog to system parameter.</td>
</tr>
<tr>
<td>GSI_SetLineMDlgText</td>
<td>Sets a line in the user definable measurement dialog to an application parameter.</td>
</tr>
<tr>
<td>GSI_SetLineSysMDlg</td>
<td>Puts a textline into the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_SetMDlgNr</td>
<td>Sets a line in the system measurement dialog.</td>
</tr>
<tr>
<td></td>
<td>Sets the number of the system measurement dialog.</td>
</tr>
</tbody>
</table>
6. System Functions

### Procedure Name

<table>
<thead>
<tr>
<th>Procedure Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_SetQCodeMode</td>
<td>Sets the Quick-Coding mode.</td>
</tr>
<tr>
<td>GSI_SetRecMask</td>
<td>Set the definition and the format of a recording mask.</td>
</tr>
<tr>
<td>GSI_SetRecMaskNr</td>
<td>Set the used recording mask.</td>
</tr>
<tr>
<td>GSI_SetRecOrder</td>
<td>Sets the recording order for Quick-Coding.</td>
</tr>
<tr>
<td>GSI_SetRecPath</td>
<td>Defines the recording path.</td>
</tr>
<tr>
<td>GSI_SetRunningNr</td>
<td>Sets the running point number and increment.</td>
</tr>
<tr>
<td>GSI_SetWiEntry</td>
<td>Set data to the Theodolite data pool.</td>
</tr>
<tr>
<td>GSI_UpdateMDlg</td>
<td>Updates the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_UpdateMeasurement</td>
<td>Update the measurement data.</td>
</tr>
</tbody>
</table>

#### 6.4.2 Constants for WI values

Definitions for WI values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_PTNR</td>
<td>String</td>
<td>Point number</td>
</tr>
<tr>
<td>GSI_ID_FNR</td>
<td>Double</td>
<td>Serial number</td>
</tr>
<tr>
<td>GSI_ID_TYPE</td>
<td>String</td>
<td>Device type</td>
</tr>
<tr>
<td>GSI_ID_TIME_1</td>
<td>String</td>
<td>First time art</td>
</tr>
<tr>
<td>GSI_ID_TIME_2</td>
<td>String</td>
<td>Second time art</td>
</tr>
<tr>
<td>GSI_ID_HZ</td>
<td>Double</td>
<td>Horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_V</td>
<td>Double</td>
<td>Vertical angle</td>
</tr>
<tr>
<td>GSI_ID_NHZ</td>
<td>Double</td>
<td>Nominal horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_DHZ</td>
<td>Double</td>
<td>Difference horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_NV</td>
<td>Double</td>
<td>Nominal vertical angle</td>
</tr>
<tr>
<td>GSI_ID_DV</td>
<td>Double</td>
<td>Difference vertical angle</td>
</tr>
<tr>
<td>GSI_ID_SLOPE</td>
<td>Double</td>
<td>Slope distance</td>
</tr>
</tbody>
</table>

6-132
<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_HOR</td>
<td>Double</td>
<td>Horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_HGT</td>
<td>Double</td>
<td>Height difference</td>
</tr>
<tr>
<td>GSI_ID_NHOR</td>
<td>Double</td>
<td>Nominal horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_DHOR</td>
<td>Double</td>
<td>Difference horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_NHGT</td>
<td>Double</td>
<td>Nominal height difference</td>
</tr>
<tr>
<td>GSI_ID_DHGT</td>
<td>Double</td>
<td>Difference height difference</td>
</tr>
<tr>
<td>GSI_ID_NSLOPE</td>
<td>Double</td>
<td>Nominal slope distance</td>
</tr>
<tr>
<td>GSI_ID_DSLOPE</td>
<td>Double</td>
<td>Difference slope distance</td>
</tr>
<tr>
<td>GSI_ID_CODE</td>
<td>String</td>
<td>Code information</td>
</tr>
<tr>
<td>GSI_ID_CODE_1</td>
<td>String</td>
<td>Information 1</td>
</tr>
<tr>
<td>GSI_ID_CODE_2</td>
<td>String</td>
<td>Information 2</td>
</tr>
<tr>
<td>GSI_ID_CODE_3</td>
<td>String</td>
<td>Information 3</td>
</tr>
<tr>
<td>GSI_ID_CODE_4</td>
<td>String</td>
<td>Information 4</td>
</tr>
<tr>
<td>GSI_ID_CODE_5</td>
<td>String</td>
<td>Information 5</td>
</tr>
<tr>
<td>GSI_ID_CODE_6</td>
<td>String</td>
<td>Information 6</td>
</tr>
<tr>
<td>GSI_ID_CODE_7</td>
<td>String</td>
<td>Information 7</td>
</tr>
<tr>
<td>GSI_ID_CODE_8</td>
<td>String</td>
<td>Information 8</td>
</tr>
<tr>
<td>GSI_ID_CODE_9</td>
<td>String</td>
<td>Information 9</td>
</tr>
<tr>
<td>GSI_ID_PPMM</td>
<td>String</td>
<td>mm and ppm</td>
</tr>
<tr>
<td>GSI_ID_SIGMA</td>
<td>String</td>
<td>Distance count and deviation</td>
</tr>
<tr>
<td>GSI_ID_MM</td>
<td>Double</td>
<td>mm</td>
</tr>
<tr>
<td>GSI_ID_PPM</td>
<td>Double</td>
<td>ppm</td>
</tr>
<tr>
<td>GSI_ID_REM_1</td>
<td>String</td>
<td>Remark 1</td>
</tr>
<tr>
<td>GSI_ID_REM_2</td>
<td>String</td>
<td>Remark 2</td>
</tr>
<tr>
<td>GSI_ID_REM_3</td>
<td>String</td>
<td>Remark 3</td>
</tr>
<tr>
<td>GSI_ID_REM_4</td>
<td>String</td>
<td>Remark 4</td>
</tr>
<tr>
<td>GSI_ID_REM_5</td>
<td>String</td>
<td>Remark 5</td>
</tr>
<tr>
<td>GSI_ID_REM_6</td>
<td>String</td>
<td>Remark 6</td>
</tr>
<tr>
<td>GSI_ID_REM_7</td>
<td>String</td>
<td>Remark 7</td>
</tr>
<tr>
<td>GSI_ID_REM_8</td>
<td>String</td>
<td>Remark 8</td>
</tr>
<tr>
<td>GSI_ID_REM_9</td>
<td>String</td>
<td>Remark 9</td>
</tr>
<tr>
<td>GSI_ID_E</td>
<td>Double</td>
<td>East co-ordinate</td>
</tr>
<tr>
<td>Name</td>
<td>Data Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>GSI_ID_N</td>
<td>Double</td>
<td>North co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_H</td>
<td>Double</td>
<td>Height</td>
</tr>
<tr>
<td>GSI_ID_E0</td>
<td>Double</td>
<td>East station co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_N0</td>
<td>Double</td>
<td>North station co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_H0</td>
<td>Double</td>
<td>Station height</td>
</tr>
<tr>
<td>GSI_ID_HR</td>
<td>Double</td>
<td>Reflector height</td>
</tr>
<tr>
<td>GSI_ID_HI</td>
<td>Double</td>
<td>Instrument height</td>
</tr>
<tr>
<td>GSI_ID_INDIV</td>
<td>String</td>
<td>Individual point number</td>
</tr>
<tr>
<td>GSI_ID_PTLA</td>
<td>String</td>
<td>Number of the last recorded point</td>
</tr>
<tr>
<td>GSI_ID_STEP</td>
<td>Double</td>
<td>Increment of the running point number</td>
</tr>
<tr>
<td>GSI_ID_SPTNR</td>
<td>String</td>
<td>Station point number</td>
</tr>
<tr>
<td>GSI_ID_SHZ</td>
<td>Double</td>
<td>Hz angle with no sign change</td>
</tr>
<tr>
<td>GSI_ID_CD_DSC</td>
<td>String</td>
<td>Code description</td>
</tr>
<tr>
<td>GSI_ID_PTCD_DSC</td>
<td>String</td>
<td>Point code description</td>
</tr>
<tr>
<td>GSI_ID_PV_CD</td>
<td>String</td>
<td>Preview code</td>
</tr>
<tr>
<td>GSI_ID_PV_PTCD</td>
<td>String</td>
<td>Preview point code</td>
</tr>
<tr>
<td>GSI_ID_ACT_PTID</td>
<td>String</td>
<td>Actual point ID</td>
</tr>
<tr>
<td>GSI_ID_BACKID</td>
<td>String</td>
<td>Backside ID</td>
</tr>
<tr>
<td>GSI_ID_APPDATA0</td>
<td>String/Double</td>
<td>Application data 0</td>
</tr>
<tr>
<td>GSI_ID_APPDATA1</td>
<td>String/Double</td>
<td>Application data 1</td>
</tr>
<tr>
<td>GSI_ID_APPDATA2</td>
<td>String/Double</td>
<td>Application data 2</td>
</tr>
<tr>
<td>GSI_ID_APPDATA3</td>
<td>String/Double</td>
<td>Application data 3</td>
</tr>
<tr>
<td>GSI_ID_APPDATA4</td>
<td>String/Double</td>
<td>Application data 4</td>
</tr>
<tr>
<td>GSI_ID_APPDATA5</td>
<td>String/Double</td>
<td>Application data 5</td>
</tr>
<tr>
<td>GSI_ID_APPDATA6</td>
<td>String/Double</td>
<td>Application data 6</td>
</tr>
<tr>
<td>GSI_ID_APPDATA7</td>
<td>String/Double</td>
<td>Application data 7</td>
</tr>
<tr>
<td>GSI_ID_APPDATA8</td>
<td>String/Double</td>
<td>Application data 8</td>
</tr>
<tr>
<td>GSI_ID_APPDATA9</td>
<td>String/Double</td>
<td>Application data 9</td>
</tr>
<tr>
<td>GSI_ID_APPDATA10</td>
<td>String/Double</td>
<td>Application data 10</td>
</tr>
<tr>
<td>GSI_ID_APPDATA11</td>
<td>String/Double</td>
<td>Application data 11</td>
</tr>
<tr>
<td>GSI_ID_FS_SCALE</td>
<td>Double</td>
<td>Free station scale</td>
</tr>
</tbody>
</table>
6.4.3 Constants for Measurement Dialog Definition

Definition of (user definable) application parameters for measurement dialogs, either Double or String. See also GSI_SetLineMDlgPar and GSI_SetLineMDlgText.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_AppData0</td>
<td>Application parameter 0</td>
</tr>
<tr>
<td>GSI_PAR_AppData1</td>
<td>Application parameter 1</td>
</tr>
<tr>
<td>GSI_PAR_AppData2</td>
<td>Application parameter 2</td>
</tr>
<tr>
<td>GSI_PAR_AppData3</td>
<td>Application parameter 3</td>
</tr>
<tr>
<td>GSI_PAR_AppData4</td>
<td>Application parameter 4</td>
</tr>
<tr>
<td>GSI_PAR_AppData5</td>
<td>Application parameter 5</td>
</tr>
<tr>
<td>GSI_PAR_AppData6</td>
<td>Application parameter 6</td>
</tr>
<tr>
<td>GSI_PAR_AppData7</td>
<td>Application parameter 7</td>
</tr>
<tr>
<td>GSI_PAR_AppData8</td>
<td>Application parameter 8</td>
</tr>
<tr>
<td>GSI_PAR_AppData9</td>
<td>Application parameter 9</td>
</tr>
<tr>
<td>GSI_PAR_AppData10</td>
<td>Application parameter 10</td>
</tr>
</tbody>
</table>
### Name | Meaning
--- | ---
GSI_PAR_AppData11 | Application parameter 11

Definition of system (defined) parameters for measurement dialogs. See also GSI_SetLineSysMDlg and GSI_SetLineMDlg.

### Name | Meaning
--- | ---
GSI_PAR_AddConst | Prism constant
GSI_PAR_Attrib1 | Point Code Attribute 1
GSI_PAR_Attrib2 | Point Code Attribute 2
GSI_PAR_Attrib3 | Point Code Attribute 3
GSI_PAR_Attrib4 | Point Code Attribute 4
GSI_PAR_Attrib5 | Point Code Attribute 5
GSI_PAR_Attrib6 | Point Code Attribute 6
GSI_PAR_Attrib7 | Point Code Attribute 7
GSI_PAR_Attrib8 | Point Code Attribute 8
GSI_PAR_AvgMeasNo | Maximal number of distance measurements of the average mode
GSI_PAR_BacksideId | Last used Backside
GSI_PAR_Code | Last used Code
GSI_PAR_CodeDescr | Last used free Code Description
GSI_PAR_CodeList | Codelist management (select, create etc)
GSI_PAR_CodeListSelect | Codelist selection (of an existing codelist)
GSI_PAR_DataJobSelect | Data job selection (of an existing job)
GSI_PAR_DataJob | Data job management (select, create etc)
GSI_PAR_TargetEast | Target point Easting
GSI_PAR_DistMeasProg | EDM measurement program selection.
GSI_PAR_Date | Current date of the instrument. The displayed format depends on the setting of the parameter "Date form."
GSI_PAR_DisplayMask | Select display mask for standard measuring dialog. Max. 3 displaymasks can be defined for this dialog. The displaymasks can also be changed with the system function "Next Displaymask".
<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_TargetElev</td>
<td>Target point Elevation</td>
</tr>
<tr>
<td>GSI_PAR_ElevDiff</td>
<td>Elevation difference</td>
</tr>
<tr>
<td>GSI_PAR_HalfLineSpace</td>
<td>This item can be used to display a half line space in order to separate or group lines on instrument screen.</td>
</tr>
<tr>
<td>GSI_PAR_DistHoriz</td>
<td>Horizontal distance</td>
</tr>
<tr>
<td>GSI_PAR_AngleHz</td>
<td>Hz-Angle</td>
</tr>
<tr>
<td>GSI_PAR_PointIdIncr</td>
<td>defines the increment step. It is used to increment the Target Point Id after recording a target point.</td>
</tr>
<tr>
<td>GSI_PAR_IndivPointId</td>
<td>Individual point identifier</td>
</tr>
<tr>
<td>GSI_PAR_Info1</td>
<td>Shows the Free Code Info 1</td>
</tr>
<tr>
<td>GSI_PAR_Info2</td>
<td>Shows the Free Code Info 2</td>
</tr>
<tr>
<td>GSI_PAR_Info3</td>
<td>Shows the Free Code Info 3</td>
</tr>
<tr>
<td>GSI_PAR_Info4</td>
<td>Shows the Free Code Info 4</td>
</tr>
<tr>
<td>GSI_PAR_Info5</td>
<td>Shows the Free Code Info 5</td>
</tr>
<tr>
<td>GSI_PAR_Info6</td>
<td>Shows the Free Code Info 6</td>
</tr>
<tr>
<td>GSI_PAR_Info7</td>
<td>Shows the Free Code Info 7</td>
</tr>
<tr>
<td>GSI_PAR_Info8</td>
<td>Shows the Free Code Info 8</td>
</tr>
<tr>
<td>GSI_PAR_InstrHeight</td>
<td>Instrument Height (hi)</td>
</tr>
<tr>
<td>GSI_PAR_LastPointId</td>
<td>Last recorded target point identifier</td>
</tr>
<tr>
<td>GSI_PAR_MeasJobSelect</td>
<td>Measurement Job selection (of an existing Job or RS232 for online recording)</td>
</tr>
<tr>
<td>GSI_PAR_MeasJob</td>
<td>Measurement Job management (select, create, etc.)</td>
</tr>
<tr>
<td>GSI_PAR_NS</td>
<td>Number of measurements and standard deviation</td>
</tr>
<tr>
<td>GSI_PAR_TargetNorth</td>
<td>Target point Northing</td>
</tr>
<tr>
<td>GSI_PAR_OffsetCross</td>
<td>Cross Offset</td>
</tr>
<tr>
<td>GSI_PAR_OffsetElev</td>
<td>Offset Elevation</td>
</tr>
<tr>
<td>Name</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_PAR_OffsetLength</td>
<td>Offset Length</td>
</tr>
<tr>
<td>GSI_PAR_OffsetMode</td>
<td>Defines the resetting of the offset</td>
</tr>
<tr>
<td>GSI_PAR_PointCode</td>
<td>Actual Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_PointId</td>
<td>Actual Target point identifier, running or individual. The Value and the display text changes if an individual number is set.</td>
</tr>
<tr>
<td>GSI_PAR_PpmAtm</td>
<td>ppm atmospheric</td>
</tr>
<tr>
<td>GSI_PAR_PpmGeom</td>
<td>ppm geometric</td>
</tr>
<tr>
<td>GSI_PAR_PpmTotal</td>
<td>Total ppm</td>
</tr>
<tr>
<td>GSI_PAR_PpmMm</td>
<td>Total ppm and prism constant</td>
</tr>
<tr>
<td>GSI_PAR_PrevCode</td>
<td>Shows the second last used Code</td>
</tr>
<tr>
<td>GSI_PAR_PrevPointCode</td>
<td>Last used Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_PointCodeDescr</td>
<td>Shows the Point Code Description of the actual Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_RecMask</td>
<td>Selected Recording mask for target point measurements</td>
</tr>
<tr>
<td>GSI_PAR_Ref1Height</td>
<td>Reflector height (hr)</td>
</tr>
<tr>
<td>GSI_PAR_Ref1Name</td>
<td>Used reflector type</td>
</tr>
<tr>
<td>GSI_PAR_Ref1Selection</td>
<td>reflector type selection. If there are user defined prism, then they will be added to this list. The User Ref1..User Ref3 are only valid, if these user definable prisms are defined.</td>
</tr>
<tr>
<td>GSI_PAR_RUNNINGPointId</td>
<td>Running target point identifier</td>
</tr>
<tr>
<td>GSI_PAR_DistSlope</td>
<td>Slope distance</td>
</tr>
<tr>
<td>GSI_PAR_StationId</td>
<td>Identifies the Station</td>
</tr>
<tr>
<td>GSI_PAR_StationEast</td>
<td>Station Easting</td>
</tr>
<tr>
<td>GSI_PAR_StationElev</td>
<td>Station Elevation</td>
</tr>
<tr>
<td>GSI_PAR_StationNorth</td>
<td>Station Northing</td>
</tr>
<tr>
<td>GSI_PAR_TargetType</td>
<td>Definition of the target type (Reflector / reflectorless)</td>
</tr>
<tr>
<td>GSI_PAR_Time</td>
<td>Current time of the instrument. The displayed format depends on the setting of the parameter &quot;Time form.&quot;</td>
</tr>
</tbody>
</table>
6. System Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_AngleV</td>
<td>V-Angle</td>
</tr>
<tr>
<td>GSI_PAR_VangleFormat</td>
<td>Vertical angle display format: Zenith angle = 0gon for zenith, angles are positive. Elev. angle = 0gon for horizontal, (+) above horizon and (-) below horizon. Elev. angle% = 0% for horizon, 100% for 50gon. V-angle is displayed (+) above and (-) below horizon but as percentage of the gradient.</td>
</tr>
<tr>
<td>GSI_PAR_NONE</td>
<td>Designates a line that is unused.</td>
</tr>
</tbody>
</table>

6.4.4 Relationship of GSI_ID’s to GSI_PAR’s

In general we can distinguish between two data value pools who are able to store values in it. Some of these values are shared between the two pools.

GSI_ID_-Ids describe the values which can be stored and requested in the (WI) data value pool. GSI_PAR_-Ids describe the values which can be used for displaying in a measurement dialog. Their sets of id’s are not associated directly in all cases. Moreover their sets of Id’s can be distinguished in their meaning.

Association in this context means that both pools, the data value pool and the data display pool, share their values directly. Nonassociated values are unique to either the data value pool or the data display pool.

Many of the GSI_IDs are recordable. Two types of recordable Ids can be distinguished:

a) Measurement block (“Meas”) (has to start with a GSI_ID_PTNR)
b) Code block (“Code”) (has to start with a GSI_ID_CODE)

They may not be mixed.

<table>
<thead>
<tr>
<th>Record-able</th>
<th>GSI_ID_-Ids</th>
<th>GSI_PAR_-Ids</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GSI_ID_NHZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSI_ID_DHZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSI_ID_NV</td>
<td></td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_FNR</td>
<td>GSI_PAR_SerialNr</td>
</tr>
<tr>
<td>------</td>
<td>--------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_TYPE</td>
<td>GSI_PAR_InstrType</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_TIME_1</td>
<td>See GSI_PAR_Date</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_TIME_2</td>
<td>See GSI_PAR_Time</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HZ</td>
<td>GSI_PAR_AngleHz</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_V</td>
<td>GSI_PAR_AngleV</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_SLOPE</td>
<td>GSI_PAR_DistSlope</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HOR</td>
<td>GSI_PAR_DistHoriz</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HGT</td>
<td>GSI_PAR_ElevDiff</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_PPMM</td>
<td>GSI_PAR_PmmMm</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_SIGMA</td>
<td>GSI_PAR_NS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meas</th>
<th>GSI_ID_PTNR</th>
<th>GSI_PAR_RunningPointId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meas</td>
<td>GSI_ID_FNR</td>
<td>GSI_PAR_SerialNr</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_TYPE</td>
<td>GSI_PAR_InstrType</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_TIME_1</td>
<td>See GSI_PAR_Date</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_TIME_2</td>
<td>See GSI_PAR_Time</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HZ</td>
<td>GSI_PAR_AngleHz</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_V</td>
<td>GSI_PAR_AngleV</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_SLOPE</td>
<td>GSI_PAR_DistSlope</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HOR</td>
<td>GSI_PAR_DistHoriz</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HGT</td>
<td>GSI_PAR_ElevDiff</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_PPMM</td>
<td>GSI_PAR_PmmMm</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_SIGMA</td>
<td>GSI_PAR_NS</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_MM</td>
<td>GSI_PAR_AddConst</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_PPM</td>
<td>GSI_PAR_PpmTotal</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_1</td>
<td>GSI_PAR_Info1</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_2</td>
<td>GSI_PAR_Info2</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_3</td>
<td>GSI_PAR_Info3</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_4</td>
<td>GSI_PAR_Info4</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_5</td>
<td>GSI_PAR_Info5</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_6</td>
<td>GSI_PAR_Info6</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_7</td>
<td>GSI_PAR_Info7</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_8</td>
<td>GSI_PAR_Info8</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_9</td>
<td>GSI_PAR_Info9</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_E</td>
<td>GSI_PAR_TargetEast</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_N</td>
<td>GSI_PAR_TargetNorth</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_H</td>
<td>GSI_PAR_TargetElev</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_E0</td>
<td>GSI_PAR_StationEast</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_N0</td>
<td>GSI_PAR_StationNorth</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_H0</td>
<td>GSI_PAR_StationElev</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HR</td>
<td>GSI_PAR_ReflHeight</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HI</td>
<td>GSI_PAR_InstrHeight</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE</td>
<td>GSI_PAR_Attrib1</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_1</td>
<td>GSI_PAR_Attrib2</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_2</td>
<td>GSI_PAR_Attrib3</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_3</td>
<td>GSI_PAR_Attrib4</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_4</td>
<td>GSI_PAR_Attrib5</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_5</td>
<td>GSI_PAR_Attrib6</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_6</td>
<td>GSI_PAR_Attrib7</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_7</td>
<td>GSI_PAR_Attrib8</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_8</td>
<td>GSI_PAR_Attrib9</td>
</tr>
</tbody>
</table>

GSI_ID_APPDATA0 are for the purpose of exchanging data between applications and between application and MDlg. They cannot be recorded. Both can be of the form GSI_ASCII or GSI_DOUBLE.
### GSI Function Reference

#### 6. System Functions

<table>
<thead>
<tr>
<th>GSI_ID_APPDATA0</th>
<th>GSI_PAR_APPDATA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_APPDATA1</td>
<td>GSI_PAR_APPDATA1</td>
</tr>
<tr>
<td>GSI_ID_APPDATA2</td>
<td>GSI_PAR_APPDATA2</td>
</tr>
<tr>
<td>GSI_ID_APPDATA3</td>
<td>GSI_PAR_APPDATA3</td>
</tr>
<tr>
<td>GSI_ID_APPDATA4</td>
<td>GSI_PAR_APPDATA4</td>
</tr>
<tr>
<td>GSI_ID_APPDATA5</td>
<td>GSI_PAR_APPDATA5</td>
</tr>
<tr>
<td>GSI_ID_APPDATA6</td>
<td>GSI_PAR_APPDATA6</td>
</tr>
<tr>
<td>GSI_ID_APPDATA7</td>
<td>GSI_PAR_APPDATA7</td>
</tr>
<tr>
<td>GSI_ID_APPDATA8</td>
<td>GSI_PAR_APPDATA8</td>
</tr>
<tr>
<td>GSI_ID_APPDATA9</td>
<td>GSI_PAR_APPDATA9</td>
</tr>
<tr>
<td>GSI_ID_APPDATA10</td>
<td>GSI_PAR_APPDATA10</td>
</tr>
<tr>
<td>GSI_ID_APPDATA11</td>
<td>GSI_PAR_APPDATA11</td>
</tr>
</tbody>
</table>

### Special Ids

<table>
<thead>
<tr>
<th>GSI_ID_NONE</th>
<th>GSI_PAR_NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_EMPTY</td>
<td>GSI_PAR_EMPTY</td>
</tr>
<tr>
<td>GSI_ID_UNKNOWN</td>
<td>GSI_PAR_UNKNOWN</td>
</tr>
</tbody>
</table>

The set of GSI_PAR-ids is not complete in this table. There exist several more Ids, which can be used for displaying.

### 6.4.5 Data Structures for GSI Functions

**GSI_WiDlg_Entry_Type**: Dialog entry information

**Description**: This data structure is used to store information about the entries (data fields) of the WI dialog.

```plaintext
TYPE GSI_WiDlg_Entry_Type
    iId AS Integer

iId
The identifier of the dialog entry. For possible value see WI constants.
```

6-142 TPS1100-Version 2.10
iDataType AS Integer

The type of the date stored in dValue or sValue. For possible value see table below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ASCII</td>
<td>ASCII data (stored in sValue)</td>
</tr>
<tr>
<td>GSI_ASCII_SIGN</td>
<td>signed ASCII data (stored in sValue)</td>
</tr>
<tr>
<td>GSI_DOUBLE</td>
<td>double data (stored in dValue)</td>
</tr>
</tbody>
</table>

lValid AS Logical

TRUE if the value is valid.

dValue AS Double

Data if value is of type Double.

sValue AS String10

Data if value is of type String.

END GSI_WiDlg_Entry_Type

Wi_List: An array of GSI_WiDlg_Entry_Type

Description
This array consists of GSI_MAX_REC_WI elements of the type GSI_WiDlg_Entry_Type.

GSI_Rec_Id_List: An array of integers (indicating WI-identifications)

Description
This array consists of GSI_MAX_REC_WI elements of the type Integer. It is used to define the recorded values (recmask).

GSI_Point_Coord_Type: Point co-ordinate data

Description
This data structure is used to store a point name and its co ordinates.

```plaintext
TYPE GSI_Point_Coord_Type
   sPtNr    AS String10  point number
   dEast    AS Double    east co-ordinate
   dNorth   AS Double    north co-ordinate
   dHeight  AS Double    height co-ordinate
   lPtNrValid AS Logical TRUE if point number is valid
   lEValid  AS Logical   TRUE if east co-ordinate is valid
   lNValid  AS Logical   TRUE if north co-ordinate is valid
END GSI_Point_Coord_Type
```
6.4.6  GSI_GetRunningNr

Description  Fetches the running point number and the increment.

Declaration  GSI_GetRunningNr( sPntId AS String20, sPntIncr AS String20 )

Remarks  Fetches the running point number and increment for it.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sPntId</td>
<td>String20</td>
<td>the running point number</td>
</tr>
<tr>
<td>sPntIncr</td>
<td>String20</td>
<td>the increment for the running point number</td>
</tr>
</tbody>
</table>

See Also  GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr, GSI_IsRunningNr

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>successful</td>
</tr>
</tbody>
</table>

Example

```plaintext
DIM sPntId AS String20
DIM sPntInc AS String20
GSI_GetRunningNr( sPntId, sPntInc )
```
6.4.7 GSI_SetRunningNr

Description
Sets the running point number and increment.

Declaration
GSI_SetRunningNr(
  BYVAL sPntId AS String20,
  BYVAL sPntIncr AS String20 )

Remarks
Sets the running point number and the increment for it. The running point number mode is switched on.

Parameters
sPntId in
  The user running point number.

sPntIncr in
  The increment for the user point running number.

See Also
GSI_GetRunningNr, GSI_GetIndivNr,
GSI_SetIndivNr, GSI_IsRunningNr

Return-Codes
RC_OK
  successful

Example
DIM sPntId AS String20
DIM sPntInc AS String20
GSI_SetRunningNr( sPntId, sPntInc )

6.4.8 GSI_GetIndivNr

Description
Fetches the individual point number.

Declaration
GSI_GetIndivNr( sPntId AS String20 )

Remarks
Fetches the individual point number.

Parameters
sPntId out
  The user-defined individual point number.

See Also
GSI_GetRunningNr, GSI_SetRunningNr,
GSI_SetIndivNr, GSI_IsRunningNr
6.4.9 GSI_SetIndivNr

Description: Sets the individual point number.

Declaration: 

GSI_SetIndivNr( BYVAL sPntId AS String20 )

Remarks: Sets the individual point number. After this call, the running point number mode is switched to the individual point number. This mode will be active until replaced by a running number or until the next save.

Parameters:

sPntId in The user-defined individual point number.

See Also:
GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_IsRunningNr

Return-Codes:

RC_OK successful

Example:

DIM sPntId AS String20
GSI_SetIndivNr( sPntId )

6.4.10 GSI_IsRunningNr

Description: Queries if running number is being used.

Declaration: 

GSI_IsRunningNr( lRunningOn AS Logical )

Remarks: If the running number is active the parameter will forced to TRUE otherwise to FALSE.
Parameters

lRunningOn  out  information about the running point number

See Also  GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr

Return-Codes

RC_OK  successful

Example

DIM lRunningOn AS Logical
GSI_IsRunningNr( lRunningOn )

6.4.11  GSI_SetIvPtNrStatus

Description  Switches the individual point number mode on/off.

Declaration  GSI_SetIvPtNrStatus(
            BYVAL lSwitch AS Logical )

Remarks  Switch the individual point number on or off. When point number is shown in the display the number will change.

Parameters

lSwitch  in  switch for the individual point-number (TRUE = on, FALSE = off)

See Also  GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr, GSI_IsRunningNr

Return-Codes

RC_OK  successful

Example  GSI_SetIvPtNrStatus( FALSE )
6.4.12 GSI_IncPNumber

Description: Automatically point number increment.

Declaration: GSI_IncPNumber()

Remarks: This function increments the running alphanumeric point number.

Parameters: none

See Also: GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr

Return Codes

RC_IVRESULT Point number is not incremented, possible reasons could be:
  wrong alphanumeric characters in point number
  alphanumeric characters in step
  overflow on a alphanumerically char step is longer as the point number

Example:

GSI_IncPNumber()

6.4.13 GSI_Coding

Description: Starts the active coding function of the TPS system.

Declaration: GSI_Coding( BYVAL Caption AS _Token)

Remarks: This routine starts the active coding function of the TPS system. Since there exist three possible locations, the TPS system follows a default ordering rule to invoke one of the programs. First it checks if there is an appropriate set up GeoBASIC coding program. If yes it will be executed, otherwise it examines the codelist management if a codelist is selected. If yes then the codelist will be opened, otherwise the standard coding will be activated.

Parameters

Caption in The left caption string of the dialog.
6.4.14 GSI_SelectCode

Description
This routine shows the codelist-coding dialog

Declaration
GSI_SelectCode( BYVAL Caption AS _Token)

Remarks
This routine starts the codelist-coding function of the TPS system. It will be executed only if a valid codelist is selected.

Parameters
Caption in The left caption string of the dialog.

Return-Codes
RC_OK successful
RC_ABORT Coding was aborted by pressing of the ESC-button
RC_ABORT_APPL Coding was aborted by pressing of the QUIT-button
COD_RC_LIST_NOT_VALID No valid codelist selected

Example
See example file „meas.gbs“.

6.4.15 GSI_GetQCodeAvailable

Description
This routine returns the status for Quick-Coding.

Declaration
GSI_GetQCodeAvailable(lAvailable As Logical, lEnabled As Logical)
Remarks  This routine returns if a valid codelist is selected and if Quick-Coding is enabled or not.

Parameters  
- **lAvailable** out  TRUE: a valid codelist is selected.
- **lEnabled** out  TRUE: Quick-Coding is activated

See Also  GSI_SetQCodeMode, GSI_ExecQCoding

Return-Codes  
- **RC_OK**  successful

Example  See example file „meas_od.gbs“.

### 6.4.16 GSI_SetQCodeMode

**Description**  Sets the Quick-Coding mode.

**Declaration**  
```vbscript
GSI_SetQCodeMode(BYVAL lEnabled As Logical)
```

**Remarks**  This routine enables or disables the Quick-Coding. It can be only activated if a valid codelist is selected (see GSI_GetQCodeAvailable)

**Parameters**  
- **lEnabled** in  TRUE: enable Quick-Coding

**See Also**  GSI_GetQCodeAvailable, GSI_ExecQCoding

**Return-Codes**  
- **RC_OK**  successful

**Example**  See example file „meas.gbs“.

### 6.4.17 GSI_ExecQCoding

**Description**  Executes the Quick-Coding.
Declaration

GSI_ExecQCoding(
    BYVAL lRecEnable AS Logical
    iButtonId AS Integer,
    lNewCode AS Logical)

Remarks

This routine executes the Quick-Coding. If Quick-Coding is enabled, it checks the button iButtonId and searches the corresponding code. If the selected code needs mandatory attributes, it shows the coding dialog. As successful coding is indicated by lNewCode=TRUE. The results are stored in the Theodolite data pool (see GSI_GetWiEntry).

If lRecEnable=TRUE, this routine executes the ALL-button functionality too, it measures a distance and records the results. The recording order (measurement block – code block or vice versa) depends on the system setting (see GSI_GetRecOrder).

If lRecEnable=FALSE, this routine forces no new distance measurement and there is no recording.

Parameters

lRecEnable in
    TRUE: Quick-Coding including distance measurement. It records a code- and a measurement-block in the correct order.
    FALSE: Quick-Coding without measurement and without recording

iButtonId inout
    In: Pressed button.
    Out: If a Quick-Coding was possible, iButtonId is changed to MMI_NO_KEY, otherwise it is unchanged

lNewCode out
    TRUE: Quick-Coding was successful

See Also

GSI_GetQCodeAvailable, GSI_SetQCodeMode, GSI_SetRecOrder

Return-Codes

RC_OK successful

Example

See example files „meas.gbs“ and „meas_od.gbs“..
6.4.18  GSI_SetRecOrder

**Description**  
Sets the recording order for Quick-Coding.

**Declaration**  
GSI_SetRecOrder(BYVAL lCodeFirst As Logical)

**Remarks**  
This routine defines the recording order for Quick-Coding.

  * If lCodeFirst=TRUE, then the code-block will be recorded before the measurement block.

**Parameters**

  * lCodeFirst
    * in
    * TRUE: code-block before measurement block

**See Also**

  * GSI_GetRecOrder, GSI_ExecQCoding

**Return-Codes**

  * RC_OK  
  * successful

**Example**

  * See example file „meas_od.gbs“.

6.4.19  GSI_GetRecOrder

**Description**  
Returns the recording order for Quick-Coding.

**Declaration**  
GSI_GetRecOrder(lCodeFirst As Logical)

**Remarks**  
This routine returns the recording order for Quick-Coding.

  * If lCodeFirst=TRUE, then the code-block will be recorded before the measurement block.

**Parameters**

  * lCodeFirst
    * out
    * TRUE: code-block before measurement block

**See Also**

  * GSI_SetRecOrder, GSI_ExecQCoding

**Return-Codes**

  * RC_OK  
  * successful

**Example**

  * See example file „meas_od.gbs“.
6.4.20 GSI_QuickSet

**Description**
Shows the Quickset dialog.

**Declaration**
GSI_QuickSet(BYVAL sCaptionLeft AS _Token)

**Remarks**
This procedure shows Quickset for station setting.

**Parameters**
- sCaptionLeft in Left caption for the Quickset dialog

**Return-Codes**
- RC_OK Successful termination.

**Example**
Show the dialog:
GSI_QuickSet ( "BASIC" )

6.4.21 GSI_SetRecPath

**Description**
Defines the recording path for the measurements.

**Declaration**
GSI_SetRecPath(
    BYVAL iPathInfo AS Integer,
    BYVAL sFileName AS FileName,
    BYVAL sFilePath AS FilePath )

**Remarks**
This procedure defines where the measurements will be recorded. If iPathInfo is set to GSI_INTERFACE, then the measurements will be sent to the RS232 line and the other parameters are not be interpreted. If iPathInfo is set to GSI_EXTERNAL, then sFileName defines the filename i.e. "MeasJob.GSI" and sFilePath defines the file-path, i.e. "A:\\GSI".

**Parameters**
- iPathInfo in Defines where the data are recorded
- sFileName in Valid Filename (8+3 format)
- sFilePath in file-path
Return-Codes

RC_OK Successful termination.

See Also GSI_GetRecPath

Example

This example shows the actual recording path and set it to the RS232 line:

```basic
DIM sFile As FileName
DIM sPath As FilePath
DIM iPathInfo As Integer

GSI_GetRecPath(iPathInfo, sFile, sPath)
IF iPathInfo = GSI_EXTERNAL THEN
    MMI_PrintStr(0, 1, "RecFile-CARD: " + sFile, TRUE)
    MMI_PrintStr(0, 2, " Path: " + sPath, TRUE)
ELSE
    MMI_PrintStr(0, 1, "RecPath - serial line", TRUE)
END IF
GSI_SetRecPath( GSI_INTERFACE, sFile, sPath)
```

### 6.4.22 GSI_GetRecPath

**Description**

Returns the recording path for the measurements.

**Declaration**

```basic
GSI_GetRecPath( 
    iPathInfo AS Integer, 
    sFileName AS FileName, 
    sFilePath AS FilePath )
```

**Remarks**

This procedure returns where the measurements will be recorded. If `iPathInfo = GSI_INTERFACE`, then the measurements will be sent to the RS232 line and the other parameters are not valid. If `iPathInfo = GSI_EXTERNAL`, then `sFileName` defines the filename i.e. "MeasJob.GSI" and `sFilePath` defines the file path i.e. "A:\\GSI".
6.4.23 GSI_SetDataPath

Description
Set the file with the import data.

Declaration
GSI_SetDataPath(  
    BYVAL iPathInfo AS Integer,  
    BYVAL sFileName AS FileName,  
    BYVAL sFilePath AS FilePath )

Remarks
This procedure sets the file from which data will be imported. Only GSI_EXTERNAL is valid for the iPathInfo. sFileName defines the filename i.e. "DataJob.GSI" and sFilePath defines the file-path, i.e. "A:\GSI".

Parameters
iPathInfo in Device info (Only GSI_EXTERNAL is valid)
sFileName in Valid Filename (8+3 format)
sFilePath in File-path

Return-Codes
RC_OK Successful termination.

See Also
GSI_GetDataPath

Example
The example defines the file “A:\GSI\DataJob.GSI” as new import file.
GSI_SetDataPath(GSI_EXTERNAL, "DataJob.GSI", "A:\\GSI")

6.4.24 GSI_GetDataPath

Description
Get the name of the file with the import data.

Declaration
GSI_GetDataPath(
   iPathInfo AS Integer,
   sFileName AS FileName,
   sFilePath AS FilePath )

Remarks
This procedure fetches the name and the path of the file from which data will be imported. If iPathInfo = GSI_EXTERNAL, then sFileName defines the filename i.e. "DataJob.GSI" and sFilePath defines the file-path, i.e. "A:\\GSI".

Parameters
   iPathInfo     out   Device info
   sFileName     out   Filename (8+3 format)
   sFilePath     out   File-path

Return-Codes
   RC_OK         Successful termination.

See Also
GSI_SetDataPath

Example
The example fetches the name and the path of the standard import data file:
   DIM iPathInfo AS Integer
   DIM sFileName AS FileName
   DIM sFilePath AS FilePath
   GSI_GetDataPath(iPathInfo, sFileName, sFilePath)

6.4.25 GSI_GetWiEntry

Description
Get data from the Theodolite data pool.
6. System Functions

### GSI_GetWiEntry

**Declaration**

```
GSI_GetWiEntry(  
    WiIdentification AS Integer,  
    WiEntry AS GSI_WiDlg_Entry_Type  
)
```

**Remarks**

This routine is used to fetch data from the Theodolite data pool. All existing wi’s can be fetched (see the description of the WI constants for possible values).

**Parameters**

- **WiIdentification in**
  - The identification of the WI.
- **WiEntry out**
  - The WI entry data. See the description of `GSI_WiDlg_Entry_Type` for further information.

**See Also**

- `GSI_SetWiEntry`

**Example**

See example `GSI_SetWiEntry`.

---

### GSI_SetWiEntry

**Description**

Put data to the Theodolite data pool.

**Declaration**

```
GSI_SetWiEntry(  
    WiIdentification AS Integer,  
    WiEntry AS GSI_WiDlg_Entry_Type  
)
```

**Remarks**

This routine is used to put data to the Theodolite data pool. See the description of the WI constants.

**Parameters**

- **WiIdentification in**
  - The identification of the WI.
- **WiEntry in**
  - The WI entry data. See the description of `GSI_WiDlg_Entry_Type` for further information.

**See Also**

- `GSI_GetWiEntry`

**Example**

`GSI_SetWiEntry` does not set WI.iId according to the first parameter, instead it will just use the value stored in WI.iId. If that value is unequal to the first parameter value, then it comes to a conflict. Use a `GSI_GetWiEntry()` first, to be sure that all values...
of the GSI_WiDlg_Entry_Type are initialized correctly. See also the example for the definition of a measurement dialog.

Save way:

```basic
GSI_GetWiEntry ( GSI_ID_HR, Wi )
Wi.lValid = TRUE
Wi.dValue = 2.12
GSI_SetWiEntry ( GSI_ID_HR, Wi )
```

### 6.4.27 GSI_GetRecMask

**Description**

Get the definition and the format of a recording mask.

**Declaration**

```basic
GSI_GetRecMask(  
    BYVAL iMaskNr AS Integer,  
    sMaskName AS String18,  
    RecWiMask AS GSI_Rec_Id_List,  
    iRecFormat AS Integer,  
    lEditMask AS Logical )
```

**Remarks**

This routine fetches the definition and the format of the recording mask with the number `iMaskNr`. Valid formats are `GSI_RECFORMAT_GSI` and `GSI_RECFORMAT_GSI16`. A recording mask can be set with `GSI_SetRecMask`. If `lEditMask` is TRUE the elements of the recording mask can be changed in `GSI_DefineRecMaskDlg`. All unused elements of the recording list are set to `GSI_ID_NONE`. All values from 0 to 5 are valid for the mask number. Mask number 0 is predefined for the station recording mask.

**Note**

Only the first 16 characters of `sMaskName` are valid.

**Parameters**

- `iMaskNr in` Number of the recording mask.
- `GSI_ACTUAL_RECMASK` can be used to retrieve settings of the actual mask
- `sMaskName out` Name of the recording mask
- `RecWiMask out` The definition of the recording mask. The elements of the array are the identification numbers of the WI’s. See the description
of the WI constants.

iRec out Recording format
Format out (GSI_RECFORMAT_GSI, GSI_RECFORMAT_GSI16)
lEditMask out Mask editable flag

See Also GSI_SetRecMask, GSI_DefineRecMaskDlg

Example The example uses the GSI_GetRecMask routine to fetch the definition and the format of the recording mask number 2.

```
DIM sMaskName AS String18
DIM RecWiMask AS GSI_Rec_Id_List
DIM iRecFormat AS Integer
DIM lEditMask AS Logical

GSI_GetRecMask(2, sMaskName, RecWiMask, iRecFormat, lEditMask)
```

6.4.28 GSI_SetRecMask

Description Set the definition and the format of a recording mask.

Declaration GSI_SetRecMask(
    BYVAL iMaskNr AS Integer,
    BYVAL sMaskName AS String18,
    BYVAL RecWiMask AS GSI_Rec_Id_List,
    BYVAL iRecFormat AS Integer,
    BYVAL lEditMask AS Logical)

Remarks This routine sets the definition and the format of the recording mask with the number iMaskNr. Valid formats are GSI_RECFORMAT_GSI and GSI_RECFORMAT_GSI16. If lEditMask is TRUE the elements of the recording mask can be changed in GSI_DefineRecMaskDlg. All unused elements should be set to GSI_ID_NONE. All values from 0 to 5 are valid for the mask number. Mask number 0 is predefined for the station recording mask.
Parameters

iMaskNr in Number of the recording mask.

GSI_ACTUAL_RECMASK can be used to set the values of the currently active mask.

sMaskName in Name of the recording mask.

RecWiMask in The definition of the recording mask. The elements of the array are the identification numbers of the WI’s. See the description of the WI constants.

iRec in Recording format

Format (GSI_RECFORMAT_GSI, GSI_RECFORMAT_GSI16)

lEditMask in Mask editable flag

See Also GSI_GetRecMask, GSI_DefineRecMaskDlg
Example

The example sets the 4th element of the currently active recording mask on GSI_ID_HZ.

```
DIM sMaskName AS String18
DIM RecWiMask AS GSI_Rec_Id_List
DIM iRecFormat AS Integer
DIM lEditMask AS Logical

GSI_GetRecMask(GSI_ACTUAL_RECMASK, sMaskName,
               RecWiMask, iRecFormat, lEditMask)
RecWiMask(4) = GSI_ID_HZ
GSI_SetRecMask(GSI_ACTUAL_RECMASK, sMaskName,
               RecWiMask, iRecFormat, lEditMask)
```

6.4.29 GSI_SetRecMaskNr

<table>
<thead>
<tr>
<th>Description</th>
<th>Set the used recording mask.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>GSI_SetRecMaskNr(BYVAL iMaskNr AS Integer)</td>
</tr>
<tr>
<td>Parameters</td>
<td>iMaskNr in Number of the recording mask. Number must be in the range 1.. GSI_MAX_REC_MASKS.</td>
</tr>
<tr>
<td>See Also</td>
<td>GSI_GetRecMaskNr</td>
</tr>
</tbody>
</table>
Example  The example sets the next recording mask.

```basic
DIM i AS Integer

GSI_GetRecMaskNr( i )
i = i + 1 ' take next mask
i = ((i - 1) MOD GSI_MAX_REC_MASKS) + 1
GSI_SetRecMaskNr( i )
```

### 6.4.30 GSI_GetRecMaskNr

**Description**  Returns the used recording mask.

**Declaration**  

```basic
GSI_GetRecMaskNr(iMaskNr AS Integer)
```

**Parameters**  

- `iMaskNr`  
  Number of the recording mask.

**See Also**  

GSI_SetRecMaskNr

### 6.4.31 GSI_DefineRecMaskDlg

**Description**  Defines the recording mask dialog.

**Declaration**  

```basic
GSI_DefineRecMaskDlg()
```

**Remarks**  

Defines the contents of the recording mask. Using a dialog with list-fields, the user can select the items for the user registration mask. This routine is an interactive equivalent to the routines GSI_GetRecMask and GSI_SetRecMask.

**See Also**  

GSI_GetRecMask, GSI_SetRecMask,

**Example**  

```basic
GSI_DefineRecMaskDlg ()
```

### 6.4.32 GSI_ManCoordDlg

**Description**  Show the manual co-ordinate input dialog.
**Declaration**

```basic
GSI_ManCoordDlg(
  BYVAL sCaption AS _Token,
  BYVAL iPointType AS Integer,
  BYVAL iFlags AS Integer,
  BYVAL sHelpText AS _Token
)
```

**Remarks**

This routine shows the manual co-ordinates input dialog and allows editing, coding and recording. The type of co-ordinates (station or target) can be selected using `iPointType`. Recording to the current data-file (defined in `GSI_ImportCoordDlg`) with `REC` or leaving this function with `CONT` is only possible if the point number is valid, and at least E- and N-co-ordinates are valid. If `GSI_HEIGHT_MUST` is included in `iFlags` the Height / Elevation-co-ordinate must be valid too. Leaving using ESC or QUIT (Shift-F6) is always possible. Recording and coding sets the according values in the Theodolite data-pool too.

**Parameters**

- `sCaption` in
  - The maximal five-character long left part of the title bar.
- `iPointType` in
  - station or target point. For the values for `PointType` see table below
  - **Point Type** | **Meaning**
    |-------------------|
    | GSI_STATION       | station point number |
    | GSI_INDIV_TG      | individual target number |
    | GSI_RUN_TG        | running target |
    | GSI_BACKSIGHT     | backside number (analog target, only changed prompts) |
GSI_POINT_CODE

PointId / CodeId
(analog target, only changed prompts)

Point in only point number, co-ordinates will be set to 0
Point out point number and co-ordinates. For further information see the description of GSI_Point_Coord_Type
iFlags in defines functionality

<table>
<thead>
<tr>
<th>Valid Flags</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ALLOW_REC</td>
<td>allows recording and coding</td>
</tr>
<tr>
<td>GSI_HEIGHT_MUST</td>
<td>height must be entered</td>
</tr>
<tr>
<td>GSI_NE_OPTIONAL</td>
<td>only height must be entered, north &amp; east are optional</td>
</tr>
<tr>
<td>GSI_MULTI_REC</td>
<td>Allows entering and recording of more than one data-set, without leaving this routine</td>
</tr>
<tr>
<td>GSI_NO_FILE_CHANGE</td>
<td>File changing is disabled</td>
</tr>
</tbody>
</table>

Flags can be combined with '+'-operator (iFlags = iFlag1 + iFlag2)

sHelpText in This text is shown, when the help button SHIFT-F1 is pressed and the help functionality of the theodolite is enabled.

See Also GSI_ImportCoordDlg

Example
```
DIM Point AS GSI_Point_Coord_Type

GSI_ManCoordDlg ( "TEST", GSI_STATION, Point, 
GSI_HEIGHT_MUST+GSI_ALLOW_REC, 
"This is the Helptext" )
```

### 6.4.33  GSI_ImportCoordDlg

**Description**  
Show the co-ordinate import dialog.

**Declaration**  
```
GSI_ImportCoordDlg(
    BYVAL sCaption AS _Token, 
    BYVAL iPointType AS Integer, 
    Point AS GSI_Point_Coord_Type, 
    BYVAL iFlags AS Integer, 
    BYVAL iImportFile AS Integer, 
    BYVAL sImportHelp AS _Token, 
    BYVAL sInputHelp AS _Token, 
    BYVAL sF2Button AS _Token, 
    BYVAL sF4Button AS _Token
)
```

**Remarks**  
This routine contains three dialogues, the search-, the view- and the manual-input dialog. The type of co-ordinates (station or target) can be selected using `iPointType`. The search dialog allows selecting the data- or the measure file and editing a point-number. Depending on the pressed button, the manual co-ordinate input function (only if `GSI_ALLOW_MAN` is included in `iFlags`, see `GSI_ManCoordDlg`) or the view-co-ordinates dialog will be called.

The start of searching is always at the top of the file. With the two search keys, the user can step from one valid point to the next in both directions.

**Rules for a valid point:**
- point number found
- E- and N-coordinates (target or station) exists and are valid
- if `GSI_HEIGHT_MUST` is included in `iFlags`, a valid
height / elevation-coordinate must exist to within the file too.

If no valid point exists or no more valid points are in the desired search direction, a warning message will be displayed.

Parameters

- **sCaption in**: The maximal five-character long left part of the title bar.
- **iPointType in**: station or target point. For the values for `PointType` see table below
  - **Point Type** | **Meaning**
    |----------------|
    | GSI_STATION    | station point number |
    | GSI_INDIV_TG   | individual target number |
    | GSI_RUN_TG     | running target |
    | GSI_BACKSIGHT  | backside number (analog target, only changed prompts) |
    | GSI_POINT_CODE | PointId / CodeId (analog target, only changed prompts) |
- **Point in**: Only point number, the co-ordinates will be set to 0.
- **Point out**: point number and -co-ordinates. For further information see the description of `GSI_Point_Coord_Type`.
- **iFlags in**: defines functionality
  - **Valid Flags** | **Meaning**
    |----------------|
    | GSI_ALLOW_REC  | allows recording and coding |
    | GSI_MULTI_REC  | Allows multiple manual coord. entering |
GeoBASIC Reference Manual

6. System Functions

GSI_ALLOW_MAN allows manual coord. entering
GSI_HEIGHT_MUST height must be entered
GSI_DIRECT_SEARCH direct searching without dialog
GSI_NO_VIEW no coord view if found
GSI_NE_OPTIONAL only height must be entered, north & east are optional
GSI_SEARCH_FROM_END Starts searching from end of file
GSI_NO_FILE_CHANGE Changing of file is disabled
GSI_GET_NEXT Return the next valid data-set, ignore sPtNr

Flags can be combined with ‘+’-operator (iFlags = iFlag1 + iFlag2)
iImportFile in defines the source file for importing

Valid Import File Meaning
GSI_FILE_MEAS MEAS file
GSI_FILE_DATA DATA file
GSI_FILE_LAST last used file

sImportHelp in Help text for import dialog. Only visible if the help functionality of the theodolite is enabled.
sInputHelp in Help text for manual input dialog. Only visible if the help functionality of the theodolite is enabled.
sF2Button in Text for activating F2 button.
sF4Button in Text for activating F4 button

See Also GSI_ManCoordDlg

TPS1100-Version 2.10 6-167
Example

```basic
DIM Point AS GSI_Point_Coord_Type
GSI_ImportCoordDlg( "IMP", GSI_INDIV_TG, Point, GSI_ALLOW_REC + GSI.Allow_MAN, GSI_FILE_DATA, "Import Help Text", "Input Help Text", "F2", "F4" )
```

6.4.34 GSI_SetLineSysMDlg

**Description**  
Sets a line in the system measurement dialog.

**Declaration**  
```basic
GSI_SetLineSysMDlg(
    BYVAL iDlgNr AS Integer
    BYVAL iLineNr AS Integer
    BYVAL iSysParamId AS Integer
)
```

**Remarks**  
This routine sets one line in the system measurement dialog. To fetch information about a line, GSI_GetLineSysMDlg can be used. Unused lines should be set to GSI_PAR_NONE.

**Note**  
1) Parameters are identified by GSI_PAR_* values and not by GSI_ID_* values.
2) A line in the system measurement dialog can only be set to a system parameter not to an application parameter.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iDlgNr</td>
<td>Integer</td>
<td>The number of the system measurement dialog where the line should be set. Possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GSI_SYS_MDLG_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GSI_SYS_MDLG_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GSI_SYS_MDLG_3</td>
</tr>
</tbody>
</table>

| iLineNr       | Integer| The number of the line to set. Valid numbers: 1.. GSI_MAX_DLG_LINES |

| iSysParamId   | Integer| Identification of the system parameter. Refer to the chapter |
“Constants for Measurement Dialog Definition”

See Also
GSI_GetLineSysMDlg
GSI_DefineMDlg

Example
See sample program “meas.gbs”.
This example uses GSI_SetLineSysMDlg to configure the first two lines of the first system measurement dialog.

GSI_SetLineSysMDlg( GSI_SYS_MDLG_1, 1, GSI_PAR_Date )
GSI_SetLineSysMDlg( GSI_SYS_MDLG_1, 2, GSI_PAR_Time )

6.4.35 GSI_GetLineSysMDlg

Description
Gets the definition of a line in the system measurement dialog.

Declaration
GSI_GetLineSysMDlg(
    BYVAL iDlgNr AS Integer
    BYVAL iLineNr AS Integer
    iSysParamId AS Integer )

Remarks
This routine fetches the information about the setting of one line in the system measurement dialog. To set a line in the system measurement dialog the routine GSI_SetLineSysMDlg can be used.
### Parameters

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iDlgNr</td>
<td>The number of the system measurement dialog where the line should be fetched. Possible values are:</td>
</tr>
<tr>
<td></td>
<td><strong>Value</strong></td>
</tr>
<tr>
<td></td>
<td>GSI_SYS_MDLG_1</td>
</tr>
<tr>
<td></td>
<td>GSI_SYS_MDLG_2</td>
</tr>
<tr>
<td></td>
<td>GSI_SYS_MDLG_3</td>
</tr>
<tr>
<td>iLineNr</td>
<td>The number of the line to fetch.</td>
</tr>
<tr>
<td>iSysParamId</td>
<td>Identification of the system parameter. Refer to the chapter “Constants for Measurement Dialog Definition”</td>
</tr>
</tbody>
</table>

### See Also

GSI_SetLineSysMDlg  
GSI_DefineMDlg

### Example

See sample program “meas.gbs”.  
This example uses GSI_GetLineSysMDlg to get information about the configuration of the first system measurement dialog’s first two lines.

```basic
DIM iParLine1 AS Integer  
DIM iParLine2 AS Integer  
GSI_GetLineSysMDlg( GSI_SYS_MDLG_1, 1, iParLine1)  
GSI_GetLineSysMDlg( GSI_SYS_MDLG_1, 2, iParLine2)
```

### 6.4.36 GSI_SetMDlgNr

**Description**  
Sets the number of the system measurement dialog.

**Declaration**  
GSI_SetMDlgNr( BYVAL iMDlgNr AS Integer)

**Remarks**  
Sets the number of the system measurement dialog. The content of these dialogs can be changed by using of DefineMDlg.

**Parameters**
6.4.37 GSI_GetMDlgNr

**Description**
Returns the number of the system measurement dialog.

**Declaration**
```
GSI_GetMDlgNr(iMDlgNr AS Integer)
```

**Remarks**
Returns the number of the system measurement dialog.

**Parameters**
- `iMDlgNr` out
  Number of the actual measurement dialog

**See Also**
GSI_SetMDlgNr

---

6.4.38 GSI_CreateMDlg

**Description**
Create and show the user definable measurement dialog.

**Declaration**
```
GSI_CreateMDlg(
  BYVAL iFixLines AS Integer
  BYVAL sCaptionLeft AS _Token
  BYVAL sCaptionRight AS _Token
  BYVAL sHelpText AS _Token
)
```

**Remarks**
This routine creates and shows the user definable measurement dialog with `iFixLines` fix lines, the left part of the title bar `sCaptionLeft`, the caption `sCaptionRight`, and the help text `sHelpText`.

---

Example
See sample program “meas_od.gbs”. This example sets the next dialog mask
```
GSI_GetMDlgNr( i )
i = (i + 1) MOD GSI_MAX_MDLG_MASKS
GSI_SetMDlgNr( i )
```
Only one measurement dialog can exist at the same time. If GSI_CreateMDlg is called and there already exists a measurement dialog, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

**Note**  
If a graphics dialog or a text dialog exist together with a measurement dialog, all button routines (MMI_AddButton, MMI_GetButton, MMI_DeleteButton) are related to the measurement dialog.

The shown parameters used in the dialog are defined in the user display mask (see GSI_DefineMDlg).

**Parameters**

- **iFixLines** in The number of fix lines. (These lines are not scrolled.)
- **sCaptionLeft** in The part of the title bar displayed on the left border (up to five characters wide)
- **sCaptionRight** in The caption of the dialog.
- **sHelpText** in This text is shown, when the help button SHIFT-F1 is pressed and the help functionality of the theodolite is enabled.

**See Also**

GSI_UpdateMDlg  
GSI_UpdateMeasurement

**Example**  
See example file "meas.gbs" too.

This example uses the measure dialog routines GSI_CreateMDlg, GSI_UpdateMDlg and GSI_UpdateMeasurement to execute a measure process.

```basic
DIM ValidForRec AS Logical
DIM RetCodeForMsg AS Integer
DIM WaitTime AS Integer
DIM iButton AS Integer

WaitTime = 10 ' ms

' user definition of measurement dialog
' can be placed here
```
GSI_CreateMDlg( 1, "WIR", "Measure Dialog", "This is the HelpText")

DO
  GSI_UpdateMeasurement( TMC_MEA_INC, 
    WaitTime, ValidForRec, 
    RetCodeForMsg, FALSE )
  GSI_UpdateMDlg (iButton)
LOOP UNTIL iButton = MMI_ESC_KEY
GSI_DeleteDialog()

### 6.4.39 GSI_SetLineMDlg

**Description**
Sets one line in the user definable measurement dialog to system parameter.

**Declaration**

```
GSI_SetLineMDlg( 
  BYVAL iLineNr AS Integer 
  BYVAL iSysParamId AS Integer )
```

**Remarks**
This routine sets the configuration of a line in the user definable measurement dialog to a system parameter. This measurement dialog is initialized automatically with the actual settings of the first system measurement dialog. Modifications of the user definable dialog have no effects on the system measurement dialog and will be lost after termination of the program. An unused line should be set to GSI_PAR_NONE. To add a user definable application parameter to the dialog use GSI_SetLineMDlgPar. To add a line of text (e.g. separator line) to the dialog use GSI_SetLineMDlgText.

**Parameters**

- **iLineNr**  
  *in* The number of the line to set.  
  Valid numbers:  
  1.. GSI_MAX_DLG_LINES

- **iSysParamId**  
  *in* Identification of the system parameter. Refer to the chapter "Constants for Measurement Dialog Definition"
See Also

GSI_SetLineMDlgPar
GSI_SetLineMDlgText
GSI_CreateMDlg

Example

This example uses GSI_SetLineMDlg to configure the user definable measurement dialog.

GSI_SetLineMDlg( 1, GSI_PAR_ReflHeight )
GSI_SetLineMDlg( 2, GSI_PAR_Info1 )
GSI_SetLineMDlg( 3, GSI_PAR_Info2 )
...
GSI_SetLineMDlg( 10, GSI_PAR_NONE )
GSI_SetLineMDlg( 11, GSI_PAR_NONE )
GSI_SetLineMDlg( 12, GSI_PAR_NONE )

6.4.40 GSI_SetLineMDlgText

Description

Puts a text line into the user definable measurement dialog.

Declaration

GSI_SetLineMDlgText(  
    BYVAL iLineNr AS Integer,  
    BYVAL iParamId AS Integer,  
    BYVAL sText AS _Token )

Remarks

This routine inserts a pure text line into the user definable measurement dialog. To add an user definable application parameter to the dialog use GSI_SetLineMDlgPar. To add a system parameter to the dialog use GSI_SetLineMDlg.

Parameters

iLineNr in The number of the line to set.
Valid numbers:
1.. GSI_MAX_DLG_LINES
iParamId in Id of the system parameter.
sText in Contents of the line.

See Also

GSI_SetLineMDlg
GSI_SetLineMDlgPar
GSI_CreateMDlg
Example

This example uses GSI_SetLineMDlg and GSI_SetLineMDlgText to configure the user definable measurement dialog.

```
GSI_SetLineMDlg( 1, GSI_PAR_Date )
GSI_SetLineMDlg( 2, GSI_PAR_Time )
GSI_SetLineMDlgText( 3, GSI_PAR_APPDATA0,
    "-----------------" )
GSI_SetLineMDlg( 4, GSI_PAR_Info1 )
GSI_SetLineMDlg( 5, GSI_PAR_Info2 )
```

6.4.41  GSI_SetLineMDlgPar

Description

Sets one line in the user definable measurement dialog to an application parameter.

Declaration

```
GSI_SetLineMDlgPar(  
    BYVAL iLineNr AS Integer
    BYVAL iApplParamId AS Integer
    BYVAL sLabel AS _Token
    BYVAL lEditable AS Logical
    BYVAL iFormat AS Integer )
```

Remarks

This routine sets the configuration of a line in the user definable measurement dialog to an application parameter. The style of the application parameter is also defined in this routine. Any floating point format and strings are valid formats. The starting values of every application parameter is not predefined and hence has to be set explicitly. To initialize an application parameter the routine GSI_SetWiEntry can be used. To add a line of text to the dialog use GSI_SetLineMDlgText. To add a system parameter to the dialog use GSI_SetLineMDlg.

Parameters

- **iLineNr**  in  The number of the line to set.  
  Valid numbers:  
  1.. GSI_MAX_DLG_LINES
- **iApplParamId** in  Id of the application parameter.
- **sLabel** in  Description of parameter on display.
lEditAble in Edit ability of the value in the measurement dialog.

iFormat in Format descriptor of the application parameter. The format defines if a dimension field is available. Following values can be used:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_FORMAT_STRING</td>
<td>string</td>
</tr>
<tr>
<td>MMI_FORMAT_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>MMI_FORMAT_DISTANCE</td>
<td>distance</td>
</tr>
<tr>
<td>MMI_FORMAT_SUBDISTANCE</td>
<td>[mm]</td>
</tr>
<tr>
<td>MMI_FORMAT_ANGLE</td>
<td>angle</td>
</tr>
<tr>
<td>MMI_FORMAT_VANGLE</td>
<td>vertical angle</td>
</tr>
<tr>
<td>MMI_FORMAT_HZANGLE</td>
<td>horizontal angle</td>
</tr>
<tr>
<td>MMI_FORMAT_TEMPERATURE</td>
<td>temperature</td>
</tr>
</tbody>
</table>

See Also  
GSI_SetLineMDlg
GSI_SetLineMDlgText
GSI_CreateMDlg

Example See also sample file “meas.gbs”. This example uses GSI_SetLineMDlgPar and GSI_SetWiEntry to configure the user definable measurement dialog.
DIM WI AS GSI_WIDLG_ENTRY_TYPE
WI.lValid = FALSE
WI.iDataType = GSI_ASCII
GSI_SetWiEntry(GSI_ID_APPDATA0, WI)
GSI_SetLineMDlgPar(1, GSI_PAR_AppData0, "Stat. Name:", TRUE, MMI_FFORMAT_STRING)
WI.lValid = TRUE
WI.iDataType = GSI_DOUBLE
WI.dValue = 2.2
GSI_SetWiEntry(GSI_ID_APPDATA3, WI)
GSI_SetLineMDlgPar(8, GSI_PAR_AppData3, "Distance : ", TRUE, MMI_FFORMAT_DISTANCE)

6.4.42 GSI_UpdateMDlg

| Description | Updates the user definable measurement dialog. |
| Declaration | GSI_UpdateMDlg( iButton As Integer) |
| Remarks | This procedure updates the user definable measurement dialog with the actual values from the Theodolite data pool and returns pressed buttons. |

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>iButton</td>
<td>out</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>See Also</th>
<th>GSI_CreateMDlg</th>
</tr>
</thead>
<tbody>
<tr>
<td>---------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>GSI_UpdateMeasurement</td>
</tr>
</tbody>
</table>

| Example | See example GSI_CreateMDlg and example file „meas.gbs“. |
### GSI_DefineMDlg

**Description**
Defines the entries of the user definable measurement dialog.

**Declaration**

\[
\text{GSI\_DefineMDlg} ( \text{BYVAL sCaption AS _Token} )
\]

**Remarks**
Interactively defines the contents of the user definable measurement dialog. Using a dialog with list fields, the user can select the items for the measurement dialog. This routine is an interactive equivalent to the routines GSI\_SetLineSysMDlg and GSI\_GetLineSysMDlg.

**Parameters**

- **sCaption in**
  The left caption of the title bar. (Up to 5 characters wide.)

**See Also**
GSI\_GetDlgMask
GSI\_SetDlgMask

**Example**

\[
\text{GSI\_DefineMDlg( "DEF" )}
\]

### GSI_UpdateMeasurment

**Description**
Update the measurement data.

**Declaration**

\[
\text{GSI\_UpdateMeasurment(}
\begin{align*}
\text{iInclinePrg} & \text{ AS Integer}, \\
\text{iWaitTime} & \text{ AS Integer}, \\
\text{lValidForRec} & \text{ AS Logical}, \\
\text{iRetCodeForMsg} & \text{ AS Integer}, \\
\text{lChkIncRangeNow} & \text{ AS Logical}
\end{align*}
\)
\]

**Remarks**
This function updates the measurement values in the Theodolite data pool. The data are the incline program, angles, distances, time, reflector height.
Parameters

iInclinePrg  in  The manner of incline compensation. Following settings are possible:

<table>
<thead>
<tr>
<th>Incline Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_MEA_INC</td>
<td>get inclination</td>
</tr>
<tr>
<td>TMC_AUTO_INC</td>
<td>get inclination with automatism</td>
</tr>
<tr>
<td>TMC_PLANE_INC</td>
<td>get inclination always with plane</td>
</tr>
</tbody>
</table>

iWaitTime  in  The wait time for a result (in ms). This time is used for synchronising the TMC task.

lValidForRec  out  Indicates validity of the registration

iRetCodeForMsg  out  Return code of the measurement

lChkIncRange  in  Now TRUE: check incline range immediate

See Also

GSI_CreateMDlg
GSI_UpdateMDlg
GSI_DeleteDialog

Example

See example GSI_CreateMDlg and example file „meas.gbs“.

6.4.45  GSI_Measure

Description  Measure and registration dialog.

Declaration  GSI_Measure ( )

Remarks  This procedure opens the measure and registration dialog.

Parameters

none
### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Success</td>
</tr>
</tbody>
</table>

### Example

Do a measure and registration dialog.

```
GSI_Measure ()
```

---

#### 6.4.46 GSI_ExecuteAutoDist

**Description**

Executes an automatic distance measurement.

**Declaration**

```
GSI_ExecuteAutoDist ()
```

**Remarks**

This procedure starts a distance measurement on condition that “Auto Dist” is enabled and one of the distance measurement-program buttons (FNC-menu) was pressed.

**Parameters**

none

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Example**

See example file „meas.gbs“ or „meas_od.gbs“.  

---

#### 6.4.47 GSI_CheckTracking

**Description**

Returns if distance tracking is running.

**Declaration**

```
GSI_CheckTracking(lTracking As Logical)
```

**Remarks**

This returns if a distance tracking is running.

An automatic start of distance tracking can be started on several conditions, i.e. by Quick-Coding, GSI_ExecuteAutoDist or by pressing buttons in the FNC-menu.

Tracking can be terminated by the instrument itself due several reasons, i.e. for laser security reasons (US-configuration).
6.4.48 GSI_RecordRecMask

Description
Recording the given wi mask.

Declaration
GSI_RecordRecMask (RecList AS GSI_REC_ID_LIST, BYVAL eProgFunction AS Logical, BYVAL bCheckStdMask AS Logical, BYVAL bIncAndSetRunPt AS Logical)

Remarks
This procedure records the given wi list. The target can be the memory card or the interface. The parameter for the interface depends on the GSI communication settings. Errors will shown on the display, when recording list will be stored in the memory card. Otherwise the error messages will be given on the interface.

Parameters
- RecList in recording list
- eProgFunction in program flag in the wi's (TRUE = ON, FALSE = OFF)
- bCheckStdMask in testing the standard recording mask
- bIncAndSetRunPt in increment the point number

Return Codes
- RC_OK Success
- RC_IVRESULT registration failure

See Also
Example  Record RecList.
     DIM RecList AS GSI_REC_ID_LIST

     ' initialize RecList with adequate values
     GSI_RecordRecMask ( RecList, TRUE, TRUE, TRUE )
### 6.5 CENTRAL SERVICE FUNCTIONS CSV

#### 6.5.1 Summarizing Lists of CSV Types and Procedures

#### 6.5.1.1 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPS_Fam_Type</td>
<td>Information about the current hardware.</td>
</tr>
<tr>
<td>Date_Time_Type</td>
<td>Date and time information.</td>
</tr>
<tr>
<td>Date_Type</td>
<td>Date information.</td>
</tr>
<tr>
<td>Time_Type</td>
<td>Time information.</td>
</tr>
</tbody>
</table>

#### 6.5.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_ChangeFace</td>
<td>Do an absolute positioning to the opposite.</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>Returns if an alternative user-task was running.</td>
</tr>
<tr>
<td>CSV_Delay</td>
<td>Delay routine.</td>
</tr>
<tr>
<td>CSV_GetATRStatus</td>
<td>Gets the current ATR state.</td>
</tr>
<tr>
<td>CSV_GetDateTime</td>
<td>Get the date and the time of the system.</td>
</tr>
<tr>
<td>CSV_GetElapseSysTime</td>
<td>Returns the difference between a reference time and the system time.</td>
</tr>
<tr>
<td>CSV_GetGBIVersion</td>
<td>Returns the release number of the GeoBASIC interpreter.</td>
</tr>
<tr>
<td>CSV_GetInstrumentFamily</td>
<td>Get information about the system.</td>
</tr>
<tr>
<td>CSV_GetInstrumentName</td>
<td>Get the LEICA specific instrument name.</td>
</tr>
<tr>
<td>CSV_GetInstrumentNo</td>
<td>Get the instrument number.</td>
</tr>
<tr>
<td>CSV_GetLaserPlummet</td>
<td>Returns the laser plummet state.</td>
</tr>
<tr>
<td>CSV_GetLockStatus</td>
<td>Gets the current state of the locking facility.</td>
</tr>
<tr>
<td>CSV_GetLRStatus</td>
<td>Returns the status of the system.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSV_GetPrismType</td>
<td>Returns the used prism</td>
</tr>
<tr>
<td>CSV_GetSWVersion</td>
<td>Get the version of the system software.</td>
</tr>
<tr>
<td>CSV_GetSysTime</td>
<td>Returns the system time.</td>
</tr>
<tr>
<td>CSV_GetTargetType</td>
<td>Get the target type for distance measurements.</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>Returns the internal temperature of the instrument.</td>
</tr>
<tr>
<td>CSV_Laserpointer</td>
<td>Switch on / off the laser pointer.</td>
</tr>
<tr>
<td>CSV_LibCall</td>
<td>Call a GeoBASIC routine from another program.</td>
</tr>
<tr>
<td>CSV_LibCallAvailable</td>
<td>Check if GeoBASIC routine from another program is available.</td>
</tr>
<tr>
<td>CSV_LockIn</td>
<td>Starts locking (ATR).</td>
</tr>
<tr>
<td>CSV_LockOut</td>
<td>Stops locking (ATR).</td>
</tr>
<tr>
<td>CSV_MakePositioning</td>
<td>Do an absolute positioning.</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>Resets the “alternative user-task was running” flag.</td>
</tr>
<tr>
<td>CSV_SetATRStatus</td>
<td>Sets the current state of Automatic Target Recognition.</td>
</tr>
<tr>
<td>CSV_SetLaserPlummet</td>
<td>Switches the laser plummet</td>
</tr>
<tr>
<td>CSV_SetLightGuide</td>
<td>Switch on / off the light guide.</td>
</tr>
<tr>
<td>CSV_SetLockStatus</td>
<td>Sets the current state of the locking facility.</td>
</tr>
<tr>
<td>CSV_SetPrismType</td>
<td>Sets the used prism</td>
</tr>
<tr>
<td>CSV_SetTargetType</td>
<td>Set the target type for distance measurements.</td>
</tr>
<tr>
<td>CSV_SysCall</td>
<td>Call a system function.</td>
</tr>
<tr>
<td>CSV_SysCallAvailable</td>
<td>Check if system function is available.</td>
</tr>
</tbody>
</table>
6.5.2 Data Structures for the Central Service Functions

6.5.2.1 Date_Time_Type: Date and Time

**Description** These data structures are used to store date and time information.

```plaintext
TYPE Date_Type
  iYear AS Integer year as a 4 digit number
  iMonth AS Integer month as a 2 digit number
  iDay AS Integer day as a 2 digit number
END Date_Type

TYPE Time_Type
  iHour AS Integer hour as a 2 digit number (24 hours format)
  iMinute AS Integer minutes as a 2 digit number
  iSecond AS Integer seconds as a 2 digit number
END Time_Type

Date_Time_Type
  Date AS Date_Type date (as defined above)
  Time AS Time_Type time (as defined above)
END Time_Type
```
6.5.2.2 TPS_Fam_Type: Information about the system

**Description**  
This data structure is used to store information about the hardware. Further information about the hardware can be obtained by your local Leica representative.

```
TYPE TPS_Fam_Type  
iClass AS Integer  
  The class of the system. Values:
  Id  Meaning
  TPS1101  TPS1100 accuracy 1"
  TPS1102  TPS1100 accuracy 2"
  TPS1103  TPS1100 accuracy 3"
  TPS1105  TPS1100 accuracy 5"

iEDMBuiltIn AS Logical  
iEDMTypeII AS Logical  
iEDMTypeIII AS Logical  
iEDMReflectorless AS Logical  
iMotorized AS Logical  
iATR AS Logical  
iEGL AS Logical  
iLaserPlummet AS Logical  
iAutoCollimation AS Logical  
iSimulator AS Logical  
  EDM built-in
  EDM built-in, type II
  EDM built-in, type III
  Red Laser
  Motorised
  Automatic Target Recognition (ATR)
  EGL Guide Light
  Laser Plummet
  Auto-collimation lamp
  Hardware is simulator on Windows-PC

END TPS_Fam_Type
```
6.5.3 CSV_GetDateTime

Description  Get the date and the time of the system.

Declaration  CSV_GetDateTime(
                DateAndTime AS Date_Time_Type )

Remarks  The CSV_GetDateTime routine reads the date and the time
           from the system's real-time clock (RTC) and returns the values in
           the structure Date_Time_Type. In the case of TPS_Sim the
           system clock will be read.

Parameters  
            DateAndTime  out  The structure for the date and the time.

Return Codes  
            RC_UNDEFINED  The date and time is not set (not
                        yet/not any longer).

Example  The example uses the CSV_GetDateTime routine to get the
           date and the time of the system and displays the values.

            DIM DT AS Date_Time_Type
            ON ERROR RESUME
            CSV_GetDateTime( DT )

            IF ERR = RC_OK THEN
                MMI_PrintInt( 0, 0, 5, DT.Date.iYear, TRUE )
                MMI_PrintInt( 6, 0, 3, DT.Date.iMonth, TRUE )
                MMI_PrintInt( 10, 0, 3, DT.Date.iDay, TRUE )
                MMI_PrintInt( 0, 1, 3, DT.Time.iHour, TRUE )
                MMI_PrintInt( 4, 1, 3, DT.Time.iMinute, TRUE )
                MMI_PrintInt( 8, 1, 3, DT.Time.iSecond, TRUE )
            ELSEIF ERR = RC_UNDEFINED THEN
                MMI_PrintStr( 0, 0,
                              "Date and time not set.", TRUE )
            ELSE
                MMI_PrintStr( 0, 0,
                              "Unexpected error code.", TRUE )
            END IF
6.5.4 CSV_GetTemperature

**Description**
Returns the internal temperature of the instrument.

**Declaration**
```basic
CSV_GetTemperature( IntTemp AS Temperature )
```

**Remarks**
This routine returns the internal temperature.

**Parameters**
- `IntTemp` out: Internal temperature

6.5.5 CSV_GetInstrumentName

**Description**
Get the LEICA specific instrument name.

**Declaration**
```basic
CSV_GetInstrumentName( sName AS String30 )
```

**Remarks**
The `CSV_GetInstrumentName` routine returns the name of the system in the string `sName`.

**Parameters**
- `sName` out: The LEICA specific instrument name.

**Return Codes**
- none

**See Also**
- CSV_GetInstrumentNo,
- CSV_GetInstrumentFamily

**Example**
The example uses the `CSV_GetInstrumentName` routine to get the instrument name and displays it.

```basic
DIM sName AS String30
CSV_GetInstrumentName ( sName )
MMI_PrintStr ( 0, 0, sName, TRUE )
```
6.5.6 CSV_GetInstrumentNo

**Description**  
Get the instrument number.

**Declaration**  
CSV_GetInstrumentNo( iSerialNo AS Integer )

**Remarks**  
The CSV_GetInstrumentNo routine returns the serial number of the system.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iSerialNo</td>
<td>out The serial number of the system.</td>
</tr>
</tbody>
</table>

**Return Codes**

none

**See Also**
CSV_GetInstrumentName, CSV_GetInstrumentFamily

**Example**
The example uses the CSV_GetInstrumentNo routine to get the instrument number and displays it.

```geo
DIM iSerialNo AS Integer
CSV_GetInstrumentNo( iSerialNo )
MMI_PrintInt( 0, 1, 20, iSerialNo, TRUE )
```

6.5.7 CSV_GetInstrumentFamily

**Description**  
Get information about the system.

**Declaration**  
CSV_GetInstrumentFamily(
    Family AS TPS_Fam_Type )

**Remarks**  
The CSV_GetInstrumentFamily routine returns the class and the instrument type of the system (see description of the data structure TPS_Fam for return values).

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family</td>
<td>out Contains the class and instrument type data. See description of the data structure TPS_Fam for return values.</td>
</tr>
</tbody>
</table>

**TPS_Sim**  
Always sets Family.lSimulator to TRUE.
6.5.8 CSV_GetSWVersion

**Description**
Get the version of the system software.

**Declaration**

```
CSV_GetSWVersion( iRelease AS Integer,
                 iVersion AS Integer )
```

**Remarks**
The CSV_GetSWVersion routine returns the Release number and the number of the system software version. These numbers can be interpreted together as software identification (Release.Version, e.g. 1.05).

**Parameters**

- `iRelease` out value of the Release number can be in the range from 0 to 99
- `iVersion` out value of the version number can be in the range from 0 to 99

**See Also**
Example

The example uses the CSV_GetSWVersion routine to get the system software version and displays it.

```
DIM iRelease AS Integer
DIM iVersion AS Integer

CSV_GetSWVersion( iRelease, iVersion )
MMI_PrintVal( 0, 0, 6, 2,
               iRelease + iVersion / 100, TRUE )
```

### 6.5.9 CSV_GetGBIVersion

**Description**

Returns the release number of the GeoBASIC interpreter.

**Declaration**

```
CSV_GetGBIVersion( 
   iRelease as Integer, 
   iVersion as Integer, 
   iSubVersion as Integer )
```

**Remarks**

This function returns the release version of the running GeoBASIC interpreter.

**Parameters**

- `iRelease` out  Release number
- `iVersion` Out  Version Number
- `iSubVersion` out  Subversion number

**Return-Codes**

- `RC_OK`  Successful termination.
Example  This example shows the currently used GeoBASIC interpreter release number.
DIM iRel As Integer
DIM iVer As Integer
DIM iSubVer As Integer

MMI_CreateTextDialog(
   6, "-CSV-", "Test CSV", "no help available")
CSV_GetGBIVersion (iRel, iVer, iSubVer)
MMI_PrintStr(0, 0,
   "GBI: "+Str$(iRel) + "." +
   Str$(iVer) + "."+Str$(iSubVer), TRUE)
MMI_DeleteDialog()

6.5.10 CSV_GetElapseSysTime

Description Returns the difference between a reference time and the system time.

Declaration CSV_GetElapseSysTime( iRefTime AS Integer,
   iElapse AS Integer )

TPS_Sim Use PC time base. Time resolution is one second.

Remarks The routine CSV_GetElapseSysTime returns the difference of between a given reference time iRefTime and the systems time. Whenever the system starts up, the system time is reset.

Parameters
   iRefTime  in  The reference time.
iElapse  out  The difference between iRefTime and the system time. The difference is returned in [ms].

See Also CSV_GetSysTime,
   CSV_GetDateTime
Example

The example uses the routine CSV_GetElapseSysTime to get a time difference.

```basic
DIM iElapse AS Integer
DIM iRefTime AS Integer

CSV_GetSysTime(iRefTime)' returns reference time
' do something...
CSV_GetElapseSysTime( iRefTime, iElapse )
MMI_PrintInt( 0, 0, 20, iElapse, TRUE )
```

### 6.5.11 CSV_GetSysTime

**Description**
Returns the system time.

**Declaration**

```basic
CSV_GetSysTime( iTime AS Integer )
```

**Remarks**
The routine returns the systems time. Whenever the system starts up, the system time is reset.

**Parameters**

- `iTime` out: The system time in ms.

**See Also**
CSV_GetElapseSysTime,
CSV_GetDateTime

**Example**

See CSV_GetElapsedTime.

### 6.5.12 CSV_GetLRStatus

**Description**
Returns the status of the system.

**Declaration**

```basic
CSV_GetLRStatus( iLRStatus AS Integer )
```

**Remarks**
The routine CSV_GetLRStatus returns the mode of the system. The system can either be in local or in Remote mode. For Release 1.0 this function always delivers local mode as an answer.
### TPS_Sim

Always delivers LOCAL_MODE.

**Note** This function is reserved for future purposes and has no special usage in the current implementation.

**Parameters**

- **iLRStatus**: The mode of the system. Possible values for the `iLRStatus` are:
  
<table>
<thead>
<tr>
<th>Mode</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCAL_MODE</td>
<td>0</td>
<td>local mode</td>
</tr>
<tr>
<td>REMOTE_MODE</td>
<td>1</td>
<td>Remote mode</td>
</tr>
</tbody>
</table>

**Example**

The example uses the routine `CSV_GetLRStatus` to get the mode of the system.

```bbcode
DIM iLRStatus AS Integer
CSV_GetLRStatus( iLRStatus )
MMI_PrintInt( 0, 0, 10, iLRStatus, TRUE )
```

### CSV_SetGuideLight

**Description**

Set the guide light intensity.

**Declaration**

```bbcode
CSV_SetGuideLight( BYVAL iLight AS Integer )
```

**Remarks**

Sets the guide light intensity.

**Parameters**

- **iLight**: Guide light intensity

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_EGL_OFF</td>
<td>Switching off</td>
</tr>
<tr>
<td>CSV_EGL_LOW</td>
<td>Low intensity</td>
</tr>
<tr>
<td>CSV_EGL_MID</td>
<td>Middle intensity</td>
</tr>
<tr>
<td>CSV_EGL_HIGH</td>
<td>High intensity</td>
</tr>
</tbody>
</table>

**Return Codes**

- **RC_SYSBUSY**: EDM is busy. Guide light cannot be switched.
- **RC_NOT_IMPL**: Guide light Hardware is not available
Example | Switch off the Light guide.
---|---
| CSV_SetGuideLight( CSV_EGL_OFF )

### 6.5.14 CSV_Laserpointer

**Description**
Switch on / off the laser pointer.

**Declaration**
```plaintext
CSV_Laserpointer( BYVAL lLaser AS Logical )
```

**Remarks**
Switches on / off the laser pointer.

**Parameters**
- `lLaser` **in** Switch on / off the Laser pointer (TRUE = on, FALSE = off)

**Return Codes**
- `RC_SYSBUSY` EDM is busy. Laser pointer cannot be switched.
- `RC_NOT_IMPL` Laser pointer Hardware is not available.

**Example**
Switch off the laser pointer.
```plaintext
CSV_Laserpointer( FALSE )
```

### 6.5.15 CSV_MakePositioning

**Description**
Do an absolute positioning.

**Declaration**
```plaintext
CSV_MakePositioning(BYVAL dHz AS Double,
 BYVAL dV AS Double)
```

**Remarks**
Absolute positioning of the Theodolite axes to the desired angles with the currently active tolerance for positioning. If any control function is active at the point of call, it will be cancelled and the positioning will be performed. After the positioning the controller will be automatically activated for manual input for the moving device. When starting the positioning the calling application has to take care that a valid inclination plane is available for an angle measure, as it can normally not be redone during positioning. The positioning is done with the planes valid at the beginning of
it. During the process no inclination will be measured. The used positioning method can cause inexact results, especially for steep \( V > \sim 25 \text{ GON} \)

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dHz</td>
<td>in Corrected Hz-angle [Radiant]</td>
</tr>
<tr>
<td>dV</td>
<td>in Corrected V-angle [Radiant]</td>
</tr>
</tbody>
</table>

**Return Codes**

- **RC_IVPARAM**: No valid positioning angle.
- **CSV_DETENT_ERROR**: Target angle is out of the limits or a collision is occurred.
- **CSV_TIMEOUT**: Time out at positioning of one or both axes.
- **CSV_MOTOR_ERROR**: Error in subsystem.
- **CSV_ANGLE_ERROR**: Error at measuring the angle.
- **RC_FATAL**: Fatal error.
- **RC_ABORT**: System abort.

**See Also**

- BAP_PosTelescope

**Example**

Perform an absolute positioning.

```plaintext
CSV_MakePositioning( 0, 2*atn(1) ) \ ' (0, \pi/2)
```

### 6.5.16 CSV_ChangeFace

**Description**

Do an absolute positioning to the opposite.

**Declaration**

```plaintext
CSV_ChangeFace()
```

**Remarks**

Perform positioning into the position opposite to the current. If any control function is active at the point of call, it will be cancelled and the positioning will be performed. After the positioning the controller will be automatically activated for manual input for the moving device. When starting the positioning the calling application has to take care that a valid inclination plane is available for an angle measure, as it can normally not be redone during positioning.
The positioning is done with the planes valid at the beginning of it. During the process no inclination will be measured. The used positioning method can cause inexact results, especially for steep $V > \sim 25$ GON

**Parameters**

none

**Return Codes**

- **RC_IVPARAM**  
  No valid positioning angle.
- **CSV_DETENT_ERROR**  
  Target angle is out of the limits or a collision is occurred.
- **CSV_TIMEOUT**  
  Time out at positioning of one or both axes
- **CSV_MOTOR_ERROR**  
  Error in subsystem
- **CSV_ANGLE_ERROR**  
  Error at measuring the angle
- **RC_FATAL**  
  Fatal error
- **RC_ABORT**  
  System abort

**See Also**

BAP_PosTelescope

**Example**

Perform a change of face.

```plaintext
CSV_ChangeFace()
```

### 6.5.17 CSV_SetLockStatus

**Description**

Sets the current state of the locking facility.

**Declaration**

```plaintext
CSV_SetLockStatus(BYVAL lOn AS Logical )
```

**Remarks**

It switches the locking facility on or off.

**Parameters**

- **lOn** in
  Switches on / off the locking facility
  (TRUE = on, FALSE = off)
6. System Functions

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_FATAL</td>
<td>fatal error</td>
</tr>
<tr>
<td>RC_NOT_IMPL</td>
<td>if ATR hardware is not available</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>system abort</td>
</tr>
</tbody>
</table>

See Also
CSV_SetLockStatus,
CSV_LockIn,
CSV_LockOut

Example
Perform an absolute positioning.

```
CSV_SetLockStatus( TRUE ) ' switches locking on
```

6.5.18 CSV_GetLockStatus

Description
Gets the current state of the locking facility.

Declaration
```
CSV_GetLockStatus( lOn AS Logical )
```

Remarks
It queries the TPS system if the locking facility is on or off.

Parameters
```
lOn out meaning
FALSE   Locking is switched off.
TRUE    Locking is switched on.
```

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_FATAL</td>
<td>fatal error</td>
</tr>
<tr>
<td>RC_NOT_IMPL</td>
<td>if ATR hardware is not available</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>system abort</td>
</tr>
</tbody>
</table>

See Also
CSV_GetLockStatus,
CSV_LockIn,
CSV_LockOut

Example
Perform an absolute positioning.

```
DIM l AS Logical
CSV_SetLockStatus( l ) ' queries locking
```
6.5.19 CSV_LockIn

**Description**
Starts the locking facility.

**Declaration**
CSV_LockIn( )

**Remarks**
If ATR is switched on then locking to the target will be done. If no target available, then manual positioning will be started.

**Parameters**
none

**Return Codes**
- **AUT_RC_NOT_ENABLED**
  Theodolite without ATR or lock status not set
- **AUT_RC_MOTOR_ERROR**
  Error at motor control.
- **AUT_RC_DETECTOR_ERROR**
  Error at ATR
- **AUT_RC_NO_TARGET**
  No target at the detection range
- **AUT_RC_BAD_ENVIRONMENT**
  Bad environment at the detection range (bad light...)
- **RC_NOT_IMPL**
  if ATR hardware is not available

**See Also**
CSV_GetLockStatus,
CSV_SetLockStatus,
CSV_LockOut

**Example**
This example starts locking.

CSV_LockIn( )
6.5.20 CSV_LockOut

**Description**
Stops a running locking function.

**Declaration**
CSV_LockOut()

**Parameters**
none

**Return Codes**
- RC_OK: no error
- RC_NOT_IMPL: if ATR hardware is not available

**See Also**
CSV_GetLockStatus, CSV_SetLockStatus,
CSV_LockIn

**Example**
This example stops locking.

```plaintext
CSV_LockOut()
```

6.5.21 CSV_SetATRStatus

**Description**
Sets the current state of Automatic Target Recognition.

**Declaration**
CSV_SetATRStatus(BYVAL lOn AS Logical)

**Remarks**
It switches the ATR facility on or off.

**Parameters**
- lOn in
  Switches on / off the ATR facility
  (TRUE = on, FALSE = off)

**Return Codes**
- RC_FATAL: fatal error
- RC_ABORT: system abort
- RC_NOT_IMPL: if ATR hardware is not available

**Example**
Perform an absolute positioning.

```plaintext
CSV_SetATRStatus(TRUE) ' switches ATR on
```
6.5.22 CSV_GetATRStatus

Description: Gets the current ATR state.

Declaration: CSV_GetATRStatus(lOnl AS Logical )

Remarks: It queries the TPS system if the ATR facility is on or off.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>lOn</td>
<td>out</td>
<td>FALSE: ATR is switched off. TRUE: ATR is switched on.</td>
</tr>
</tbody>
</table>

Return Codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_FATAL</td>
<td>fatal error</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>system abort</td>
</tr>
<tr>
<td>RC_NOT_IMPL</td>
<td>if ATR hardware is not available</td>
</tr>
</tbody>
</table>

Example:

Get current ATR status.

```basic
DIM l AS Logical
CSV_SetATRStatus( l )
```

6.5.23 CSV_Delay

Description: This routine delays the execution of a program.

Declaration: CSV_Delay( BYVAL iDelay AS Integer )

Remarks: This routine delay using the operating system, that means that other Theodolite tasks can run during the delay (It is not a busy waiting).

Note: Avoid busy waiting using FOR - or WHILE loops.

TPS_Sim: Delay resolution is one second. iDelay < 500 means no delay.
Parameters

iDelay in Time to delay [ms]

Example

This example „waits“ 2 seconds until it goes on.
CSV_Delay( 2000 )

6.5.24 CSV_SetTargetType

Description
Set the target type for distance measurements.

Declaration
CSV_SetTargetType( 
    BYVAL iTARGET as Integer )

Remarks
This routine sets the target type for distance measurements. The target type defines if the next distance measurement happens with prism or without prism.

Parameters

iTargetType in Target type

Valid target types

- CSV_WITH_REFLECTOR
- CSV_WITHOUT_REFLECTOR

Meaning

- With reflector
- Without reflector

Return-Codes

RC_OK Successful termination.
RC_IVPARAM Instrument don’t support this target type
6.5.25 CSV_GetTargetType

Description: Get the target type for distance measurements.

Declaration: CSV_GetTargetType( iTargetType as Integer )

Remarks: This routine fetches the target type for distance measurements. The target type defines if the next distance measurement happens with prism or without prism.

Parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iTargetType</td>
<td>Integer</td>
<td>Target type</td>
</tr>
</tbody>
</table>

Valid target types:
- CSV_WITH_REFLECTOR: With reflector
- CSV_WITHOUT_REFLECTOR: Without reflector

Return-Codes:
- RC_OK: Successful termination.

See: CSV_SetTargetType, BAP_SetMeasPrg, BAP_GetMeasPrg

Example: The example fetches the target type.

```basic
DIM iTargetType AS Integer
CSV_GetTargetType(iTargetType)
```
6.5.26 CSV_SetPrismType

Description  Sets the used prism.

Declaration  CSV_SetPrismType( BYVAL iPrism as Integer)

Remarks  This routine sets the used prism iPrism(BAP_PRISM_ROUND, BAP_PRISM_TAPE, BAP_PRISM_MINI, BAP_PRISM_360, BAP_PRISM_USER1, BAP_PRISM_USER2 or BAP_PRISM_USER3). If iPrism is one of the user defined prisms and this prism is actually not defined then this routine will return RC_IVRESULT.

Parameters  

iPrism  in  Used prism

Return-Codes  

RC_OK  Successful termination.
RC_IVRESULT  Prism not defined.

See  CSV_GetPrismType

Example  The example sets the 360 degrees prism.

CSV_SetPrismType(BAP_PRISM_360)

6.5.27 CSV_GetPrismType

Description  Returns the used prism.

Declaration  CSV_GetPrismType(iPrism as Integer)

Remarks  This routine returns the used prism iPrism.

Parameters  

iPrism  out  Used prism

Return-Codes  

RC_OK  Successful termination.
### CSV_SetLaserPlummet

**Description**
Switches the laser plummet.

**Declaration**
```
CSV_SetLaserPlummet( BYVAL lOn as Logical )
```

**Remarks**
This function switches the optional laser plummet. The plummet will be switched off automatically after 3 minutes.

**Parameters**
- **lOn**
  - **in**
  - **TRUE:** switch plummet on

**Return-Codes**
- **RC_OK**
  - Successful termination.

**See**
- CSV_GetLaserPlummet
- CSV_GetInstrumentFamily

### CSV_GetLaserPlummet

**Description**
Returns the laser plummet state.

**Declaration**
```
CSV_GetLaserPlummet( lOn as Logical )
```

**Remarks**
This function returns the state of the optional laser plummet.

**Parameters**
- **lOn**
  - **out**
  - **TRUE:** plummet is switched on

**Return-Codes**
- **RC_OK**
  - Successful termination.
6.5.30 CSV_CheckAltUserTask

**Description**
Returns if an alternative user-task was running.

**Declaration**
CSV_CheckAltUserTask(lWasRunning AS Logical)

**Remarks**
This routine returns if an alternative user-task was running. One of these tasks can be started by pressing one of the buttons FNC, Shift-FNC, PROG, Shift-PROG, Light and Level.

Functions, executed by an alternative user task, can change several system settings. The CSV_CheckAltUserTask routine notifies the running GeoBASIC application that it was interrupted by another program. With this information, the GeoBASIC program is able to respond to these changes.

After processing this information, the subroutine CSV_ResetAltUserTask must be called.

**Parameters**
lWasRunning out TRUE: a task was running

**Return-Codes**
RC_OK Successful termination.

**See**
CSV_ResetAltUserTask

**Example**
The example checks if an alternative task was running.

```basic
CSV_CheckAltUserTask( l )
IF l THEN
   send("AltUserTask: was running")
ELSE
   send("AltUserTask: was NOT running")
END IF
CSV_ResetAltUserTask( )
```

6.5.31 CSV_ResetAltUserTask

**Description**
Resets the “alternative user-task was running” flag.
### Declaration

CSV_ResetAltUserTask()

### Remarks

This routine restarts the alternative user-task tracking.

### Parameters

none

### Return-Codes

- **RC_OK**  
  Successful termination.

### See

CSV_CheckAltUserTask

---

### 6.5.32 CSV_SysCall

#### Description

Call a system function.

#### Declaration

CSV_SysCall( BYVAL CId AS CIdType)

#### Remarks

This routine works in two different forms depending on the parameter CId. If CId is a system function CSV_SysCall calls the function directly. In the other form the CId is a system event. In this case CSV_SysCall calls the system function (or dialog, menu, macro, application) which is defined in the current configuration to handle this event. See description of the system functions and system events in the appendix H.

#### Parameters

- **CId**  
  in  
  System function or system event

#### Return-Codes

- **RC_OK**  
  Successful termination.
- **RC_IVPARAM**  
  No function defined to handle the event
- **RC_NOT_IMPL**  
  System function not available
See CSV_SysCallAvailable

Example The example calls the system function electronic level.

CSV_SysCall(CSV_SFNC_Libelle)

6.5.33 CSV_SysCallAvailable

Description Check if system function is available.

Declaration CSV_SysCallAvailable(
    BYVAL CId AS CIdType,
    lAvailable AS Logical )

Remarks This routine checks, if it is possible to call the function CId if CId is a system function or if there is a function defined and available to handle the event CId if CId is an system event. See the description of system functions and system events in appendix H.

Parameters

CId in System function or system event.

lAvailable out TRUE: System function is available or function (dialog, menu, macro, application) to handle the event is defined and available.

Return-Codes

RC_OK Successful termination.
6. System Functions

See CSV_SysCall

Example The example checks if the red laser is available.

```basic
DIM lAvailable AS Logical
CSV_SysCallAvailable(CSV_SFNC_ToggleRedLaser, lAvailable)
```

6.5.34 CSV_LibCall

Description Call a GeoBASIC or C application routine of another program.

Declaration `CSV_LibCall( BYVAL PrgName AS String255, BYVAL FuncName AS String255, BYVAL CptShort AS _Token )`

Remarks This routine is used to call a GeoBASIC routine which is defined in another program. Please refer also to Appendix

Parameters

- `PrgName` in Program name
- `FuncName` in Function name
- `CptShort` In Short caption for dialogs

Return-Codes

| RC_OK | Successful termination. |

See CSV_LibCallAvailable

Example See IAC.GBS and IAC2.GBS for an example.

6.5.35 CSV_LibCallAvailable

Description Check if the GeoBASIC routine from another program is available.

Declaration `CSV_LibCallAvailable( BYVAL PrgName AS String255, BYVAL FuncName AS String255, lAvailable AS Logical )`
Remarks
This routine checks if a GeoBASIC routine which is defined in another program is available. Usually this means that it checks if the other program is loaded and the specified entry point exists.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgName</td>
<td>in</td>
<td>Program name</td>
</tr>
<tr>
<td>FuncName</td>
<td>in</td>
<td>Function name</td>
</tr>
<tr>
<td>lAvailable</td>
<td>out</td>
<td>Routine is available</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See
CSV_LibCall

Example
See IAC.GBS and IAC2.GBS for an example.
6.1.28 MMI_WriteMsgStr ................................................................. 6-46
6.1.29 MMI_DrawLine ................................................................. 6-48
6.1.30 MMI_DrawRect ................................................................. 6-49
6.1.31 MMI_DrawCircle ............................................................... 6-51
6.1.32 MMI_DrawText ................................................................. 6-52
6.1.33 MMI_DrawBusyField ........................................................... 6-53
6.1.34 MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong ........ 6-54
6.1.35 MMI_StartVarBeep ............................................................ 6-54
6.1.36 MMI_SwitchVarBeep ......................................................... 6-55
6.1.37 MMI_GetVarBeepStatus ..................................................... 6-56
6.1.38 MMI_SwitchAFKey ............................................................. 6-57
6.1.39 MMI_SwitchIconsBeep ....................................................... 6-58
6.1.40 MMI_SetAngleRelation ...................................................... 6-59
6.1.41 MMI_GetAngleRelation ...................................................... 6-60
6.1.42 MMI_SetVAngleMode ......................................................... 6-60
6.1.43 MMI_GetVAngleMode ......................................................... 6-61
6.1.44 MMI_SetAngleUnit ............................................................ 6-61
6.1.45 MMI_GetAngleUnit ............................................................ 6-63
6.1.46 MMI_SetDistUnit ............................................................... 6-63
6.1.47 MMI_GetDistUnit ............................................................... 6-65
6.1.48 MMI_SetPressUnit ............................................................. 6-65
6.1.49 MMI_GetPressUnit ............................................................. 6-67
6.1.50 MMI_SetTempUnit ............................................................ 6-67
6.1.51 MMI_GetTempUnit ............................................................ 6-68
6.1.52 MMI_SetDateFormat .......................................................... 6-69
6.1.53 MMI_GetDateFormat .......................................................... 6-70
6.1.54 MMI_SetTimeFormat ........................................................ 6-70
6.1.55 MMI_GetTimeFormat ........................................................ 6-71
6.1.56 MMI_SetCoordOrder ........................................................ 6-72
6.1.57 MMI_GetCoordOrder ........................................................ 6-73
6.1.58 MMI_SetLanguage ............................................................ 6-73
6.1.59 MMI_GetLanguage ............................................................ 6-74
6.1.60 MMI_GetLangName .......................................................... 6-75
6.2 BASIC APPLICATIONS BAP ................................................................. 6-76
  6.2.1 Summarizing Lists of BAP Types and Procedures ............... 6-76
  6.2.2 BAP_SetAccessoriesDlg ............................................................. 6-77
  6.2.3 BAP_MeasDistAngle ................................................................. 6-77
  6.2.4 BAP_MeasRec ................................................................. 6-81
  6.2.5 BAP_FineAdjust ................................................................. 6-84
  6.2.6 BAP_SearchPrism ................................................................. 6-85
  6.2.7 BAP_SetManDist ................................................................. 6-86
  6.2.8 BAP_SetPpm ................................................................. 6-87
  6.2.9 BAP_SetPrism ................................................................. 6-88
  6.2.10 BAP_SetMeasPrg ................................................................. 6-88
  6.2.11 BAP_GetMeasPrg ................................................................. 6-90
  6.2.12 BAP_PosTelescope ................................................................. 6-91
  6.2.13 BAP_SetHz ................................................................. 6-93

6.3 Measurement Functions TMC ......................................................... 6-94
  6.3.1 Summarizing Lists of TMC Types and Procedures ............... 6-94
  6.3.2 TMC Data Structures ................................................................. 6-96
  6.3.3 TMC_DoMeasure ................................................................. 6-99
  6.3.4 TMC_GetPolar ................................................................. 6-101
  6.3.5 TMC_GetCoordinate ................................................................. 6-104
  6.3.6 TMC_GetAngle ................................................................. 6-107
  6.3.7 TMC_GetAngle_WInc ................................................................. 6-109
  6.3.8 TMC_QuickDist ................................................................. 6-110
  6.3.9 TMC_GetSimpleMea ................................................................. 6-114
  6.3.10 TMC_Get/SetAngleFaceDef ................................................................. 6-117
  6.3.11 TMC_Get/SetHzOffset ................................................................. 6-118
  6.3.12 TMC_Get/SetDistPpm ................................................................. 6-119
  6.3.13 TMC_Get/SetHeight ................................................................. 6-119
  6.3.14 TMC_Get/SetRefractiveCorr ..................................................... 6-120
  6.3.15 TMC_Get/SetRefractiveMethod ..................................................... 6-120
  6.3.16 TMC_Get/SetStation ................................................................. 6-121
  6.3.17 TMC_IfDistTapeMeasured ................................................................. 6-121
  6.3.18 TMC_SetHandDist ................................................................. 6-122
GeoBASIC Reference Manual

6.3.19 TMC_SetDistSwitch ................................................................. 6-123
6.3.20 TMC_GetDistSwitch ............................................................... 6-123
6.3.21 TMC_SetOffsetDist ............................................................... 6-124
6.3.22 TMC_GetOffsetDist ............................................................... 6-125
6.3.23 TMC_IfOffsetDistMeasured ................................................... 6-125
6.3.24 TMC_GetFace1 ..................................................................... 6-126
6.3.25 TMC_SetAngSwitch ............................................................... 6-126
6.3.26 TMC_GetAngSwitch ............................................................... 6-127
6.3.27 TMC_SetInclineSwitch .......................................................... 6-127
6.3.28 TMC_GetInclineSwitch .......................................................... 6-128
6.3.29 TMC_GetInclineStatus .......................................................... 6-128

6.4 Functions for GSI.................................................................................... 6-130
6.4.1 Summarizing Lists of GSI Types and Procedures..................... 6-130
6.4.2 Constants for WI values............................................................ 6-132
6.4.3 Constants for Measurement Dialog Definition ......................... 6-135
6.4.4 Relationship of GSI_ID’s to GSI_PAR’s................................. 6-139
6.4.5 Data Structures for GSI Functions ............................................ 6-142
6.4.6 GSI_GetRunningNr ................................................................. 6-144
6.4.7 GSI_SetRunningNr ................................................................. 6-145
6.4.8 GSI_GetIndivNr ...................................................................... 6-145
6.4.9 GSI_SetIndivNr ...................................................................... 6-146
6.4.10 GSI_IsRunningNr .................................................................... 6-146
6.4.11 GSI_SetIvPtNrStatus .............................................................. 6-147
6.4.12 GSI_IncPNumber................................................................. 6-148
6.4.13 GSI_Coding ............................................................................ 6-148
6.4.14 GSI_SelectCode ..................................................................... 6-149
6.4.15 GSI_GetQCodeAvailable......................................................... 6-149
6.4.16 GSI_SetQCodeMode ............................................................... 6-150
6.4.17 GSI_ExecQCoding ................................................................. 6-150
6.4.18 GSI_SetRecOrder ................................................................. 6-152
6.4.19 GSI_GetRecOrder ................................................................. 6-152
6.4.20 GSI_QuickSet ....................................................................... 6-153
6.4.21 GSI_SetRecPath ................................................................. 6-153
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.22 GSI_GetRecPath</td>
<td>6-154</td>
</tr>
<tr>
<td>6.4.23 GSI_SetDataPath</td>
<td>6-155</td>
</tr>
<tr>
<td>6.4.24 GSI_GetDataPath</td>
<td>6-156</td>
</tr>
<tr>
<td>6.4.25 GSI_GetWiEntry</td>
<td>6-156</td>
</tr>
<tr>
<td>6.4.26 GSI_SetWiEntry</td>
<td>6-157</td>
</tr>
<tr>
<td>6.4.27 GSI_GetRecMask</td>
<td>6-158</td>
</tr>
<tr>
<td>6.4.28 GSI_SetRecMask</td>
<td>6-159</td>
</tr>
<tr>
<td>6.4.29 GSI_SetRecMaskNr</td>
<td>6-161</td>
</tr>
<tr>
<td>6.4.30 GSI_GetRecMaskNr</td>
<td>6-162</td>
</tr>
<tr>
<td>6.4.31 GSI_DefineRecMaskDlg</td>
<td>6-162</td>
</tr>
<tr>
<td>6.4.32 GSI_ManCoordDlg</td>
<td>6-162</td>
</tr>
<tr>
<td>6.4.33 GSI_ImportCoordDlg</td>
<td>6-165</td>
</tr>
<tr>
<td>6.4.34 GSI_SetLineSysMDlg</td>
<td>6-168</td>
</tr>
<tr>
<td>6.4.35 GSI_GetLineSysMDlg</td>
<td>6-169</td>
</tr>
<tr>
<td>6.4.36 GSI_SetMDlgNr</td>
<td>6-170</td>
</tr>
<tr>
<td>6.4.37 GSI_GetMDlgNr</td>
<td>6-171</td>
</tr>
<tr>
<td>6.4.38 GSI_CreateMDlg</td>
<td>6-171</td>
</tr>
<tr>
<td>6.4.39 GSI_SetLineMDlg</td>
<td>6-173</td>
</tr>
<tr>
<td>6.4.40 GSI_SetLineMDlgText</td>
<td>6-174</td>
</tr>
<tr>
<td>6.4.41 GSI_SetLineMDlgPar</td>
<td>6-175</td>
</tr>
<tr>
<td>6.4.42 GSI_UpdateMDlg</td>
<td>6-177</td>
</tr>
<tr>
<td>6.4.43 GSI_DefineMDlg</td>
<td>6-178</td>
</tr>
<tr>
<td>6.4.44 GSI_UpdateMeasurement</td>
<td>6-178</td>
</tr>
<tr>
<td>6.4.45 GSI_Measure</td>
<td>6-179</td>
</tr>
<tr>
<td>6.4.46 GSI_ExecuteAutoDist</td>
<td>6-180</td>
</tr>
<tr>
<td>6.4.47 GSI_CheckTracking</td>
<td>6-180</td>
</tr>
<tr>
<td>6.4.48 GSI_RecordRecMask</td>
<td>6-181</td>
</tr>
</tbody>
</table>

6.5 Central Service Functions CSV ................................................... 6-183

6.5.1 Summarizing Lists of CSV Types and Procedures ................. 6-183
6.5.2 Data Structures for the Central Service Functions ............... 6-185
6.5.3 CSV_GetDateTime ............................................................... 6-187
6.5.4 CSV_GetTemperature .......................................................... 6-188
6.5.5 CSV_GetInstrumentName ....................................................... 6-188
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_GetInstrumentNo</td>
<td>6-189</td>
</tr>
<tr>
<td>CSV_GetInstrumentFamily</td>
<td>6-189</td>
</tr>
<tr>
<td>CSV_GetSWVersion</td>
<td>6-190</td>
</tr>
<tr>
<td>CSV_GetGBIVersion</td>
<td>6-191</td>
</tr>
<tr>
<td>CSV_GetElapseSysTime</td>
<td>6-192</td>
</tr>
<tr>
<td>CSV_GetSysTime</td>
<td>6-193</td>
</tr>
<tr>
<td>CSV_GetLRStatus</td>
<td>6-193</td>
</tr>
<tr>
<td>CSV_SetGuideLight</td>
<td>6-194</td>
</tr>
<tr>
<td>CSV_Laserpointer</td>
<td>6-195</td>
</tr>
<tr>
<td>CSV_MakePositioning</td>
<td>6-195</td>
</tr>
<tr>
<td>CSV_ChangeFace</td>
<td>6-196</td>
</tr>
<tr>
<td>CSV_SetLockStatus</td>
<td>6-197</td>
</tr>
<tr>
<td>CSV_GetLockStatus</td>
<td>6-198</td>
</tr>
<tr>
<td>CSV_LockIn</td>
<td>6-199</td>
</tr>
<tr>
<td>CSV_LockOut</td>
<td>6-200</td>
</tr>
<tr>
<td>CSV_SetATRStatus</td>
<td>6-200</td>
</tr>
<tr>
<td>CSV_GetATRStatus</td>
<td>6-201</td>
</tr>
<tr>
<td>CSV_Delay</td>
<td>6-201</td>
</tr>
<tr>
<td>CSV_SetTargetType</td>
<td>6-202</td>
</tr>
<tr>
<td>CSV_GetTargetType</td>
<td>6-203</td>
</tr>
<tr>
<td>CSV_SetPrismType</td>
<td>6-204</td>
</tr>
<tr>
<td>CSV_GetPrismType</td>
<td>6-204</td>
</tr>
<tr>
<td>CSV_SetLaserPlummet</td>
<td>6-205</td>
</tr>
<tr>
<td>CSV_GetLaserPlummet</td>
<td>6-205</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>6-206</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>6-206</td>
</tr>
<tr>
<td>CSV.SysCall</td>
<td>6-207</td>
</tr>
<tr>
<td>CSV.SysCallAvailable</td>
<td>6-208</td>
</tr>
<tr>
<td>CSV.LibCall</td>
<td>6-209</td>
</tr>
<tr>
<td>CSV.LibCallAvailable</td>
<td>6-209</td>
</tr>
</tbody>
</table>
6.1 MMI FUNCTIONS

6.1.1 Summarising Lists of MMI Types and Procedures

### 6.1.1.1 Types

<table>
<thead>
<tr>
<th>Type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListArray</td>
<td>List field Data structure</td>
</tr>
<tr>
<td>sLine</td>
<td>Display line</td>
</tr>
</tbody>
</table>

### 6.1.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_AddButton</td>
<td>Add a Button to a dialog.</td>
</tr>
<tr>
<td>MMI_AddGBMenuButton</td>
<td>Adds a button to a menu</td>
</tr>
<tr>
<td>MMI_BeepAlarm</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_BeepLong</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_BeepNormal</td>
<td>Create an alert beep.</td>
</tr>
<tr>
<td>MMI_CheckButton</td>
<td>Checks if a button was pressed.</td>
</tr>
<tr>
<td>MMI_CreateGBMenu</td>
<td>Creates a menu</td>
</tr>
<tr>
<td>MMI_CreateGBMenuItem</td>
<td>Creates an item to an existing menu</td>
</tr>
<tr>
<td>MMI_CreateGBMenuStr</td>
<td>Creates an item with a variable string</td>
</tr>
<tr>
<td>Str</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateGBMenuStr</td>
<td>Creates a menu with variable strings</td>
</tr>
<tr>
<td>MMI_CreateGraphDialog</td>
<td>Create and show a graphics dialog.</td>
</tr>
<tr>
<td>MMI_CreateMenuItem</td>
<td>Creates a menu item on the Theodolite menu.</td>
</tr>
<tr>
<td>MMI_CreateTextDialog</td>
<td>Create and show a text dialog.</td>
</tr>
<tr>
<td>MMI_DeleteButton</td>
<td>Delete a button from a dialog.</td>
</tr>
<tr>
<td>MMI_DeleteDialog</td>
<td>Deletes a dialog.</td>
</tr>
<tr>
<td>MMI_DeleteGBMenu</td>
<td>Deletes a menu</td>
</tr>
<tr>
<td>MMI_DrawBusyField</td>
<td>Shows or hides the Busy-Icon</td>
</tr>
<tr>
<td>MMI_DrawCircle</td>
<td>Draw a circle / ellipse.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>MMI_DrawLine</td>
<td>Draw a line.</td>
</tr>
<tr>
<td>MMI_DrawRect</td>
<td>Draw a rectangle.</td>
</tr>
<tr>
<td>MMI_DrawText</td>
<td>Draw / delete text.</td>
</tr>
<tr>
<td>MMI_FormatVal</td>
<td>Convert a value to a string.</td>
</tr>
<tr>
<td>MMI_GetAngleRelation</td>
<td>Request the current angle relationships.</td>
</tr>
<tr>
<td>MMI_GetAngleUnit</td>
<td>Return the currently displayed unit of angle.</td>
</tr>
<tr>
<td>MMI_GetButton</td>
<td>Get the button identifier of the pressed button.</td>
</tr>
<tr>
<td>MMI_GetCoordOrder</td>
<td>Retrieve the co-ordinate order.</td>
</tr>
<tr>
<td>MMI_GetDateFormat</td>
<td>Retrieves the date display format.</td>
</tr>
<tr>
<td>MMI_GetDistUnit</td>
<td>Return the currently displayed unit of distance.</td>
</tr>
<tr>
<td>MMI_GetLangName</td>
<td>Gets the name to a language number.</td>
</tr>
<tr>
<td>MMI_GetLanguage</td>
<td>Query the current language.</td>
</tr>
<tr>
<td>MMI_GetPressUnit</td>
<td>Return the currently displayed unit of pressure.</td>
</tr>
<tr>
<td>MMI_GetTempUnit</td>
<td>Return the currently displayed unit of temperature.</td>
</tr>
<tr>
<td>MMI_GetTimeFormat</td>
<td>This function retrieves the format used to display the time.</td>
</tr>
<tr>
<td>MMI_GetVAngleMode</td>
<td>Returns the V-Angle mode</td>
</tr>
<tr>
<td>MMI_GetVarBeepStatus</td>
<td>Read the switch status for a variable signal beep.</td>
</tr>
<tr>
<td>MMI_InputInt</td>
<td>Get an integer input value in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputList</td>
<td>Shows a list field in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputStr</td>
<td>Get a string input in a text dialog.</td>
</tr>
<tr>
<td>MMI_InputVal</td>
<td>Get a numerical input value in a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintInt</td>
<td>Print an integer value on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintStr</td>
<td>Print a string on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintTok</td>
<td>Print a token on a text dialog.</td>
</tr>
<tr>
<td>MMI_PrintVal</td>
<td>Print a value on a text dialog.</td>
</tr>
<tr>
<td>MMI_SelectGBMenuItem</td>
<td>Select a menu item</td>
</tr>
<tr>
<td>MMI_SetAngleRelation</td>
<td>Set the angle relationship.</td>
</tr>
<tr>
<td>MMI_SetAngleUnit</td>
<td>Set the displayed unit of angle.</td>
</tr>
<tr>
<td>MMI_SetCoordOrder</td>
<td>Set the co-ordinate order.</td>
</tr>
</tbody>
</table>
### MMI Data Types

#### ListArray – List field data structure

**Description**
This array is used for list fields and consists of `LIST_ARRAY_MAX_ELEMENT` (200) elements of the type `STRING30`.

**Note**
Each variable of this data type reserves 6400 Bytes.

#### sLine – Display line

**Description**
This type is used to define a string with 29 characters, which is necessary to print variable strings on the display. The length depends on the actual display width, which is 29 for TPS1100 instruments.
**6.1.3 MMI_CreateMenuItem**

**Description**  Creates a system menu item on the Theodolite menu to establish the invocation of a GeoBASIC application.

**Declaration**

```plaintext
MMI_CreateMenuItem(
    BYVAL sAppName AS String,
    BYVAL sFuncName AS String,
    BYVAL iMenuNum AS Integer,
    BYVAL sMenuText AS _Token )
```

**Remarks**  The `CreateMenuItem` creates a menu item in a system menu with the text `MenuText` on the chosen entry point `MenuNum` in the menu-system. By clicking the new menu item on the Theodolite, the subroutine with the name `FuncName` in the Program `AppName` will be executed. The number of applications which can be loaded at a time are limited to 25. The maximum number of entry points over all applications (C and GeoBASIC applications) is 50. All `GLOBAL` declared subroutines count as entry points. Be aware of the fact that the interpreter and a possible Coding function also count for the number of application. The same is true for any C-application which has been loaded onto the TPS.

**Note**  The subroutine denoted in `sFuncName` must be declared as `GLOBAL`. The intended use for this procedure is during the installation phase only!

**Parameters**

- `sAppName` in  The name of the program where the function or subroutine is defined.
- `sFuncName` in  The name of the global function or subroutine to be called.
- `iMenuNum` in  Defines in which menu the menu-entry is generated. There are three possible menus where a menu item can be added. For multiple menu items the menus can be combined with `+'-operator.
valid menus | meaning
---|---
MMI_MENU_PROGRAMS | Add to menu "Main menu"
MMI_MENU_PROGMENU | Add to "PROG" - Key menu
MMI_MENU_AUTOEXEC | Add to menu "Autoexec"

sMenuText in The text of the menu-entry which should be displayed on the Theodolite.

Return-Codes

RC_OK Successful termination.

**Note** Since this procedure will be called during installation phase you do not have the possibility to do any error handling. Only the loader will report an error which may be caused by an erroneous call.

Example

The example uses the MMI_CreateMenuItem routine to create a menu entry named "START THE PROGRAM" under the main menu. The function "Main" in the GeoBASIC program "ExampleProgram" will be called when this menu item is selected.

```plaintext
MMI_CreateMenuItem( "ExampleProgram", "Main", MMI_MENU_PROGRAMS, "START THE PROGRAM" )
```

6.1.4 MMI_CreateGBMenu

**Description** Creates a menu.

**Declaration**

```plaintext
MMI_CreateGBMenu( 
  BYVAL sMenuName AS _Token, 
  iMenuId AS Integer )
```

**Remarks** This routine creates an empty menu and the caption sMenuName. The function MMI_CreateGBMenuItem adds items to a menu.
Before terminating a GeoBASIC program, all menus must be deleted.

The GeoBASIC menus system has the following limitations:
- The maximal number of menus for a GeoBASIC program is 5.
- The maximal number of items / menu is 49.
- The maximal number of items over all menus plus menus is 254.

Parameters

- **sMenuName** in

  The caption of the menu.

- **iMenuId** out

  Returned menu identifier. It is the handle for using this menu.

Return-Codes

- **RC_OK**
  
  Successful termination.

- **MMI_NOMORE_MENUS**
  
  No more menus available

See Also

- MMI_CreateGBMenuItem, MMI_DeleteGBMenu, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example

The example creates a menu with a button. For a complete example see sample program MENU.GBS

```
CONST MHELP = "Help for measurement type...."

DIM iMenu AS Integer ' menu identifier
DIM iSelection AS Integer ' selected item
DIM iButton AS Integer ' used button

'MCreate main menu
MMI_CreateGBMenu("MEASUREMENT TYPE", iMenu)
```
Create menu items - all items use the same help text

`MMI_CreateGBMenuItem(iMenu, "Polygon", MHELP)`
`MMI_CreateGBMenuItem(iMenu, "Border point", MHELP)`
`MMI_CreateGBMenuItem(iMenu, "Situation point", MHELP)`

Create the button supported in this menu

`MMI_AddGBMenuButton(iMenu, MMI_F5_KEY, "EXIT ")`

Show and execute menu

`MMI_SelectGBMenuItem(iMenu, "TEST", iSelection, iButton)`

```
SELECT CASE iSelection
    CASE 1 ' Polygon
    ... CASE ELSE
        MMI_BeepAlarm()
    END SELECT
    MMI_DeleteGBMenu(iMenu)
```

### 6.1.5 MMI_CreateGBMenuItem

**Description**
Creates an item in an existing menu.

**Declaration**
```
MMI_CreateGBMenuItem(
    BYVAL iMenuId AS Integer,
    BYVAL sMenuItemName AS _Token,
    BYVAL sHelpText AS _Token )
```

**Remarks**
This function adds one menu item to an existing menu `iMenuId`. This item will be displayed as the last item.

**Parameters**
- `iMenuId` in Menu identifier
- `sMenuItemName` in Displayed text
- `sHelpText` in Help text; only visible if the help functionality of theodolite is enabled

**Return-Codes**
6. System Functions

RC_OK Successful termination.
BAS_MENU_ID_INVALID Bad iMenuId
BAS_MENU_TABLE_FULL No more free menu items

See Also MMI_CreateGBMenu, MMI_DeleteGBMenu,
MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example see MMI_CreateGBMenu

6.1.6 MMI_CreateGBMenuStr

Description Creates a menu with variable strings as menu name and menu items.

Declaration MMI_CreateGBMenuStr(
    BYVAL sMenuName AS sLine,
    iMenuId AS Integer )

Remarks This routine creates an empty menu and the caption sMenuName. sMenuName need not be constant, it can be generated during the execution of the program. The function MMI_CreateGBMenuItemStr adds items to this kind of menu.

Note Before terminating a GeoBASIC program, all menus must be deleted.

The GeoBASIC menus system has the following limitations:

The maximal number of menus for a GeoBASIC program is 5.
The maximal number of items / menu is 49.
The maximal number of items over all menus plus menus is 254.

Parameters

sMenuName in The caption of the menu.
iMenuId  out  Returned menu identifier. It is the handle for using this menu.

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>MMI_NOMORE_</td>
<td>No more menus available</td>
</tr>
<tr>
<td>MENUS</td>
<td></td>
</tr>
</tbody>
</table>

See Also

MMI_CreateGBMenuItemStr, MMI_DeleteGBMenu, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

Example

The example creates a menu with a button. The menu name is a composition with a constant string and the instrument name. The menu item names are extended with the current language name.

```
CONST MHELP = "Help for measurement type...."

DIM iMenu AS Integer ' menu identifier
DIM iSelection AS Integer ' selected item
DIM iButton AS Integer ' used button
DIM sMenuName AS sLine ' menu name
DIM sMenuItemName1 AS sLine ' menu item 1 name
DIM sMenuItemName2 AS sLine ' menu item 2 name
DIM iLangNr AS Integer ' language number
DIM sLangName AS String20' language name
DIM sInstrumentName AS String30' instrument name

' generate menu name
CSV_GetInstrumentName(sInstrumentName)
sMenuName = "Programs on " + sInstrumentName
' Create menu
MMI_CreateGBMenuStr(sMenuName, iMenu)
' generate menu item names
MMI_GetLanguage(iLangNr, sLangName)
sMenuItemName1 = "Polygon in " + sLangName
sMenuItemName2 = "Border point in " + sLangName
' Create menu items - all items use the same help text
MMI_CreateGBMenuItemStr(iMenu, sMenuItemName1, MHELP)
MMI_CreateGBMenuItemStr(iMenu, sMenuItemName2, MHELP)
```
'Create the button supported in this menu
MMI_AddGBMenuButton(iMenu, MMI_F5_KEY, "EXIT ")

'show and execute menu
MMI_SelectGBMenuItem(iMenu, "TEST",
iSelection, iButton)
SELECT CASE iSelection
  CASE 1 ' Polygon
  ...  CASE ELSE
  MMI_BeepAlarm()
END SELECT
MMI_DeleteGBMenu(iMenu)

6.1.7  MMI_CreateGBMenuItemStr

Description  Creates an item with a variable string in an existing menu.

Declaration
MMI_CreateGBMenuItemStr(
  BYVAL iMenuId AS Integer,
  BYVAL sMenuItemName AS sLine,
  BYVAL sHelpText AS _Token )

Remarks  This routine adds one menu item to an existing menu iMenuId. This item will be displayed as the last item. The menu must be created with MMI_CreateGBMenuStr. sMenuItemName need not be constant, it can be generated during the execution of the program.

Parameters
  iMenuId  in  Menu identifier
  sMenuItemName  in  Displayed text
  sHelpText  in  Help text; only visible if the help functionality of the theodolite is enabled

Return-Codes
  RC_OK  Successful termination.
  BAS_MENU_ID_INVALID  Bad iMenuId
6.1.8 MMI_DeleteGBMenu

**Description**  
Deletes a menu.

**Declaration**  
```vbnet
MMI_DeleteGBMenu( BYVAL iMenuId AS Integer )
```

**Remarks**  
This function deletes the menu `iMenuId`.

**Parameters**  
- `iMenuId`  
  - in  
  - Menu identifier

**Return-Codes**  
- **RC_OK**  
  - Successful termination.
- **BAS_MENU_ID_INVALID**  
  - Bad `iMenuId`

**See Also**  
MMI_CreateGBMenu, MMI_CreateGBMenuItem, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

**Example**  
see MMI_CreateGBMenu

6.1.9 MMI_SelectGBMenuItem

**Description**  
Select a menu item.

**Declaration**  
```vbnet
MMI_SelectGBMenuItem(  
  BYVAL iMenuId AS Integer,  
  BYVAL sCaptionLeft AS _Token,  
  BYVAL iSelItem AS Integer,  
  BYVAL iButtonId AS Integer )
```

**Remarks**  
This function shows and executes a menu `iMenuId` and returns the selected item `iSelItem` or pressed button `iButtonId`.

**See Also**  
MMI_CreateGBMenu, MMI_CreateGBMenuItem, MMI_SelectGBMenuItem, MMI_AddGBMenuButton

**Example**  
see MMI_CreateGBMenu
Parameters

- `iMenuId` in: Menu identifier
- `sCaption` in: The maximal five-character long part of the title bar displayed left of the menu title, with a separation symbol.
- `iSelItem` in/out: Selected item
- `iButtonId` out: Pressed button

Return-Codes

- `RC_OK`: Successful termination.
- `BAS_MENU_ID_INVALID`: Bad `iMenuId`

See Also

- `MMI_CreateGBMenu`, `MMI_CreateGBMenuItem`, `MMI_DeleteGBMenu`, `MMI_AddGBMenuButton`

Example

see `MMI_CreateGBMenu`

6.1.10 MMI_AddGBMenuButton

Description

Adds a button to a menu.

Declaration

```plaintext
MMI_AddGBMenuButton(
    BYVAL iMenuId AS Integer,
    BYVAL iButtonId AS Integer,
    BYVAL sCaption AS _Token )
```

Remarks

This function adds a button with the identifier `iButtonId` to the menu `iMenuId` and shows the caption `sCaption`.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMenuId in</td>
<td>Menu identifier</td>
</tr>
<tr>
<td>iButtonId in</td>
<td>Identifier of the button to be added. Valid buttons are MMI_F1_KEY.. MMI_F6_KEY and MMI_SHF2_KEY.. MMI_SHF6_KEY.</td>
</tr>
<tr>
<td>sCaption in</td>
<td>Text placed onto the button (max. 5 characters)</td>
</tr>
</tbody>
</table>

Return-Codes

- **RC_OK** : Successful termination.
- **BAS_MENU_ID_INVALID** : Bad iMenuId

See Also

- MMI_CreateGBMenu, MMI_CreateGBMenuItem, MMI_DeleteGBMenu, MMI_SelectGBMenuItem

Example

see MMI_CreateGBMenu

6.1.11 MMI_CreateTextDialog

Description

Create and show a text dialog.

Declaration

```vbnet
MMI_CreateTextDialog(  
    BYVAL iLines AS Integer,  
    BYVAL sCaptionLeft AS _Token,  
    BYVAL sCaptionRight AS _Token,  
    BYVAL sHelptext AS _Token )
```

Remarks

The routine creates and shows a dialog with iLines lines, the left part of the title bar sCaptionLeft, the caption sCaptionRight and the help text sHelpText. Only one text dialog can exist at the same time. If MMI_CreateTextDialog is called while already a text dialog or a measurement dialog exists, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.
Only a text dialog or a measurement dialog is valid at a time. They cannot be defined at the same time. A graphic dialog overrides a text or measurement dialog but does not delete the definition of it.

On the dialog field strings, numerical values and list fields can be displayed or edited using the routines MMI_PrintStr, MMI_PrintVal, MMI_PrintInt, MMI_InputStr, MMI_InputVal, MMI_InputInt and MMI_InputList.

Parameters

iLines      in    The number of lines of the dialog. There are up to 12 lines possible. If the dialog has more than 6 lines, a scrollbar on the right side appear and it is possible to scroll up and down with the cursor keys.

sCaptionLeft in    The maximal five-character long part of the title bar displayed left of the CaptionRight, with a separation symbol.

sCaptionRight in    The caption of the dialog.

sHelpText      in    This text is shown, when the help button SHIFT-F1 is pressed and the help functionality of the theodolite is enabled.

Return-Codes

RC_OK    Successful termination.

See Also

MMI_DeleteDialog, MMI_CreateGraphDialog, GSI_CreateMDlg, MMI_PrintVal, MMI_PrintStr, MMI_PrintTok, MMI_PrintInt, MMI_InputVal, MMI_InputStr, MMI_InputInt, MMI_InputList
Example  The example uses the MMI_CreateTextDialog routine to create and display a text dialog.

Define a help text containing the 'inverse written word "Help"

CONST Helptext = MMI_INVERSE_ON + "Help" + MMI_INVERSE_OFF + " Test"

MMI_CreateTextDialog(5, "TEXT", "DIALOG CAPTION", Helptext)

6.1.12 MMI_CreateGraphDialog

Description  Create and show a graphics dialog.

Declaration  MMI_CreateGraphDialog(
  BYVAL sCaptionLeft AS _Token,
  BYVAL sCaptionRight AS _Token,
  BYVAL sHelptext AS _Token )

Remarks  The routine creates and shows a graphics dialog filled with the left part of the title bar sCaptionLeft, the caption sCaptionRight and the help text sHelpText for later use of MMI graphics functions. The size of the field is the whole dialog display area = 232 x 48 pixels. Only one graphics dialog can exist at the same time. If CreateGraphDialog is called while already a graphics dialog exists, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

Note  Only a text dialog or a measurement dialog is valid at a time. They cannot be defined at the same time. A graphic dialog overrides a text or measurement dialog but does not delete the definition of it.
Parameters

- **sCaptionLeft** in
  The maximal five-character long part of the title bar displayed left of the sCaptionRight, with a separation symbol

- **sCaptionRight** in
  The caption of the dialog.

- **sHelpText** in
  This text is shown, when the help button Shift-F1 is pressed and the help functionality of the theodolite is enabled.

Return-Codes

- **RC_OK**
  Successful termination.

See Also

- MMI_DeleteDialog, MMI_CreateTextDialog, GSI_CreateMDlg, MMI Graphic Functions

Example

The example uses the MMI_CreateGraphDialog routine to create and display a graphic dialog field.

```geo
MMI_CreateGraphDialog( "GRAPH", 
                      "DIALOG CAPTION",
                      "This is a help text")
```

6.1.13 MMI_DeleteDialog

Description

Deletes a dialog.

Declaration

```geo
MMI_DeleteDialog()
```

Remarks

The routine deletes the currently active dialog. It makes no distinction between graphic, measure and text dialog. By deleting the dialog all user defined buttons added with MMI_AddButton are deleted as well.

Return-Codes

- **RC_OK**
  Successful termination.

- **BAS_NO_DLG_EXIST**
  No dialog exists for this operation.

See Also

- MMI_CreateTextDialog, MMI_CreateGraphDialog, GSI_CreateMDlg
Example

The example uses the MMI_DeleteDialog routine to delete a text, measure or graphic dialog.

MMI_DeleteDialog()
Example

The example uses the MMI_CheckButton routine to wait until a (valid) key was pressed.

```
DIM lKeyPressed AS Logical

DO
    MMI_CheckButton( lKeyPressed )
LOOP UNTIL lKeyPressed

'do something ..
```

### 6.1.15 MMI_GetButton

**Description**
Get the button identifier of the pressed button.

**Declaration**

```
MMI_GetButton( iButtonId AS Integer, 
               BYVAL lAllKeys AS Logical )
```

**Remarks**
Waits until a valid key is pressed and returns the button Identifier iButtonId of the pressed button.

If lAllKeys = FALSE, the keys ESC, ENTER, ON/OFF or any assigned button (added with MMI_AddButton) terminates this function and the iButtonId of the pressed button is returned. If lAllKeys = TRUE, additional keys i.e. the cursor keys terminates this routine too. For details see table below.

**Parameters**

- **iButtonId** Out
  The identifier of the pressed button. For values of iButtonId see the table below.

- **lAllKeys** In
  Determines which keys exit the routine. If lAllKeys = TRUE any valid pressed key exit the routine, otherwise only normal ones.

**Note**
This function relates to the currently active dialog.
### Button pressed | iButtonId returned | 1AllKeys = \* | 1AllKeys = \*
---|---|---|---
assigned (using MMI_AddButton) | MMI_F1_KEY, MMI_F6_KEY, MMI_SHF2_KEY, MMI_SHF6_KEY | MMI_F1_KEY, MMI_F6_KEY, MMI_SHF2_KEY, MMI_SHF6_KEY | TRUE | FALSE
assigned "CODE" | MMI_CODE_KEY | MMI_CODE_KEY | no return
unassigned "F1".."F6", "SHIFT-F2".."SHIFT-F6" | MMI_UNASS_KEY | MMI_UNASS_KEY | no return
"ENTER" within dialog, focus on a field | MMI_UNASS_KEY | MMI_UNASS_KEY | no return
"ENTER" within dialog, no focus | MMI_UNASS_KEY | MMI_UNASS_KEY | no return
"ENTER" after editing | MMI_EDITENTER_KEY | MMI_EDITENTER_KEY | no return
"ESC" within dialog | MMI_ESC_KEY | MMI_ESC_KEY | no return
"ESC" after editing | MMI_EDITESC_KEY | no return
"SHIFT" | MMI_UNASS_KEY | no return
"0".."9", focus on spin/list-field | MMI_UNASS_KEY | no return
"0".."9", no focus | MMI_NUM0_KEY, MMI_NUM9_KEY | MMI_NUM0_KEY, MMI_NUM9_KEY | no return
"CE" | MMI_UNASS_KEY | no return
cursor keys | MMI_UP_KEY, MMI_DOWN_KEY, MMI_RIGHT_KEY, MMI_LEFT_KEY | no return

### Return-Codes
- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

### See Also
- MMI_AddButton, MMI_CheckButton
Example

The example uses the `MMI_GetButton` routine to react to a pressed button. To make a function key valid for `MMI_GetButton` it must be added to the dialog (with `MMI_AddButton`).

```basic
DIM iActionButton AS Integer
DIM iPressedButton AS Integer

iActionButton = MMI_F2_KEY
MMI_GetButton ( iPressedButton, TRUE )
IF iPressedButton = iActionButton THEN
    'any actions
END IF
```

### 6.1.16 MMI_AddButton

**Description**
Add a button to a dialog.

**Declaration**

```basic
MMI_AddButton( BYVAL iButtonId AS Integer,
               BYVAL sCaption AS _Token )
```

**Remarks**

The routine `MMI_AddButton` adds the button with the Identifier `iButtonId` to the actual dialog and places the text `sCaption` onto the button. These added buttons are valid for the routines `MMI_CheckButton` and `MMI_GetButton` and the input routines (`MMI_InputStr`, `MMI_InputVal`, `MMI_InputInt` and `MMI_InputList`) which means the according button identifier can be returned from these routines.

**Note**

Either a text dialog or a measurement dialog can be defined at a time. Additionally a graphics dialog can override one of these above. Then the functionality applies to the graphics dialog.

The added buttons can be deleted with the routine `MMI_DeleteButton` while the dialog exists. Closing the dialog with `MMI_DeleteDialog` deletes all buttons attached to this dialog.
### Parameters

**iButtonId in** Identifier of the button to be added. See for the values that can be used for the iButtonId under the routine description MMI_GetButton. Only MMI_F1_Key..MMI_F5_KEY, MMI_SHF2_KEY..MMI_SHF6_KEY and MMI_CODE_KEY are available for the AddButton routine.

**sCaption in** The text placed onto the button, left alignment (max. 5 characters).

### Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.
- **MMI_BUTTON_ID_EXISTS** This button has been defined already.

### See Also

- MMI_GetButton
- MMI_CheckButton
- MMI_DeleteButton

### Example

The example uses the MMI_AddButton routine to add the F2-KEY with the caption "EXIT" to the dialog.

```geo
MMI_AddButton( MMI_F2_KEY, "EXIT" )
```

### 6.1.17 MMI_DeleteButton

**Description** Delete a button from a dialog.

**Declaration**

```
MMI_DeleteButton( iButtonId AS Integer )
```

**Remarks**

The routine MMI_DeleteButton deletes the button with the Identifier iButtonId from the actual dialog. Only a button that was added with MMI_AddButton can be deleted. Closing the dialog with MMI_DeleteDialog deletes all buttons attached to this dialog.
### MMI_DeleteButton

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iButtonId</td>
<td>Identifier of the button to be deleted. See for the values that can be used for iButtonId under the routine description MMI_GetButton.</td>
</tr>
</tbody>
</table>

**Return-Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
<tr>
<td>MMI_ILLEGAL_BUTTON_ID</td>
<td>This button has not been defined by MMI_AddButton.</td>
</tr>
</tbody>
</table>

**See Also**

MMI_AddButton

**Example**

The example uses the MMI_DeleteButton routine to delete the F2-KEY from the dialog.

```c
MMI_DeleteButton( MMI_F2_KEY )
```

### MMI_PrintStr

**Description**

Print a string on a text dialog.

**Declaration**

```c
MMI_PrintStr( BYVAL iColumn AS Integer, 
              BYVAL iLine AS Integer, 
              BYVAL sText AS String30, 
              BYVAL lValid AS Logical )
```

**Remarks**

The text string sText is placed on position iColumn and iLine on the text dialog. If lValid is not TRUE, then the symbols for invalid values are displayed. Too long text strings are truncated, illegal co-ordinates are adjusted.

**Note**

A text dialog must already exist. Only display length number of character will be displayed, hence 29.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn</td>
<td>The horizontal position (0..28)</td>
</tr>
</tbody>
</table>
iLine in  The vertical position (0..number of lines defined with MMI_CreateTextDialog)

sText in  The text string to display

lValid in  Determines if the value should be shown as valid. If lValid = TRUE the value sText is displayed, otherwise the symbols for invalid values are displayed.

Return-Codes

RC_OK  Successful termination.

BAS_NO_DLG_EXIST  No dialog exists for this operation.

See Also  MMI_InputStr

Example  The example uses the MMI_PrintStr routine to print the text string „Hello World“ in the first line on row 2 of the actual text dialog.

MMI_PrintStr( 2, 0, "Hello World", TRUE )

6.1.19  MMI_PrintTok

Description  Print a string on a text dialog.

Declaration  MMI_PrintTok( BYVAL iColumn AS Integer,
                                 BYVAL iLine AS Integer,
                                 BYVAL sText AS _Token )

Remarks  The text token sText is placed on position iColumn and iLine on the text dialog. Too long text strings are truncated, illegal co-ordinates are adjusted. This routine may be used instead of MMI_PrintStr to support internationalisation of multiple language applications.

Note  A text dialog must already exist.

Parameters  

iColumn in  The horizontal position (0..28)

iLine in  The vertical position (0..number of lines defined with MMI_CreateTextDialog)
sText in The text string to display

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
<tr>
<td>TXT_UNDEF_TOKEN</td>
<td>The given token could not be found in the database. Most probably an old</td>
</tr>
<tr>
<td></td>
<td>version is loaded either on TPS or simulator.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>No text token database is loaded with the currently set language.</td>
</tr>
</tbody>
</table>

See Also

MMI_PrintStr

Example

The example uses the MMI_PrintTok routine to print the text string „Hello World“ in the first line on row 2 of the actual text dialog:

```basic
MMI_PrintTok( 2, 0, "Hello World" )
```

6.1.20 MMI_PrintVal

Description

Print a value on a text dialog.

Declaration

```basic
MMI_PrintVal( BYVAL iColumn AS Integer, 
               BYVAL iLine AS Integer, 
               BYVAL iLen AS Integer, 
               BYVAL iDecimals AS Integer, 
               BYVAL dVal AS Double, 
               BYVAL lValid AS Logical, 
               BYVAL iMode AS Integer )
```

Remarks

This routine can be used to display double values (or values with equal type, e.g. dimension). If lValid = TRUE the value dVal is placed on position iColumn and iLine on the text dialog, otherwise the symbols for invalid values "-----" are displayed. Too long value strings are truncated, illegal co-ordinates are adjusted. If iMode = MMI_DIM_ON, a dimension field is automatically displayed when the type of dVal has units.
If the dVal cannot be displayed in iLen characters, then "xxx" will be displayed instead.

| Note | A text dialog must already exist. |

### Parameters

- **iColumn in** The horizontal position (0..28).
- **iLine in** The vertical position (0..number of lines defined with CreateTextDialog).
- **iLen in** The length of the value consisting of a sign, the characters before and after the comma and the comma itself. The dimension field is not included.
- **iDecimals in** The number of decimals. If iDecimals = -1 then the number of decimals set by the system is taken.
- **dVal in** The value to display. Use this routine to display double (and equal to double) values with the correct units. For integer values a separate routine (MMI_PrintInt) exists.
- **lValid in** Determines if the value should be shown as valid. If lValid = TRUE the value dVal is displayed, otherwise the symbols for invalid values are displayed.
- **iMode in** Determines the display of the dimension. If Mode = MMI_DIM_ON a dimension field is automatically displayed when the type dVal has units. Otherwise use MMI_DEFAULT_MODE.

### Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

### See Also

MMI_PrintInt, MMI_InputVal

### Example

The example uses the MMI_PrintVal routine to print the value of TestVal as distance (with corresponding dimension) in the first line on row 2 of the currently open text dialog.
DIM TestVal AS Distance
TestVal = 287.47

MMI_PrintVal( 2, 0, 10, 2, TestVal, TRUE, 
               MMI_DIM_ON )

6.1.21 MMI_PrintInt

Description
Print an integer value on a text dialog.

Declaration
MMI_PrintInt( BYVAL iColumn AS Integer, 
               BYVAL iLine AS Integer, 
               BYVAL iLen AS Integer, 
               BYVAL iVal AS Integer, 
               BYVAL lValid AS Logical )

Remarks
This routine can be used to display integer values. Too long value 
strings are truncated, illegal co-ordinates are adjusted. If 
lValid = TRUE the value iVal is placed on position 
iColumn and iLine on the text dialog, otherwise the symbols 
for invalid values are displayed. 
If the iVal can not be displayed in iLen characters, then "xxx" 
will be displayed instead.

Note   A text dialog must already exist.

Parameters
iColumn in   The horizontal position (0..28).
iLine in     The vertical position (0..number of lines 
defined with MMI_CreateTextDialog).
iLen in      The length of the value plus the sign.
iVal in      The value to display. Use this routine to 
display integer values. For double values a 
separate routine (MMI_PrintVal) exists.
lValid in     Determines if the value should be shown as 
valid. If lValid = TRUE the value iVal is 
displayed, otherwise the symbols for invalid 
values are displayed.
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

See Also

- MMI_PrintVal
- MMI_InputInt

Example

The example uses the MMI_PrintInt routine to print the value of TestVal in the first line on row 2 of the currently open text dialog.

```basic
DIM TestVal AS Integer
TestVal = 1000
MMI_PrintInt( 2, 0, 5, TestVal, TRUE )
```

6.1.22 MMI_InputStr

Description

Get a string input in a text dialog.

Declaration

```basic
MMI_InputStr( BYVAL iColumn AS Integer, 
              BYVAL iLine AS Integer, 
              BYVAL iLen AS Integer, 
              BYVAL iMode AS Integer, 
              sText AS String30, 
              lValid AS Logical, 
              iButtonId AS Integer )
```

Remarks

If lValid = TRUE the text string sText is placed on position iColumn and iLine on the text dialog, otherwise the symbols for invalid values are displayed. Illegal co-ordinates are adjusted. If the length of the string exceeds the given length iLen the string is truncated at position iLen. After the edit process the string is returned and the text is placed right aligned on the display. If the length iLen <= 0 or no part of the field is in the dialog area the Text is not edited and the routine exits.

The string can be edited by pressing αEDIT or a numerical key. If iMode = MMI_DEFAULT_MODE the keys ESC, ENTER,
ON/OFF or any user defined button (added with MMI_AddButton) terminates the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputStr too. For details see MMI_GetButton.

### Note
A text dialog must already exist.

### Parameters

- **iColumn** *in*
  - The horizontal position (0..28).

- **iLine** *in*
  - The vertical position (0..number of lines defined with MMI_CreateTextDialog).

- **iLen** *in*
  - The length of the input field.

- **iMode** *in*
  - Defines the editing mode.
    - MMI_DEFAULT_MODE defines normal editing
    - MMI_SPECIALKEYS_ON allows editing with full cursor control

- **sText** *inout*
  - The text string to edit.

- **lValid** *inout*
  - Determines if the value should be shown as valid. If lValid=TRUE the string sText is displayed, otherwise the symbols for invalid values are displayed.

- **iButtonId** *out*
  - The identifier of the pressed valid button to exit the edit process.

### Return-Codes

- **RC_OK**
  - Successful termination.

- **BAS_NO_DLG_EXIST**
  - No dialog exists for this operation.

### See Also

- MMI_PrintStr
Example

The example uses the `MMI_InputStr` routine to get the text string `sInputString` in the first line on row 2 of the actual text dialog.

```vbnet
DIM sInputString AS String30
DIM iButton AS Integer
DIM lValid AS Logical

sInputString = "The input text"
lValid = TRUE
MMI_InputStr( 2, 0, 20, MMI_DEFAULT_MODE, sInputString, lValid, iButton )
```

6.1.23 MMI_InputVal

**Description**
Get a numerical input for double values in a text dialog.

**Declaration**

```vbnet
MMI_InputVal( BYVAL iColumn AS Integer, 
               BYVAL iLine AS Integer, 
               BYVAL iLen AS Integer, 
               BYVAL iDecimals AS Integer, 
               BYVAL dMin AS Double, 
               BYVAL dMax AS Double, 
               BYVAL iMode AS Integer, 
               dVal AS Double, 
               lValid AS Logical, 
               iButtonId AS Integer )
```

**Remarks**

If `lValid = TRUE` then the value `dVal` is placed on position `iColumn` and `iLine` on the text dialog, otherwise the symbols for invalid values are displayed. Illegal co-ordinates are adjusted. If `iMode = MMI_DIM_ON`, a dimension field is automatically displayed when the type of `dVal` has units. If the length `iLen` <= 0 or no part of the field is in the dialog area the value is not edited and the routine exits.

The value within the bounds `dMin` and `dMax` can be edited by pressing `EDIT` or the numerical block keys. If `iMode = MMI_DEFAULT_MODE` the keys `ESC`, `ENTER`, `ON/OFF` or any user defined button (added with `MMI_AddButton`) terminates.
the edit process and the iButtonId of the pressed button is returned. If iMode = MMI_SPECIALKEYS_ON additional keys i.e. the cursor keys terminates MMI_InputVal too. For details see MMI_GetButton.

| Note | A text dialog must already exist. |

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iColumn in</td>
<td>The horizontal position (0..28).</td>
</tr>
<tr>
<td>iLine in</td>
<td>The vertical position (0..number of lines defined with MMI_CreateTextDialog).</td>
</tr>
<tr>
<td>iLen in</td>
<td>The length of the value inclusive decimals, sign and the comma, exclusive the dimension field</td>
</tr>
<tr>
<td>iDecimals in</td>
<td>The number of decimals. If iDecimals = -1 the number of decimals set by the system is taken.</td>
</tr>
<tr>
<td>dMin in</td>
<td>The lower and upper bounds.</td>
</tr>
<tr>
<td>dMax</td>
<td></td>
</tr>
<tr>
<td>iMode in</td>
<td>Defines the editing mode.</td>
</tr>
<tr>
<td>dVal inout</td>
<td>The value to edit. Use this routine to edit double (and equal to double) values. For integer values a separate routine (MMI_InputInt) exists.</td>
</tr>
</tbody>
</table>

- **MMI_DEFAULT_MODE** defines normal editing
- **MMI_SPECIALKEYS_ON** allows editing with full cursor control
- **MMI_DIM_ON** shows a dimension field if dVal has units.

Modes can be added, i.e.

- **MMI_SPECIALKEYS_ON** + **MMI_DIM_ON**
lValid inout Determines if the value should be shown as valid. If lValid=TRUE the value dVal is displayed, otherwise the symbols for invalid values are displayed.

iButtonId out The identifier of the pressed valid button to exit the edit process.

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No dialog exists for this operation.</td>
</tr>
</tbody>
</table>

See Also

MMI_InputInt
MMI_PrintVal

Example

See example file "cursor.gbs" too.

The example uses the MMI_InputVal routine to get the distance of TestVal with default decimal places. Input field is placed in the second line on row 2 of the actual text dialog. The entered values must lie in the range 0..1000.

```basic
CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM TestVal AS Distance
DIM iButton AS Integer
DIM lValid AS Logical

lValid = FALSE

MMI_InputVal( 2, 1, 8, -1, 0, 1000, MODE, TestVal, lValid, iButton )
```
### 6.1.24 MMI_InputInt

**Description**  
Get an integer input value in a text dialog.

**Declaration**  
```basic  
MMI_InputInt(  
    BYVAL iColumn AS Integer,  
    BYVAL iLine AS Integer,  
    BYVAL iLen AS Integer,  
    BYVAL iMin AS Integer,  
    BYVAL iMax AS Integer,  
    BYVAL iMode AS Integer,  
    iVal AS Integer,  
    iValid AS Logical,  
    iButtonId AS Integer  
)  
```

**Remarks**  
If `iValid = TRUE` then the integer value `iVal` is placed on position `iColumn` and `iLine` on the text dialog. Illegal coordinates are adjusted. If the length `iLen ≤ 0` or no part of the field is in the dialog area the value is not edited and the routine exits.

The integer value within the bounds `iMin` and `iMax` can be edited by pressing EDIT or the numerical block keys. If `iMode = MMI_DEFAULT_MODE` the keys ESC, ENTER, ON/OFF or any user defined button (added with `MMI_AddButton`) terminates the edit process and the `iButtonId` of the pressed button is returned. If `iMode = MMI_SPECIALKEYS_ON` additional keys i.e. the cursor keys terminates `MMI_InputInt` too.

**Note**  
A text dialog must already exist.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iColumn</code></td>
<td>The horizontal position (0..28).</td>
</tr>
<tr>
<td><code>iLine</code></td>
<td>The vertical position (0..number of lines defined with <code>MMI_CreateTextDialog</code>).</td>
</tr>
<tr>
<td><code>iLen</code></td>
<td>The length of the value plus the sign.</td>
</tr>
<tr>
<td><code>iMin</code></td>
<td>The lower and upper bounds.</td>
</tr>
</tbody>
</table>
Defines the editing mode.

- **MMI_DEFAULT_MODE** defines normal editing
- **MMI_SPECIALKEYS_ON** allows editing with full cursor control

**iVal** inout

The value to display. Use this routine to edit integer values. For double values a separate routine (**MMI_InputVal**) exists.

**lValid** inout

Determines if the value should be shown as valid. If **lValid**=TRUE the value **iVal** is displayed, otherwise the symbols for invalid values are displayed.

**iButtonId** out

The identifier of the pressed valid button to exit the edit process.

**Return-Codes**

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No dialog exists for this operation.

**See Also**

- **MMI_PrintInt**, **MMI_InputVal**

**Example**

See example file "cursor.gbs" too.

The example uses the **MMI_InputInt** routine to get the value of **iTestVal** in the second line on row 2 of the actual text dialog. The entered values must lie in the range 0..1000.

```basic
CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM iTestVal AS Integer
DIM iButton AS Integer
DIM lValid AS Logical

lValid = FALSE
MMI_InputInt( 2,1,5,0,1000, MODE,iTestVal,lValid,iButton )
```
6.1.25 MMI_InputList

**Description**
Shows a list field in a text dialog.

**Declaration**
```basic
MMI_InputList( ByVal iColumn As Integer,
                ByVal iLine As Integer,
                ByVal iLen As Integer,
                ByVal iElements As Integer,
                ByVal iMode As Integer,
                List As ListArray,
                iIndex As Integer,
                lValid As Logical,
                iButtonId As Integer )
```

**Remarks**
If `lValid = TRUE` then a list field is placed on position `iColumn` and `iLine` on the text dialog. Too long list elements are truncated, illegal co-ordinates are adjusted. The `ListArray` is an array of `String30` with `LIST_ARRAY_MAX_ELEMENT` Elements. Only the first `iElements` are displayed. The value of `iIndex` defines which element is shown first.

The list can be edited by pressing F6 (LIST). With the cursor keys UP and DOWN a field element can be selected. If the list elements are numbered (begins with a number), then the elements can be selected directly by pressing numerical buttons. If `iMode = MMI_DEFAULT_MODE` the keys ESC, ENTER, ON/OFF or any user defined button (added with `MMI_AddButton`) terminates the edit process and the `iButtonId` of the pressed button is returned. If `iMode = MMI_SPECIALKEYS_ON` additional keys i.e. the cursor keys terminates `MMI_InputList` too.

**Parameters**
- **iColumn** in
  The horizontal position (0..28).
- **iLine** in
  The vertical position (0..number of lines defined with `MMI_CreateTextDialog`).
- **iLen** in
  The displayed length of the list elements.

---

Note
A text dialog must already exist.
### GeoBASIC Reference Manual

#### 6. System Functions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iElements</td>
<td>The number of list elements. The maximum number is limited to ( \text{LIST_ARRAY_MAX_ELEMENT} ).</td>
</tr>
<tr>
<td>iMode</td>
<td>Defines the editing mode. ( \text{MMI_DEFAULT_MODE} ) defines normal editing. ( \text{MMI_SPECIALKEYS_ON} ) allows editing with full cursor control.</td>
</tr>
<tr>
<td>List</td>
<td>The array of the list elements.</td>
</tr>
<tr>
<td>iIndex</td>
<td>Index (number of the line) of the first shown and selected field respectively. Possible value for ( \text{iIndex} ) are in the range of 1 up to ( \text{Elements} ).</td>
</tr>
<tr>
<td>lValid</td>
<td>Determines if the value should be shown as valid. If ( \text{lValid}=\text{TRUE} ) the value is displayed, otherwise the symbols for invalid values are displayed.</td>
</tr>
<tr>
<td>iButtonId</td>
<td>The identifier of the pressed valid button to exit the list process.</td>
</tr>
</tbody>
</table>

**Return-Codes**

- RC_OK: Successful termination.
- BAS_NO_DLG_EXIST: No dialog exists for this operation.

**Example**

See example file "cursor.gbs" too.

The example uses the \( \text{MMI\_InputList} \) routine to get the value of the selected list element (the selected line) of a list field displayed in the second line on row 2 of the actual text dialog. The first displayed line is the line with the number Index.
CONST MODE = MMI_DEFAULT_MODE 'define editmode

DIM iLen AS Integer
DIM iElements AS Integer
DIM List AS ListArray
DIM iIndex AS Integer
DIM iButton AS Integer
DIM lValid AS Logical

'initialize the variables
iLen = 10 'displayed length of the list
iElements = 7 'number of available fields
iIndex = 3 'number of the first shown list element
lValid = TRUE

List(1) = "1 Line No.: 1"
List(2) = "2 Line No.: 2"
List(3) = "3 Line No.: 3"
List(4) = "4 Line No.: 4"
List(5) = "5 Line No.: 5"
List(6) = "6 Line No.: 6"
List(7) = "7 Line No.: 7"

InputList( 5, 1, iLen, iElements, MODE,
            List, iIndex, lValid, iButton )

### 6.1.26 MMI_FormatVal

**Description** Convert a value to a string and use TPS system formatting rules.

**Declaration**

```basic
MMI_FormatVal( BYVAL iType AS Integer,
                BYVAL iLen AS Integer,
                BYVAL iDecimals AS Integer,
                BYVAL dVal AS Double,
                BYVAL lValid AS Logical,
                BYVAL iMode AS Integer,
                sValStr AS String30 )
```

**Remarks** If lValid = TRUE then this routine converts a double value (or values with equal type, e.g. dimension) to a text string, otherwise the symbols for invalid values are returned. The returned string
sValStr contains the value string in the same kind as it would be displayed on the Theodolite: the value is placed right aligned with the number iDecimals of decimals. If \texttt{iMode = MMI\_DIM\_ON}, a dimension field is appended to the output string when the type \texttt{iType} allows it. If the \texttt{dVal} can not be displayed in \texttt{iLen} characters, then "xxx" will be returned instead.

This routine is useful, if numeric values should be written on files (see chapter file handling for further information).

### Parameters

- \texttt{iType in} The type of the numerical field. The type defines if a dimension field is available. Following values for the type can be used:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_FFORMAT_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>MMI_FFORMAT_DISTANCE</td>
<td>distance</td>
</tr>
<tr>
<td>MMI_FFORMAT_SUBDISTANCE</td>
<td>sub-distance [mm]</td>
</tr>
<tr>
<td>MMI_FFORMAT_ANGLE</td>
<td>angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_VANGLE</td>
<td>vertical angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_HZANGLE</td>
<td>horizontal angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_TEMPERATURE</td>
<td>temperature</td>
</tr>
<tr>
<td>MMI_FFORMAT_TIME</td>
<td>time 12h/24h-format</td>
</tr>
<tr>
<td>MMI_FFORMAT_DATE</td>
<td>date</td>
</tr>
<tr>
<td>MMI_FFORMAT_DATE_TIME</td>
<td>date/time</td>
</tr>
</tbody>
</table>

- \texttt{iLen in} The length of the value consisting of a sign, the characters before and after the comma and the comma itself. The dimension field is not included.

- \texttt{iDecimals in} The number of decimals. If \texttt{iDecimals = -1} the number of decimals set by the system is taken.
dVal in  The value to convert. Use this routine to convert double (and equal to double) values.

iMode in  If iMode = MMI_DIM_ON a dimension string is automatically added to sValStr when the type dVal has units. Otherwise use MMI_DEFAULT_MODE.

sValStr out  sValStr contains the string representation of the value dVal.

Return-Codes

RC_OK    Successful termination.

RC_IVRESULT    The result is not valid due to an illegal input value.

See Also

sFormatVal

Example

The example uses the MMI_FormatVal routine to convert the value dTestVal as distance (with corresponding dimension).

```
DIM dTestVal AS Distance
DIM sVString AS String30

dTestVal = 287.47

MMI_FormatVal( MMI_FFORMAT_DISTANCE, 10, -1, dTestVal, TRUE, MMI_DIM_ON, sVString )
```

6.1.27  MMI_WriteMsg

Description

Output to a message window.

Declaration

```
MMI_WriteMsg( BYVAL sText AS _Token, 
    BYVAL sCaption AS _Token, 
    BYVAL iMsgType AS Integer, 
    iRetKey AS Integer )
```

Remarks

The function opens a message window on the display, which shows the text specified by sText. Lines that are too long to fit into the window are split automatically.
sText may contain a carriage return (character code 10) which breaks a line explicitly. The predefined constants MMI_INVERSE_ON and MMI_INVERSE_OFF can be used for inverse text.

Text lines, that exceed the size of the window, are not displayed. A title text, which will be printed on the first line of the message box, can be set with sCaption, which may not be longer than one line and contain neither font attributes nor type information.

**Parameters**

- **sText** in
  Text-token to be displayed on the window (on the Theodolite).

- **sCaption** in
  Text-token that will be displayed as title of the window.

- **iMsgType** in
  Defines the type of the message window to be displayed, with the corresponding text on the buttons; possible types:
  - MMI_MB_OK
  - MMI_MB_ABORT
  - MMI_MB_OK_ABORT
  - MMI_MB_ABORT_RETRY_CONT
  - MMI_MB_YES_NO_ABORT
  - MMI_MB_YES_NO
  - MMI_MB_RETRY_ABORT
  - MMI_MB_ABORT_CONT
  - MMI_MB_ABORT_RETRY_IGNORE
  - MMI_MB_ABORT_IGNORE

- **iRetKey** out
  Returns the button pressed, i.e.
  - iRetKey:
    - MMI_MB_RET_OK
    - MMI_MB_RET_ABORT
    - MMI_MB_RET_RETRY
    - MMI_MB_RET_CONT
    - MMI_MB_RET_YES
    - MMI_MB_RET_NO
    - MMI_MB_RET_IGNORE

Return-Codes

- **RC_OK**  
Successful termination.

- **BAS_NO_DLG_EXIST**  
No dialog exists for this operation.

Example

The example uses the `MMI_WriteMsg` routine to display a message box with the title text "Warning" and the text "timeout" and shows the buttons "Retry", "Abort" returning the button-id in iRetKey.

```plaintext
MMI_WriteMsg( "Warning", "timeout", MMIMB_RETRY_ABORT, iMBRetKey )
```

### 6.1.28 MMI_WriteMsgStr

**Description**  
Output to a message window.

**Declaration**

```plaintext
MMI_WriteMsgStr( BYVAL sText AS String255, 
                  BYVAL sCaption AS _Token, 
                  BYVAL iMsgType AS Integer, 
                  iRetKey AS Integer )
```

**Remarks**

The function opens a message window on the display, which shows the text specified by `sText`. Lines, which are too long to fit into the window, are split automatically. `sText` may contain a carriage return (character code 10) which breaks a line explicitly. The predefined constants `MMI_INVERSE_ON` and `MMI_INVERSE_OFF` can be used for inverse text. Text lines, that exceed the size of the window, are not displayed. A title text, which will be printed on the first line of the message box, can be set with `sCaption`, which may not be longer than one line and contain neither font attributes nor type information.

**Note**  
This routine is different to `MMI_WriteMsg` in such a way that `sText` may be computed. But, of course, `sText` will not be entered into the text token data base.

**Parameters**

- **sText**  
Text string to be displayed in a message box.
sCaption in
Text-token that will be displayed as title of the window.

iMsgType in
Defines the type of the message window to be displayed, with the corresponding text on the buttons; possible types:
- MMI_MB_OK
- MMI_MB_ABORT
- MMI_MB_OK_ABORT
- MMI_MB_ABORT_RETRY_CONT
- MMI_MB_YES_NO_ABORT
- MMI_MB_YES_NO
- MMI_MB_RETRY_ABORT
- MMI_MB_ABORT_CONT
- MMI_MB_ABORT_RETRY_IGNORE
- MMI_MB_ABORT_IGNORE

iRetKey out
Returns the button pressed, i.e.
iRetKey:
- MMI_MB_RET_OK
- MMI_MB_RET_ABORT
- MMI_MB_RET_RETRY
- MMI_MB_RET_CONT
- MMI_MB_RET_YES
- MMI_MB_RET_NO
- MMI_MB_RET_IGNORE

Return-Codes
- RC_OK Successful termination.
- BAS_NO_DLG_EXIST No dialog exists for this operation.

See Also
- MMI_WriteMsg
Example  
The example uses the `MMI_WriteMsgStr` routine to display a message box with the title text "Warning" and the text:

```
MessageStr
time out in 10 seconds
```
and shows the buttons "Retry", "Abort" returning the button-id in `iRetKey`.

```plaintext
CONST iTimeOut AS Integer = 10
DIM sMessage As String255
DIM iMBRetKey AS Integer

sMessage = "MessageStr\d010time out in " + Str$(iTimeOut) + "seconds"
MMI_WriteMsgStr( "Warning", sMessage, MMI_MB_RETRY_ABORT,iMBRetKey )
```

6.1.29  **MMI_DrawLine**

**Description**  
Draw a line.

**Declaration**  
`MMI_DrawLine( BYVAL iX1 AS Integer, BYVAL iY1 AS Integer, BYVAL iX2 AS Integer, BYVAL iY2 AS Integer, BYVAL iPen AS Integer )`

**Remarks**  
The function draws a line within the graphic field using the line-style `iPen`.

```
Note   A graphics dialog has to be set up before.
```

**Parameters**  
- `iX1` in  x-co-ordinate of the beginning of the line [pixel]
- `iY1` in  y-co-ordinate of the beginning of the line [pixel]
- `iX2` in  x-co-ordinate of the end of the line [pixel]
- `iY2` in  y-co-ordinate of the end of the line [pixel]
iPen in Line-style; possible values:
- MMI_PEN_WHITE
- MMI_PEN_BLACK
- MMI_PEN_DASHED

Return-Codes
- RC_OK: Successful termination.
- BAS_NO_DLG_EXIST: No graphics dialog exists for this operation.

See Also
- MMI_CreateGraphDialog, MMI_DrawRect,
- MMI_DrawCircle, MMI_DrawText

Example
The example uses the MMI_DrawLine routine to draw a line with the specified attributes.

```plaintext
MMI_DrawLine( 10, 10, 100, 50, MMI_PEN_BLACK )
```

6.1.30 MMI_DrawRect

Description
Draw a rectangle.

Declaration
```plaintext
MMI_DrawRect( BYVAL iX1 AS Integer,
               BYVAL iY1 AS Integer,
               BYVAL iX2 AS Integer,
               BYVAL iY2 AS Integer,
               BYVAL iBrush AS Integer,
               BYVAL iPen AS Integer )
```

Remarks
This function draws a rectangle in the graphic field using the fill-style iBrush and the line-style iPen.

Note
A graphics dialog has to be set up before.
Parameters

- **iX1 in** x-co-ordinate at the upper left-hand corner of the rectangle [pixel]
- **iY1 in** y-co-ordinate at the upper left-hand corner of the rectangle [pixel]
- **iX2 in** x-co-ordinate at the bottom right-hand corner of the rectangle [pixel]
- **iY2 in** y-co-ordinate at the bottom right-hand corner of the rectangle [pixel]
- **iBrush in** Fill-style for the rectangle; possible values:
  - MMI_BRUSH_WHITE
  - MMI_BRUSH_BLACK
  - MMI_NO_BRUSH
- **iPen in** Line-style:
  - MMI_PEN_WHITE
  - MMI_PEN_BLACK
  - MMI_PEN_DASHED

Return-Codes

- **RC_OK** Successful termination.
- **BAS_NO_DLG_EXIST** No graphics dialog exists for this operation.

See Also

- MMI_CreateGraphDialog, MMI_DrawLine, MMI_DrawCircle, MMI_DrawText

Example

The example uses the **MMI_DrawRect** routine to draw a rectangle with the specified attributes.

```
MMI_DrawRect( 10, 10, 100, 50, MMI_NO_BRUSH, MMI_PEN_BLACK )
```
6.1.31 MMI_DrawCircle

**Description**  
Draw a circle / ellipse.

**Declaration**  
```plaintext
MMI_DrawCircle( BYVAL iX AS Integer,
                 BYVAL iY AS Integer,
                 BYVAL iRx AS Integer,
                 BYVAL iRy AS Integer,
                 BYVAL iBrush AS Integer,
                 BYVAL iPen AS Integer )
```

**Remarks**  
This function draws a circle in the graphic field, using the radius `iRx`, the fill-style `iBrush`, and the line-style `iPen`, as long as `iRx = iRy`. Otherwise, an ellipse is drawn, where `iRx` and `iRy` are the lengths of the perpendicular radii.

**Parameters**

- `iX` in x-coordinate at the centre of the circle/ellipse [pixel]
- `iY` in y-coordinate at the centre of the circle/ellipse [pixel]
- `iRx` in Radius of the circle, horizontal radius [pixel]
- `iRy` in Radius of the circle, vertical radius [pixel]
- `iBrush` in Fill-style for the rectangle; possible values:  
  - MMI_BRUSH_WHITE
  - MMI_BRUSH_BLACK
  - MMI_NO_BRUSH
- `iPen` in Line-style; possible values:  
  - MMI_PEN_WHITE
  - MMI_PEN_BLACK
  - MMI_PEN_DASHED

**Note**  
A graphics dialog has to be set up before.
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No graphics dialog exists for this operation.</td>
</tr>
</tbody>
</table>

See Also

MMI_CreateGraphDialog, MMI_DrawLine, MMI_DrawRect, MMI_DrawText

Example

Draw a circle with a radius of 10.

```geo
MMI_DrawCircle( 80, 25, 10, 10,
                MMI_BRUSH_BLACK,
                MMI_PEN_BLACK )
```

6.1.32 MMI_DrawText

Description

Draw / delete text.

Declaration

```geo
MMI_DrawText( BYVAL iX AS Integer, 
              BYVAL iY AS Integer, 
              BYVAL sText AS String20, 
              BYVAL iAttr AS Integer, 
              BYVAL iPen AS Integer )
```

Remarks

This function either draws (`iPen = MMI_PEN_BLACK`) or deletes (`iPen = MMI_PEN_WHITE`) a text string in graphic field. The co-ordinates (`iX, iY`) correspond to the upper left-hand corner of the first character. The character size is 6 x 8 pixel.

Note

A graphics dialog has to be set up before.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iX</td>
<td>Integer</td>
<td>x-co-ordinate at the upper left-hand corner of the first character [pixel]</td>
</tr>
<tr>
<td>iY</td>
<td>Integer</td>
<td>y-co-ordinate at the upper left-hand corner of the first character [pixel]</td>
</tr>
<tr>
<td>sText</td>
<td>String20</td>
<td>Pointer to the text string</td>
</tr>
<tr>
<td>iAttr</td>
<td>Integer</td>
<td>Text attribute</td>
</tr>
<tr>
<td>iPen</td>
<td>Integer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TXT_NORMAL</td>
<td>normal text</td>
</tr>
<tr>
<td>MMI_TXT_INVERSE</td>
<td>inverted text</td>
</tr>
</tbody>
</table>
### iPen in MMI_PEN_BLACK

**iPen** in **MMI_PEN_BLACK**
- **draw text**
- **MMI_PEN_WHITE**
- **delete text**

#### Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>No graphics dialog exists for this operation.</td>
</tr>
</tbody>
</table>

### See Also

- MMI_CreateGraphDialog, MMI_DrawLine, MMI_DrawRect, MMI_DrawCircle

### Example

Print a text at position 10, 10.

```basic
DIM sOutput AS String20
sOutput = "distance"
MMI_DrawText( 10, 10, sOutput, MMI_TXT_NORMAL, MMI_PEN_BLACK )
```

---

### 6.1.33 MMI_DrawBusyField

#### Description
Shows or hides the Busy-Icon.

#### Declaration

```basic
MMI_DrawBusyField( 
BYVAL lVisible as Logical )
```

#### Remarks
This function controls the Busy-Icon (Hourglass).

#### Parameters

- **lVisible** in **TRUE:** Icon is visible

#### Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>
Example

The example shows and hides the Busy-Icon

`MMI_DrawBusyField(TRUE) ' show icon
  ' time consuming function....
MMI_DrawBusyField(FALSE) ' hide icon`

### 6.1.34 MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong

#### Description
Create an alert beep.

#### Declaration

```vba
MMI_BeepAlarm()
MMI_BeepNormal()
MMI_BeepLong()
```

#### Remarks

The functions create one or a sequence of alert beeps with configurable volume, if the boxes are turned on.

Any previously set continuous signal beep will be finished.

#### Return-Codes

- **RC_OK** Successful termination.

#### See Also

- `MMI_StartVarBeep`
- `MMI_SwitchVarBeep`
- `MMI_GetVarBeepStatus`

#### Example

The example uses the `MMI_BeepNormal` to sound a signal beep.

`MMI_BeepNormal()`

### 6.1.35 MMI_StartVarBeep

#### Description
Start beep sequences with configurable interrupts.

#### Declaration

```vba
MMI_StartVarBeep( BYVAL iRate AS Integer )
```

#### Remarks

The function creates sequences of beeps with configurable interrupts.
If previously a continuous signal beep has been set, the new rate will be established.

Parameters

iRate in  frequency in [%]; 0 is very slow, 100 is very fast

Return-Codes

RC_OK  Successful termination.

See Also  MMI_BeepAlarm,  
           MMI_BeepNormal,  
           MMI_BeepLong,  
           MMI_SwitchVarBeep,  
           MMI_GetVarBeepStatus

Example

The example uses the MMI_StartVarBeep to create a very fast sequence of signal beeps.

MMI_StartVarBeep(100)

6.1.36  MMI_SwitchVarBeep

Description  Switch a varying beep.

Declaration  MMI_SwitchVarBeep(BYVAL lOn AS Logical)

Remarks  The function allows the general switching (on/off) of a signal beep. A continuous signal beep will be switched off immediately.

Parameters

lOn in  switches the beep on or off

lOn meaning

FALSE  the beep is switched off generally
TRUE  beep is on; the functions MMI_BeepNormal etc. will only work if the beep is switched on.

Return-Codes

RC_OK  Successful termination.
### 6.1.37 MMI_GetVarBeepStatus

**Description**
Read the switch status for a variable signal beep.

**Declaration**
```plaintext
MMI_GetVarBeepStatus( lOn AS Logical )
```

**Remarks**
The function retrieves the state of the general signal beep switch.

**Parameters**

<table>
<thead>
<tr>
<th>lOn</th>
<th>Meaning</th>
<th>State of the Switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>off</td>
<td></td>
</tr>
<tr>
<td>TRUE</td>
<td>on</td>
<td></td>
</tr>
</tbody>
</table>

**Return-Codes**

- **RC_OK**  
  Successful termination.

**See Also**
- MMI_BeepNormal,
- MMI_BeepLong,
- MMI_BeepAlarm,
- MMI_StartVarBeep,
- MMI_SwitchVarBeep

---

Example

The example uses the `MMI_SwitchVarBeep` to switch off the beep.

```plaintext
MMI_SwitchVarBeep( TRUE )
```
Example  The example uses the MMI_GetVarBeepStatus to revert the beep status (i.e. switch on when it is off and vice versa).

    DIM lOn AS Logical
    MMI_GetVarBeepStatus(lOn)
    MMI_SwitchVarBeep( NOT lOn )

6.1.38 MMI_SwitchAFKey

Description  Switch the aF... key on or off.

Declaration  MMI_SwitchAFKEY( BYVAL lOn AS Logical )

Remarks  The function allows the switching (on/off) off the aF... key. Normally it is enabled, but during tracking distances it is disabled.

Parameters  

    lOn   in   switches the beep on or off

1On    meaning
    FALSE   Key is switched off generally
    TRUE    Key is active

Return-Codes

    RC_OK   Successful termination.

See Also  BAP_MeasRec,
   BAP_MeasDistAng

Example  The example uses the MMI_SwitchAFKey to disable the aF... key.

    MMI_SwitchAFKey( FALSE )
6.1.39  MMI_SwitchIconsBeep

Description  Switches measurement icons and special beeps on or off.

Declaration  MMI_SwitchIconsBeep( BYVAL lOn AS Logical )

Remarks  The function allows the switching (on/off) of the measurement icons and special beeps (sector and lost lock).

Parameters  
  lOn  in  switches the icons and beep on or off

  lOn  meaning
       FALSE  no measurement icons and no special beep
       TRUE  the measurement icons will be updated and the beeps are enabled. This is the normal state during a measurement dialog with continuous measurements.

Return-Codes

  RC_OK  Successful termination.

See Also  BAP_MeasRec
          BAP_MeasDistAng

Example  The example uses the MMI_SwitchIconsBeep to disable the icons and beeps.

          MMI_SwitchIconsBeep( FALSE )
6.1.40  MMI_SetAngleRelation

**Description**  
Set the angle relationship.

**Declaration**  
```basic
MMI_SetAngleRelation(
    BYVAL iVertRel AS Integer,
    BYVAL iHorzRel AS Integer)
```

**Remarks**  
This function sets the relationship of the vertical and horizontal angles. Fields already displayed are not updated.

**Parameters**

- **iVertRel**  
  Relationship of the vertical angle; valid values:
  - MMI_VANGLE_IN_PERCENT
  - MMI_VANGLE_REL_HORIZON
  - MMI_VANGLE_REL_ZENIT

- **iHorzRel**  
  Relationship of the horizontal angle; valid values:
  - MMI_HANGLE_CLOCKWISE
  - MMI_HANGLE_ANTICLOCKWISE
  - MMI_HANGLE_CLOCKWISE_SOUTH
  - MMI_HANGLE_BEARING

**Return Codes**

- **RC_OK**  
  Successful termination.

- **RC_IVPARAM**  
  The function has been called with an invalid parameter

**See Also**  
MMI_GetAngleRelation

**Example**

Set the angle relations (with internal default values).

```basic
MMI_SetAngleRelation(
    MMI_VANGLE_IN_PERCENT,
    MMI_HANGLE_CLOCKWISE)
```
6.1.41 MMI_GetAngleRelation

Description  Request the current angle relationships.

Declaration  

\[
\text{MMI\_GetAngleRelation}( \text{iVertRel AS Integer}, \text{iHorzRel AS Integer})
\]

Remarks  This function returns the current vertical- and horizontal- angle relationships.

Parameters  

\[
\begin{align*}
\text{iVertRel} & \quad \text{out} & \text{Relationship of the vertical angle} \\
\text{iHorzRel} & \quad \text{out} & \text{Relationship of the horizontal angle}
\end{align*}
\]

Return Codes  

none

See Also  MMI_SetAngleRelation

Example  Get the angle relations.

\[
\begin{align*}
\text{DIM iVertRel AS Integer} \\
\text{DIM iHorzRel AS Integer} \\
\text{MMI\_GetAngleRelation( iVertRel, iHorzRel )}
\end{align*}
\]

6.1.42 MMI_SetVAngleMode

Description  Set the V-Angle mode.

Declaration  

\[
\text{MMI\_SetVAngleMode(BYVAL lAngleFree AS Logical)}
\]

Remarks  This function sets the vertical angle mode. Normally (lAngleFree=FALSE), the vertical angle is fix if there is a valid distance available. If lAngleFree=TRUE, the vertical angle will be updated including all corresponding values (slope distance, vertical distance, coordinates etc)
6.1.43  **MMI_GetVAngleMode**

**Description**
Returns the V-Angle mode.

**Declaration**
```
MMI_GetVAngleMode(lAngleFree AS Logical)
```

**Remarks**
This function returns the vertical angle mode.

**Parameters**
- `lAngleFree` in TRUE: V-Angle is free (running)

**Return Codes**
- **RC_OK** Successful termination.

**See Also**
- `MMI_SetVAngleMode`

**Example**
See example file „meas.gbs“.

6.1.44  **MMI_SetAngleUnit**

**Description**
Set the displayed unit of angle.

**Declaration**
```
MMI_SetAngleUnit(BYVAL iUnit AS Integer, BYVAL iDigits AS Integer)
```

**Remarks**
This function sets the displayed unit of angle. Existing display fields are not updated. If `iDigits` is greater than the maximal number it will be reset to it without notifying the user. A negative value of `iDigits` is not allowed.

**Note**
The maximal number of decimal digits depends on the Theodolite class.
Parameters

iUnit in  Specified unit of angle; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_ANGLE_GON</td>
<td>400 Gon</td>
</tr>
<tr>
<td>MMI_ANGLE_DEC</td>
<td>360 Decimal</td>
</tr>
<tr>
<td>MMI_ANGLE_SEXADEC</td>
<td>360 Sexadecimal</td>
</tr>
<tr>
<td>MMI_ANGLE_MIL</td>
<td>6400 Mil</td>
</tr>
<tr>
<td>MMI_ANGLE_PERCENT</td>
<td>-300 ≤ x ≤ 300; only for vertical angles</td>
</tr>
</tbody>
</table>

iDigits in  Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_ANGLE_GON</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_DEC</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_SEXADEC</td>
<td>0-4</td>
</tr>
<tr>
<td>MMI_ANGLE_MIL</td>
<td>0-3</td>
</tr>
<tr>
<td>MMI_ANGLE_PERCENT</td>
<td>don’t care</td>
</tr>
</tbody>
</table>

Return Codes

RC_OK  Successful termination.
RC_IVPARAM  The function has been called with an invalid parameter.

See Also  MMI_GetAngleUnit

Example

Set the angle unit.

```
MMI_SetAngleUnit( MMI_ANGLE_GON, 3 )
```
6.1.45 MMI_GetAngleUnit

**Description**
Return the currently displayed unit of angle.

**Declaration**
```
MMI_GetAngleUnit(iUnit AS Integer, iDigits AS Integer)
```

**Remarks**
This function returns the current unit of angle.

**Parameters**
- **iUnit** out Specified unit of angle
- **iDigits** out Number of decimal places.

**Return Codes**
- **RC_OK** Successful termination.

**See Also**
MMI_SetAngleUnit

**Example**
Get the angle unit.
```
DIM iUnit AS Integer
DIM iDigits AS Integer
MMI_GetAngleUnit( iUnit, iDigits )
```

6.1.46 MMI_SetDistUnit

**Description**
Set the displayed unit of distance.

**Declaration**
```
MMI_SetDistUnit(BYVAL iUnit AS Integer, BYVAL iDigits AS Integer)
```

**Remarks**
This function sets the display unit for distance. Fields already displayed are not updated. If **iDigits** is greater than the maximal number it will be reset to it without notifying the user. A negative value of **iDigits** is not allowed.

**Note**
The maximal number of decimal digits depends on the Theodolite class
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iUnit</code></td>
<td>Specified unit of distance; possible values:</td>
<td><code>MMI_DIST_METER</code> (Meter), <code>MMI_DIST_FOOT</code> (normal foot), <code>MMI_DIST_FOOT_INCH</code> (normal foot / inch / 1/8inch), <code>MMI_DIST_US_FOOT</code> (US-foot), <code>MMI_DIST_US_FOOT_INCH</code> (US-foot / inch / 1/8inch), <code>MMI_DIST_MM</code> (Millimetre), <code>MMI_DIST_INCH</code> (inches)</td>
</tr>
<tr>
<td><code>iDigits</code></td>
<td>Number of decimal places. The maximum number of decimal places (<code>iDigits</code>) for each unit is set to the following values:</td>
<td><code>MMI_DIST_METER</code> (0-4), <code>MMI_DIST_FOOT</code> (0-4), <code>MMI_DIST_FOOT_INCH</code> (0-1), <code>MMI_DIST_US_FOOT</code> (0-4), <code>MMI_DIST_US_FOOT_INCH</code> (0-1), <code>MMI_DIST_MM</code> (0), <code>MMI_DIST_INCH</code> (0-3)</td>
</tr>
</tbody>
</table>

### Return Codes

- **RC_OK**: Successful termination.
- **RC_IVPARAM**: The function has been called with an invalid parameter.

### See Also

- `MMI_GetDistUnit`

### Example

Set the distance unit.

```plaintext
MMI_SetDistUnit( MMI_DIST_METER, 4 )
```
6.1.47 MMI_GetDistUnit

**Description**
Return the currently displayed unit of distance.

**Declaration**
```
MMI_GetDistUnit( iUnit AS Integer,
                 iDigits AS Integer)
```

**Remarks**
This function returns the current unit of distance.

**Parameters**
- `iUnit` out: Specified unit of distance
- `iDigits` out: Number of decimal places.

**Return Codes**
- `RC_OK`: Successful termination.

**See Also**
- `MMI_SetDistUnit`

**Example**
Get the distance unit.
```
DIM iUnit AS Integer
DIM iDigits AS Integer

MMI_GetDistUnit( iUnit, iDigits )
```

6.1.48 MMI_SetPressUnit

**Description**
Set the displayed unit of pressure.

**Declaration**
```
MMI_SetPressUnit(BYVAL iUnit AS Integer,
                  BYVAL iDigits AS Integer)
```

**Remarks**
This function sets the display unit for pressure. Fields already displayed are not updated. If `iDigits` is greater than 1 it will be reset to it without notifying the user. A negative value of `iDigits` is not allowed.
### Parameters

<table>
<thead>
<tr>
<th>iUnit in</th>
<th>Specified unit of pressure; possible values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>meaning</td>
</tr>
<tr>
<td>MMI_PRESS_MBAR</td>
<td>MilliBar</td>
</tr>
<tr>
<td>MMI_PRESS_MMHG</td>
<td>Millimetre mercury</td>
</tr>
<tr>
<td>MMI_PRESS_INCHHG</td>
<td>Inch mercury</td>
</tr>
<tr>
<td>MMI_PRESS_HPA</td>
<td>Hekto-Pascal</td>
</tr>
<tr>
<td>MMI_PRESS_PSI</td>
<td>PSI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>iDigits in</th>
<th>Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle unit</td>
<td>places</td>
</tr>
<tr>
<td>MMI_PRESS_MBAR</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_MMHG</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_INCHHG</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_HPA</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_PRESS_PSI</td>
<td>0-1</td>
</tr>
</tbody>
</table>

### Return Codes

- **RC_OK**  
  Successful termination.
- **RC_IVPARAM**  
  The function has been called with an invalid parameter

### See Also

- MMI_GetPressUnit

### Example

Set the pressure unit.

```basic
MMI_SetPressUnit( MMI_PRESS_MBAR, 1 )
```
6.1.49 MMI_GetPressUnit

**Description**
Return the currently displayed unit of pressure.

**Declaration**
```
MMI_GetPressUnit(iUnit AS Integer, 
iDigits AS Integer)
```

**Remarks**
This function returns the current unit of pressure.

**Parameters**
- **iUnit** out
  Specified unit of pressure
- **iDigits** out
  Number of decimal places.

**Return Codes**
- **RC_OK**
  Successful termination.

**See Also**
MMI_SetPressUnit

**Example**
Get the pressure unit.
```
DIM iUnit AS Integer
DIM iDigits AS Integer

MMI_GetPressUnit( iUnit, iDigits )
```

6.1.50 MMI_SetTempUnit

**Description**
Set the displayed unit of temperature.

**Declaration**
```
MMI_SetTempUnit(BYVAL iUnit AS Integer, 
BYVAL iDigits AS Integer)
```

**Remarks**
This function sets the display unit for temperature. Fields already displayed are not updated. If **iDigits** is greater than 1 it will be reset to it without notifying the user. A negative value of **iDigits** is not allowed.
Parameters

iUnit in  Specified unit of temperature; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TEMP_C</td>
<td>Celsius</td>
</tr>
<tr>
<td>MMI_TEMP_F</td>
<td>Fahrenheit</td>
</tr>
</tbody>
</table>

iDigits in  Number of decimal places. The maximum number of decimal places (iDigits) for each unit is set to the following values:

<table>
<thead>
<tr>
<th>angle unit</th>
<th>places</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_TEMP_C</td>
<td>0-1</td>
</tr>
<tr>
<td>MMI_TEMP_F</td>
<td>0-1</td>
</tr>
</tbody>
</table>

Return Codes

RC_OK  Successful termination.

RC_IVPARAM  The function has been called with an invalid parameter

See Also  MMI_GetTempUnit

Example  Set the temperature unit.

MMI_SetTempUnit( MMI_TEMP_C, 1 )

6.1.51  MMI_GetTempUnit

Description  Return the currently displayed unit of temperature.

Declaration  MMI_GetTempUnit(iUnit AS Integer, iDigits AS Integer)

Remarks  This function returns the current unit of temperature.

Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>direction</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iUnit</td>
<td>out</td>
<td>Specified unit of temperature</td>
</tr>
<tr>
<td>iDigits</td>
<td>out</td>
<td>Number of decimal places</td>
</tr>
</tbody>
</table>
Return Codes

- RC_OK: Successful termination.

See Also
- MMI_SetTempUnit

Example
- Get the temperature unit.
  ```plaintext
dim iunit as integer
  dim idigits as integer
  MMI_GetTempUnit( iUnit, iDigits )
```

6.1.52 MMI_SetDateFormat

Description
- Set the date display format.

Declaration
- `MMI_SetDateFormat(BYVAL iFormat AS Integer)`

Remarks
- This function sets the format in which the date is to be displayed. Existing fields remain unchanged.

Parameters
- `iFormat in` Specified date format; possible values:
  - `value` `meaning`
    - `MMI_DATE_EU` European: `DD.MM.YY`
    - `MMI_DATE_US` US: `MM/DD/YY`
    - `MMI_DATE_JP` Japanese: `YY/MM/DD`

Return Codes
- RC_OK: Successful termination.
- RC_IVPARAM: The function has been called with an invalid parameter.

See Also
- MMI_GetDateFormat
Example  Set the date format (internal default value).

`MMI_SetDateFormat( MMI_DATE_EU )`

### 6.1.53 MMI_GetDateFormat

**Description**  Retrieves the date display format.

**Declaration**  `MMI_GetDateFormat(iFormat AS Integer)`

**Remarks**  This function retrieves the format used to display the date.

**Parameters**

- `iFormat`  out  Specified date format

**Return Codes**

- `RC_OK`  Successful termination.

**See Also**  `MMI_SetDateFormat`

**Example**  Get the date format.

```basic
DIM iFormat AS Integer
MMI_GetDateFormat( iFormat )
```

### 6.1.54 MMI_SetTimeFormat

**Description**  Set the time display format.

**Declaration**  `MMI_SetTimeFormat(BYVAL iFormat AS Integer)`

**Remarks**  This function sets the format in which the time is to be displayed. Existing fields remain unchanged.

**Parameters**

- `iFormat`  in  Specified time format; possible values:
  - `MMI_TIME_12H`  12 hour display
  - `MMI_TIME_24H`  24 hour display
Return Codes

RC_OK  Successful termination.
RC_IVPARAM  The function has been called with an invalid parameter

See Also  MMI_GetTimeFormat

Example  Set the time format (internal default value).

MMI_SetTimeFormat( MMI_TIME_12H )

6.1.55  MMI_GetTimeFormat

Description  Retrieves the time display format.

Declaration  MMI_GetTimeFormat(iFormat AS Integer)

Remarks  This function retrieves the format used to display the time.

Parameters

iFormat  out  Specified time format

Return Codes

RC_OK  Successful termination.
RC_IVPARAM  The function has been called with an invalid parameter

See Also  MMI_SetTimeFormat

Example  Get the time format.

DIM iFormat AS Integer

MMI_GetTimeFormat( iFormat )
6.1.56 MMI_SetCoordOrder

Description  Set the co-ordinate order.

Declaration  MMI_SetCoordOrder(BYVAL iOrder AS Integer)

Remarks  This function sets the order of co-ordinates. The fields already displayed are not changed.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOrder</td>
<td>in</td>
</tr>
</tbody>
</table>

Specifies the co-ordinate order; possible values:

<table>
<thead>
<tr>
<th>value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_COORD_N_E</td>
<td>Order North East</td>
</tr>
<tr>
<td>MMI_COORD_E_N</td>
<td>Order East North</td>
</tr>
</tbody>
</table>

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
</tbody>
</table>

See Also  MMI_GetCoordOrder

Example  Set the co-ordinate order (internal default value).

MMI_SetCoordOrder( MMI_COORD_N_E )
6.1.57  MMI_GetCoordOrder

**Description**  Retrieve the co-ordinate order.

**Declaration**  

```plaintext
MMI_GetCoordOrder(iOrder AS Integer)
```

**Remarks**  This function retrieves the order in which co-ordinates are displayed.

**Parameters**  

- `iOrder`  out  Specified co-ordinate order

**Return Codes**  

- `RC_OK`  Successful termination.

**See Also**  

- `MMI_SetCoordOrder`

**Example**  

Get the co-ordinate order.

```plaintext
DIM iOrder AS Integer
MMI_GetCoordOrder( iOrder )
```

6.1.58  MMI_SetLanguage

**Description**  Set the display language.

**Declaration**  

```plaintext
MMI_SetLanguage(
    BYVAL iLanguageNr AS Integer )
```

**Remarks**  This function sets the current language. All displayed text are immediately shown in the new language.

**Parameters**  

- `iLanguageNr`  in  Specifies the language number; possible values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_REF_LANGUAGE</td>
<td>Reference language (English) = 1</td>
</tr>
<tr>
<td>2..</td>
<td>Language numbers</td>
</tr>
</tbody>
</table>

```
### 6.1.59 MMI_GetLanguage

#### Description
Query the current language.

#### Declaration
```
MMI_GetLanguage( iLangNr AS Integer, 
                 sLangName AS String20)
```

#### Remarks
This function returns the current language and the associated character symbols.

#### Parameters
- `iLangNr` out  Language number
- `sLangName` out  Language description

#### Return Codes
- **RC_OK**  Successful termination.

#### See Also
- MMI_SetLanguage

#### Example
Get the current language.
```
DIM iLangNr AS Integer
DIM sLangName AS String20
MMI_GetLanguage( iLangNr, sLangName )
```
6.1.60  MMI_GetLangName

Description  Gets the name to a language number.

Declaration  MMI_GetLangName(
            ByVal iLangNr AS Integer,
            sLangName AS String20)

Remarks  This routine delivers the name associated with the number iLangNr.

Parameters  
            iLangNr    in    Language number
            sLangName  out   Language description

Return Codes  
            RC_OK      Successful termination.
            RC_IVPARAM iLangNr is invalid

See Also  MMI_SetLanguage
            MMI_GetLanguage

Example  Get the name of a language.
            DIM sLangName AS String20
            MMI_GetLangName( 2, sLangName )
### 6.2 BASIC APPLICATIONS BAP

#### 6.2.1 Summarizing Lists of BAP Types and Procedures

#### 6.2.1.1 Procedures

<table>
<thead>
<tr>
<th>Procedure Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SetAccessories DLg</td>
<td>Sets the used accessories</td>
</tr>
<tr>
<td>BAP_FineAdjust</td>
<td>Automatic target positioning</td>
</tr>
<tr>
<td>BAP_GetMeasPrg</td>
<td>Get the current distance measure program.</td>
</tr>
<tr>
<td>BAP_MeasDistAngle</td>
<td>Measures distance and angles.</td>
</tr>
<tr>
<td>BAP_MeasRec</td>
<td>Measures and record distance and angles.</td>
</tr>
<tr>
<td>BAP_PosTelescope</td>
<td>Positioning of the Telescope.</td>
</tr>
<tr>
<td>BAP_SearchPrism</td>
<td>Searches the prism.</td>
</tr>
<tr>
<td>BAP_SetHz</td>
<td>Sets the horizontal angle to 0 or another given value.</td>
</tr>
<tr>
<td>BAP_SetManDist</td>
<td>Set the distance manually.</td>
</tr>
<tr>
<td>BAP_SetMeasPrg</td>
<td>Set the distance measure program.</td>
</tr>
<tr>
<td>BAP_SetPpm</td>
<td>Sets the ppm for distance measurements.</td>
</tr>
<tr>
<td>BAP_SetPrism</td>
<td>Sets the current prism type and constant.</td>
</tr>
</tbody>
</table>
6.2.2  BAP_SetAccessoriesDlg

Description  Sets the used accessories.

Declaration  BAP_SetAccessoriesDlg()

Remarks  This function displays the accessories dialog.

Parameters  

Return-Codes

-  RC_OK  Successful termination.

Example  The example displays the accessories dialog

BAP_SetAccessoriesDlg()

6.2.3  BAP_MeasDistAngle

Description  Measures distance and angles.

Declaration  BAP_MeasDistAngle( iDistMode AS Integer,  
                            dHz AS Angle,  
                            dV AS Angle,  
                            dDist AS Distance,  
                            BYVAL lDisplayOn AS Logical,  
                            BYVAL sCaptionLeft AS _Token )

Remarks  Measures distance and angles and updates the data pool after  
correct measurements. It controls the special beep (Sector or Lost  
Lock) and switches measurement icons and disables the aF...  
key during tracking.
### Parameters

The `iDistMode` parameter determines the distance measuring mode.

#### Mode as Input

<table>
<thead>
<tr>
<th>Mode as Input</th>
<th>Distance measuring modes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_NO_MEAS</td>
<td>No new measurement, get last one</td>
</tr>
<tr>
<td>BAP_NO_DIST</td>
<td>No distance measurement, get only angles</td>
</tr>
<tr>
<td>BAP_DEF_DIST</td>
<td>Measure distance and angles using default measurement program</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Measure distance and angles using the tracking measurement program</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Measure distance and angles using the fast tracking measurement program</td>
</tr>
<tr>
<td>BAP_STOP_TRK</td>
<td>Stop tracking, no measurement. No valid results returned.</td>
</tr>
<tr>
<td>BAP_CLEAR_DIST</td>
<td>Clear distance (Theodolite data-pool), no measurement. No valid results returned.</td>
</tr>
<tr>
<td>BAP_RED_TRK_DIST</td>
<td>Measure distance and angles using the tracking with red laser measurement program</td>
</tr>
</tbody>
</table>

#### Mode returned

<table>
<thead>
<tr>
<th>Mode returned</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_DEF_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
</tbody>
</table>

**Returns**

- `BAP_DEF_DIST`

**Angles [rad], depends on**

<table>
<thead>
<tr>
<th>dHz, dV out</th>
<th></th>
</tr>
</thead>
</table>
iDistMode
dDist out Distance [m], depends on iDistMode
sCaptionLeft in Left caption for the distance measurement display.
lDisplayOn in TRUE: shows the distance measurement display during distance measurement.

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_  Warning, only angle measurement FULL_  valid, accuracy cannot be guaranteed CORRECTION
TMC_ANGLE_OK  Warning, only angle measurement OK  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_  Warning, measurement without full CORRECTION correction
TMC_SIGNAL_ERROR  Error, no signal on EDM (only in signal mode)
RC_ABORT  Error, measurement aborted
RC_IVPARAM  Error, invalid DistMode

See Also
BAP_MeasRec

Example
See example file „meas.gbs“.
The example uses the BAP_MeasDistAngle routine to measure a distance and angles.
DIM iDistMode  AS Integer
DIM dHz  AS Angle
DIM dV  AS Angle
DIM dDist  AS Distance

iDistMode = BAP_DEF_DIST
BAP_MeasDistAngle(iDistMode, dHz, dV, dDist, TRUE, "TEST")
6.2.4 BAP_MeasRec

**Description** Measures distance and angles records.

**Declaration**

```basic
BAP_MeasRec( iDistMode AS Integer,
             BYVAL lDisplayOn AS Logical,
             BYVAL sCaptionLeft AS _Token )
```

**Remarks** Measures distance and angles and updates the Theodolite data pool after correct measurements and records values according the predefined record mask. After recording, a running point number will be incremented.

It controls the special beep (Sector or Lost Lock), switches Measurement icons and disables aF... Key during tracking.

**Parameters**

- **iDistMode** Distance measuring modes:
  - **Mode as Input**
    - **BAP_NO_MEAS**
    - **BAP_NO_DIST**
      No distance measurement before recording
    - **BAP_DEF_DIST**
      Use default distance measurement program and record values
    - **BAP_TRK_DIST**
      Use the tracking measurement program and record values
    - **BAP_RTRK_DIST**
      Use the fast tracking measurement program and record values
    - **BAP_STOP_TRK**
      Stop tracking, no measurement and no recording
    - **BAP_CLEAR_DIST**
      Clear distance (Theodolite data pool), no measurement and no recording.
    - **BAP_RED_TRK_DIST**
      Use the tracking with red laser measurement program and record values
<table>
<thead>
<tr>
<th>Mode returned</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_DEF_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_TRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>BAP_RTRK_DIST</td>
<td>Depends on distance measurement. Can be changed during distance measurement.</td>
</tr>
<tr>
<td>All other modes</td>
<td>Returns BAP_DEF_DIST.</td>
</tr>
</tbody>
</table>

- **sCaptionLeft** in `in` Left caption for the distance measurement display.
- **lDisplayOn** in `TRUE`: shows the distance measurement display during distance measurement.

### Return Codes

- **RC_OK**: Successful termination.
- **WIR_NO_MEDIUM**: No storage medium is available.
- **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
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>No target detected</td>
</tr>
<tr>
<td>AUT_RC_TIMEOUT</td>
<td>Position not reached</td>
</tr>
<tr>
<td>BAP_CHANGE_ALL_TO_DIST</td>
<td>No prism has been found during distance measurement with ATR, command changed from &quot;All&quot; to &quot;Dist&quot;</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ACCURACY_GUARANTEE</td>
<td>Info, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>Warning, only angle measurement valid, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC sub-module already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>An error occurred during distance measurement.</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>Error, wrong setting of PPM or MM on EDM</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>TMC_SIGNAL_ERROR</td>
<td>Error, no signal on EDM (only in signal mode)</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Error, measurement aborted</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

**See Also**

BAP_MeasDistAngle, GSI_SetRecMask
6. System Functions

**Example**

See example file `.meas.gbs`.

The example uses the BAP_MeasMeasRec routine to record actual distance and angles (no new measurement).

```plaintext
DIM iDistMode AS Integer

iDistMode = BAP_NO_MEAS ' no measurement
BAP_MeasRec(iDistMode, FALSE, "")
```

---

### 6.2.5 BAP_FineAdjust

**Description**

Automatic target positioning.

**Declaration**

```plaintext
BAP_FineAdjust( 
    ByVal dSearchHz AS Angle, 
    ByVal dSearchV AS Angle )
```

**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 `dSearchV` in V-direction and by parameter `dSearchHz` in Hz-direction. If no target is found, the instrument turns back to the initial start position. The ATR mode must be enabled for this functionality, see `CSV_SetATRStatus` and `CSV_GetATRStatus`.

**Parameters**

- `dSearchHz` in Search range Hz
- `dSearchV` in Search range V

**Return Codes**

- **RC_OK** Successful termination.
- **AUT_RC_TIMEOUT** Timeout while positioning of one or both axes. The position fault lies above 100[cc].
- **AUT_RC_MOTOR_ERROR** Instrument has no ‘motorization’.
- **RC_FATAL** Fatal error.
- **RC_ABORT** Function aborted.
- **AUT_RC_NO_TARGET** No target found.
Multiple targets found.

Inadequate environment conditions.

During the determination of the angle deviation error detected, repeat fine positioning.

ATR mode not enabled, enable ATR mode.

ATR error, at repeated occur call service.

See Also
CSV_SetATRStatus, CSV_GetATRStatus

Example
The example see sample TRACKING.GBS.

### 6.2.6 BAP_SearchPrism

**Description**
Searches the prism.

**Declaration**
```basic
BAP_SearchPrism(
    BYVAL lShowMessages As Logical )
```

**Remarks**
This procedure searches the prism. The searching area depends on the defined searching area and on the setting of the additional working area.
This routine works only in ATR instruments and needs at least Firmware-Release 2.00.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lShowMessages</td>
<td>TRUE: show error-messages if there are problems to find the prism</td>
</tr>
</tbody>
</table>

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>AUT_RC_TIMEOUT</td>
<td>Timeout while positioning of one or both axes. The position fault lies above 100[cc].</td>
</tr>
</tbody>
</table>
6. System Functions

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT_RC_MOTOR_ERROR</td>
<td>Instrument has no ‘motorization’.</td>
</tr>
<tr>
<td>RC_FATAL</td>
<td>Fatal error.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Function aborted.</td>
</tr>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>No target found.</td>
</tr>
<tr>
<td>AUT_RC_MULTIPLE_</td>
<td>Multiple targets found.</td>
</tr>
<tr>
<td>TARGETS</td>
<td></td>
</tr>
<tr>
<td>AUT_RC_BAD_ENVIRONMENT</td>
<td>Inadequate environment conditions.</td>
</tr>
<tr>
<td>AUT_RC_DEV_ERROR</td>
<td>During the determination of the angle deviation error detected, repeat fine positioning</td>
</tr>
<tr>
<td>AUT_RC_NOT_ENABLED</td>
<td>ATR mode not enabled, enable ATR mode</td>
</tr>
</tbody>
</table>

See Also: CSV_SetATRStatus, CSV_GetATRStatus

6.2.7 BAP_SetManDist

**Description**
Set the distance manually.

**Declaration**
```plaintext
BAP_SetManDist ( 
    BYVAL sCaptionLeft AS _Token, 
    BYVAL dDistance AS Double, 
    iButtonId AS Integer )
```

**Remarks**
The BAP_SetManDist routine starts a dialog with the caption `sCaption` where the user can enter a horizontal distance. The distance will be stored into the Theodolite data pool.

**Parameters**

- **sCaptionLeft** in
  - left caption string of the dialog
- **dDistance** in
  - initial value for the distance. A negative value will be displayed as "----"
- **iButtonId** out
  - identifier of the pressed valid button to exit the dialog
Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>Info, accuracy cannot be guaranteed</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Error, no valid angle measurement</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Warning, only angle measurement valid</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Error, TMC sub-module already in use by another subsystem, command not processed</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>Warning, measurement without full correction</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>Error, invalid DistMode</td>
</tr>
</tbody>
</table>

See Also

TMC_IfDistTapeMeasured, TMC_SetHandDist, TMC_GetPolar, TMC_GetCoordinate

Example

The example uses the BAP_SetManDist routine to enter a distance.

```basic
DIM iButton AS Integer
DIM dInitDist AS Distance

dInitDist = 15.0 'initial value
BAP_SetManDist( "BASIC", dInitDist, iButton )
```

6.2.8 BAP_SetPpm

Description

Sets the PPM for distance measurements.

Declaration

BAP_SetPpm()

Remarks

The BAP_SetPpm routine opens a dialog which the user can complete in order to calculate the PPM (parts per million) correction to be used to reduce the distance measured by the EDM.

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>
6.2.9 BAP_SetPrism

**Description**
Sets the current prism type and constant.

**Declaration**
BAP_SetPrism()

**Remarks**
The `BAP_SetPrism` routine opens a dialog which the user can complete in order to choose one of five prism types/constants. Two types are LEICA defaults, whereas the other three can be named and the constant values given/changed by the user. The prism constants are always given and displayed in millimetres, regardless of the distance units in use at the time.

**Return Codes**
- **RC_OK** Successful termination.

**See Also**
BAP_SetManDist, BAP_SetPpm

**Example**
The example uses the `BAP_SetPrism` routine to open the Prism dialog.

BAP_SetPrism()

6.2.10 BAP_SetMeasPrg

**Description**
Set the distance measure program.

**Declaration**
BAP_SetMeasPrg( BYVAL iMeasPrg AS Integer )
### Remarks

The `BAP_SetMeasPrg` routine sets the program for the distance measurement.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iMeasPrg</code></td>
<td>Distance measure program</td>
</tr>
</tbody>
</table>

### Valid measure programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BAP_SINGLE_REF_STANDARD</code></td>
<td>Single measurement, with reflector, standard speed</td>
</tr>
<tr>
<td><code>BAP_SINGLE_REF_FAST</code></td>
<td>Single measurement, with reflector, fast</td>
</tr>
<tr>
<td><code>BAP_SINGLE_REF_VISIBLE</code></td>
<td>Single measurement, with reflector and red laser</td>
</tr>
<tr>
<td><code>BAP_SINGLE_RLESS_VISIBLE</code></td>
<td>Single measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td><code>BAP_CONT_REF_STANDARD</code></td>
<td>Continuous measurement, with reflector, standard speed</td>
</tr>
<tr>
<td><code>BAP_CONT_REF_FAST</code></td>
<td>Continuous measurement, with reflector, fast</td>
</tr>
<tr>
<td><code>BAP_CONT_RLESS_VISIBLE</code></td>
<td>Continuous measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td><code>BAP_AVG_REF_STANDARD</code></td>
<td>Average measurement, with reflector, standard speed</td>
</tr>
<tr>
<td><code>BAP_AVG_REF_VISIBLE</code></td>
<td>Average measurement, with reflector and red laser</td>
</tr>
<tr>
<td><code>BAP_AVG_RLESS_VISIBLE</code></td>
<td>Average measurement, reflectorless, with red laser</td>
</tr>
</tbody>
</table>

### See Also

`BAP_GetMeasPrg`
Example
The example uses the BAP_SetMeasPrg routine to set the
distance measurement program on single measurement without
reflector.
BAP_SetMeasPrg(BAP_SINGLE_RLESS_VISIBLE)

6.2.11 BAP_GetMeasPrg

Description
Get the current distance measure program.

Declaration
BAP_GetMeasPrg( iMeasPrg AS Integer )

Remarks
The BAP_GetMeasPrg routine fetches the current program for
the distance measurement.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMeasPrg</td>
<td>out</td>
<td>Distance measure program</td>
</tr>
</tbody>
</table>

Valid measure programs

<table>
<thead>
<tr>
<th>Measure Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_SINGLE_REF_STANDARD</td>
<td>Single measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>BAP_SINGLE_REF_FAST</td>
<td>Single measurement, with reflector, fast</td>
</tr>
<tr>
<td>BAP_SINGLE_REF_VISIBLE</td>
<td>Single measurement, with reflector and red laser</td>
</tr>
<tr>
<td>BAP_SINGLE_RLESS_VISIBLE</td>
<td>Single measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>BAP_CONT_REF_STANDARD</td>
<td>Continuous measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>BAP_CONT_REF_FAST</td>
<td>Continuous measurement, with reflector, fast</td>
</tr>
<tr>
<td>BAP_CONT_RLESS_VISIBLE</td>
<td>Continuous measurement, reflectorless, with red laser</td>
</tr>
<tr>
<td>BAP_AVG_REF_STANDARD</td>
<td>Average measurement, with reflector, standard speed</td>
</tr>
<tr>
<td>BAP_AVG_REF_VISIBLE</td>
<td>Average measurement, with reflector and red laser</td>
</tr>
<tr>
<td>BAP_AVG_RLESS_VISIBLE</td>
<td>Average measurement, reflectorless, with red laser</td>
</tr>
</tbody>
</table>
See Also  BAP_SetMeasPrg

Example  The example uses the BAP_GetMeasPrg routine to fetch the current distance measurement program.
DIM iMeasPrg AS Integer
BAP_GetMeasPrg(iMeasPrg)

6.2.12 BAP_PosTelescope

Description  Positioning of the Telescope.

Declaration  BAP_PosTelescope(
    BYVAL eMode AS Integer,
    BYVAL eDspMode AS Integer,
    BYVAL dHz AS Double,
    BYVAL dV AS Double,
    BYVAL dHzTolerance AS Double,
    BYVAL dVTolerance AS Double)

Remarks  This procedure positions the telescope according to the specified mode and angles.

Parameters  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMode</td>
<td>Positioning mode.</td>
</tr>
<tr>
<td>BAP_POSIT</td>
<td>positioning on Hz and V angle</td>
</tr>
<tr>
<td>BAP_POSIT_HZ</td>
<td>positioning on Hz angle</td>
</tr>
<tr>
<td>BAP_POSIT_V</td>
<td>positioning on V angle</td>
</tr>
<tr>
<td>BAP_CHANGE_FACE</td>
<td>change face</td>
</tr>
</tbody>
</table>
eDspMode

Controls the context and layout of the display during manual positioning.
This parameter has no effect on motorised Theodolites.

BAP_POS_NOMSG
No message will be displayed

BAP_POS_MSG
Only a message will be displayed

BAP_POS_DLG
Positioning will be guided with a dialog if it is a non motorised Theodolite

dHz, dV

Target position

dHzTolerance, dVTolerance

In case of manual positioning, the tolerances define the upper and lower boundaries of the target position. For successful termination of the positioning, the final target position must be within these boundaries. If the tolerance is lower then the default accuracy of the Theodolite, the tolerance will be the default accuracy.

Return Codes

RC_OK
Positioning successful

RC_ABORT
Abnormal termination (No positioning possible, ESC-Key)

See Also

CSV_MakePositioning
CSV_ChangeFace

Example

Position the telescope.

BAP_PosTelescope(BAP_CHANGE_FACE, BAP_POS_DLG, 0, 0, .5, .5)
6.2.13 BAP_SetHz

Description
Sets the horizontal angle to 0 or another given value.

Declaration
BAP_SetHz( BYVAL sCaptionLeft AS _Token )

Remarks
This procedure offers a dialogue which the user can complete in order to influence the angular offset provided by the TMC subsystem for the horizontal angle encoder. A button is provided for setting the angle to zero, directly, or the user may prefer to input another given value. Furthermore, the angle beep (at the quarter circle positions from 0°) can be turned on and off.

Note
If the instrument is in Lock mode, then the instrument tries to lock first before it sets the angle to 0.

Parameters
sCaptionLeft Left caption text for dialog

See Also

Return Codes
RC_OK Horizontal angular offset correct.

Example
Set the horizontal angle.

BAP_SetHz("BASIC")
6.3 MEASUREMENT FUNCTIONS TMC

This section contains the lower level measurement procedures.

6.3.1 Summarizing Lists of TMC Types and Procedures

6.3.1.1 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_ANG_SWITCH_Type</td>
<td>Angle measurement switches</td>
</tr>
<tr>
<td>TMC_Angle_Type</td>
<td>Data structure for measuring angles.</td>
</tr>
<tr>
<td>TMC_Coordinate_Type</td>
<td>Data structure for the co-ordinates (tracking and fixed co-ordinates).</td>
</tr>
<tr>
<td>TMC_DIST_SWITCHES_Type</td>
<td>Distance measurement switches</td>
</tr>
<tr>
<td>TMC_Distance_Type</td>
<td>Data structure for the distance measurement.</td>
</tr>
<tr>
<td>TMC_HZ_V_Ang_Type</td>
<td>Horizontal and vertical angle.</td>
</tr>
<tr>
<td>TMC_Incline_Type</td>
<td>Data structure for the inclination measurement.</td>
</tr>
<tr>
<td>TMC_OFFSET_DIST_Type</td>
<td>Target offset</td>
</tr>
<tr>
<td>TMC_PPM_CORR_Type</td>
<td>Correction for distance measurement</td>
</tr>
<tr>
<td>TMC_REFRACTION_Type</td>
<td>Refraction correction for distance measurement</td>
</tr>
<tr>
<td>TMC_STATION_Type</td>
<td>Station co-ordinates</td>
</tr>
</tbody>
</table>

6.3.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_DoMeasure</td>
<td>Start a measure program.</td>
</tr>
<tr>
<td>TMC_Get/</td>
<td>Gets and sets the current face definition.</td>
</tr>
<tr>
<td>SetAngleFaceDef</td>
<td></td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TMC_Get/GetRefractiveCorr</td>
<td>Gets and sets the refractive correction for measuring the distance.</td>
</tr>
<tr>
<td>TMC_Get/GetRefractiveMethod</td>
<td>Gets and sets the method of refractive correction for measuring the distance.</td>
</tr>
<tr>
<td>TMC_Get/GetDistPpm</td>
<td>Gets and sets the correction values for distance measurements.</td>
</tr>
<tr>
<td>TMC_Get/GetHeight</td>
<td>Gets and sets the current height of the reflector.</td>
</tr>
<tr>
<td>TMC_Get/GetHzOffset</td>
<td>Gets and sets the current horizontal offset.</td>
</tr>
<tr>
<td>TMC_Get/GetStation</td>
<td>Gets and sets station co-ordinates.</td>
</tr>
<tr>
<td>TMC_Get/Angle</td>
<td>Measure angles.</td>
</tr>
<tr>
<td>TMC_Get/Angle_Winc</td>
<td>Measure angles with inclination control</td>
</tr>
<tr>
<td>TMC_Get/AngleSwitch</td>
<td>Returns the angle measurement correction switches</td>
</tr>
<tr>
<td>TMC_Get/Coordinate</td>
<td>Calculate and read co-ordinates.</td>
</tr>
<tr>
<td>TMC_Get/DistSwitch</td>
<td>Returns the distance measurement correction switches</td>
</tr>
<tr>
<td>TMC_Get/Face1</td>
<td>Get face information of current telescope position</td>
</tr>
<tr>
<td>TMC_Get/InclineStatus</td>
<td>Returns the inclination compensator status.</td>
</tr>
<tr>
<td>TMC_Get/InclineSwitch</td>
<td>Returns the compensator switch</td>
</tr>
<tr>
<td>TMC_Get/OffsetDist</td>
<td>Returns the distance measurement offset</td>
</tr>
<tr>
<td>TMC_Get/Polar</td>
<td>Calculate and read polar co-ordinates.</td>
</tr>
<tr>
<td>TMC_Get/SimpleMea</td>
<td>Gets the results of distance and angle measurement</td>
</tr>
<tr>
<td>TMC_IfDistTapeMeasured</td>
<td>Gets information about manual measurement.</td>
</tr>
<tr>
<td>TMC_IfOffsetDistMeasured</td>
<td>Returns the EDM measurement mode</td>
</tr>
<tr>
<td>TMC_QuickDist</td>
<td>Measure slope distance and angles</td>
</tr>
<tr>
<td>TMC_Set/AngleSwitch</td>
<td>Defines the angle measurement correction switches</td>
</tr>
<tr>
<td>TMC_Set/DistSwitch</td>
<td>Defines the distance measurement correction switches</td>
</tr>
<tr>
<td>TMC_Set/HandDist</td>
<td>Sets distance manually.</td>
</tr>
</tbody>
</table>
6.3.2 TMC Data Structures

6.3.2.1 TMC_INCLINE - Data structure for the inclination measurement

```
TYPE TMC_Incline_Type
  dCrossIncline AS Double  cross inclination
  dLengthIncline AS Double  alongside inclination
  dAccuracyIncline AS Double  accuracy of measuring
  InclineTime AS Integer  time of measuring
END TMC_Incline_Type
```

6.3.2.2 TMC_ANGLE - Data structure for measuring angles

```
TYPE TMC_Angle_Type
  dHz AS Double  horizontal angle
  dV AS Double  vertical angle
  dAngleAccuracy AS Double  accuracy of angle
  iAngleTime AS Integer  time of measurement
  Incline AS TMC_Incline_Type  inclination belonging to the measurement
  iFace AS Integer  information about position of the telescope
END TMC_Angle_Type
```
6.3.2.3 TMC_DISTANCE - Data structure for the distance measurement

```
TYPE TMC_Distance_Type
    Angle AS TMC_Angle_Type
    set of angles belonging to distance
    dSlopeDist AS Double
    slope distance
    dSlopeDistAccuracy AS Double
    accuracy of distance
    dHorizDist AS Double
    horizontal distance
    dHeightDiff AS Double
    difference in altitude
    AngleCont AS TMC_Angle_Type
    set of angles, measured continuously
    dSlopeDistCont AS Double
    slope distance, measured continuously
    dHeightDiffCont AS Double
    distance in altitude, measured continuously
    dHorizDistCont AS Double
END TMC_Distance_Type
```

6.3.2.4 TMC_COORDINATE - Data structure for the coordinates (tracking and fixed co-ordinates)

```
TYPE TMC_Coordinate_Type
    dE AS Double
    east co-ordinate
    dN AS Double
    north co-ordinate
    dH AS Double
    height co-ordinate
    iCoordTime AS Integer
    time of measurement
    dE_Cont AS Double
    east coordinate, measured continuously
    dN_Cont AS Double
    north co-ordinate, measured continuously
    dH_Cont AS Double
    height co-ordinate, measured continuously
    iCoordContTime AS Integer
    time of continuous measurement
END TMC_Coordinate_Type
```

6.3.2.5 TMC_HZ_V_Ang - Horizontal and vertical angle

```
TYPE TMC_HZ_V_Ang_Type
    dHz AS Double
    horizontal angle
    dV AS Double
    vertical angle
END TMC_HZ_V_Ang_Type
```
6.3.2.6 TMC_PPM_CORR - Correction for distance measurement

    TYPE TMC_PPM_CORR_Type
        dPpmI AS Double individual
        dPpmA AS Double atmospheric
        dPpmR AS Double height relative
        dPpmP AS Double projection contortion
    END TMC_PPM_CORR_Type

6.3.2.7 TMC_STATION - Station coordinates

    TYPE TMC_STATION_Type
        dE0 AS Double easting co-ordinate
        dN0 AS Double northing co-ordinate
        dH0 AS Double height co-ordinate
        dHi AS Double instrument height
    END TMC_STATION_Type

6.3.2.8 TMC_REFRACTION- Refraction correction for distance measurement

    TYPE TMC_REFRACTION_Type
        bOnOff AS Logical TRUE if refraction is valid
        dEarthRadius AS Double earth radius
        dRefractiveScale AS Double refraction coefficient
    END TMC_REFRACTION_Type

6.3.2.9 TMC_DIST_SWITCH_Type– Distance measurement switches

    TYPE TMC_DIST_SWITCHES_Type
        lAxisDifferCorr AS Logical ' EDM to optical axis correction
        lProjectScaleCorr AS Logical ' Projection scale correction
        lHgtReductionCorr AS Logical ' Height reduction correction
    END TMC_DIST_SWITCHES_Type
6.3.2.10 TMC_ANGLE_SWITCH_Type – Angle measurement switches

TYPE TMC_ANG_SWITCH_Type
  lInclineCorr AS Logical ' Inclination correction
  lStandAxisCorr AS Logical ' Standing axis correction
  lCollimationCorr AS Logical ' Collimation error correction
  lTiltAxisCorr AS Logical ' Tilting axis correction
END TMC_ANG_SWITCH_Type

6.3.2.11 TMC_OFFSET_DIST_Type – Target offset

TYPE TMC_OFFSET_DIST_Type
  dLengthVal AS Distance ' Target - Offset Length
  dCrossVal AS Distance ' Target - Offset Cross
  dHeightVal AS Distance ' Target - Offset Height
END TMC_OFFSET_DIST_Type

6.3.3 TMC_DoMeasure

Description Start a measure program.

Declaration TMC_DoMeasure( BYVAL iCommand AS Integer )

Remarks With this function a measure program is started. The commands start a distance measurement and / or a test mode. In addition an angle- and an inclination-measure are done (not at measurement).

The tracking measure program performs e.g. as follows: Start the measure program with TMC_DoMeasure(TMC_TRK_DIST). The electronic distance measuring device (EDM) begins to run. Now the co-ordinates can be read, e.g. with TMC_GetCoordinate(). Tracking can be stopped with TMC_DoMeasure(TMC_STOP). With TMC_DoMeasure(TMC_CLEAR) the function will be stopped and the distance cleared.
Note: After calling a measure program, the last valid distance results will be cleared (as after TMC_STOP).

Parameters

<table>
<thead>
<tr>
<th>iCommand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_STOP</td>
<td>switch off EDM and finish program</td>
</tr>
<tr>
<td>TMC_DEF_DIST</td>
<td>do default distance measure</td>
</tr>
<tr>
<td>TMC_TRK_DIST</td>
<td>do tracking distance measure</td>
</tr>
<tr>
<td>TMC_RTRK_DIST</td>
<td>do fast tracking distance measure</td>
</tr>
<tr>
<td>TMC_CLEAR</td>
<td>clear distance and switch off EDM</td>
</tr>
<tr>
<td>TMC_SIGNAL</td>
<td>start signal measurement (test mode)</td>
</tr>
<tr>
<td>TMC_RED_TRK_DIST</td>
<td>do tracking distance measure with red laser</td>
</tr>
</tbody>
</table>

See Also

TMC_GetPolar
TMC_GetCoordinate

Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>measure program started</td>
</tr>
<tr>
<td>RC_INVALIDPARAM</td>
<td>The function has been called with an invalid parameter</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>Measurement system is busy</td>
</tr>
</tbody>
</table>
Example  
Start a distance measure, do something, stop it and clear results.

The following variable has to be defined:

```
TMC_DoMeasure (TMC_DEF_DIST) ' ... do a measure
TMC_DoMeasure (TMC_CLEAR)
```

6.3.4 TMC_GetPolar

Description  
Calculate and read polar co-ordinates.

Declaration  
```
TMC_GetPolar(
    BYVAL iWaitTime AS Integer,
    Polar AS TMC_Distance_Type,
    iReturnCode AS Integer )
```

Remarks  
The function corrects and takes in calculation a measured distance. Angle and possibly inclination are being calculated. The result is a point in polar co-ordinates.

Simple and multiple measures (distance tracking, altitude tracking) are supported. The horizontal and the inclined distance with the difference in altitude are read. The delay (iWaitTime) just works on the distance measure, not on the measure of the angle. As long as no new measure program is started, the results can be read. Additional to the normal return codes iReturnCode delivers also informational return codes which will not interrupt program execution.

Note  
The measure program must have been started (see TMC_DoMeasure).

Parameters  
```
iWaitTime  in  delay time [ms] until a result is available
            =0  returns results with an already measured distance.
```
>0 waits maximal the time 
iWaitTime for a result. If 
iWaitTime is chosen big 
enough (e. g. 60000, which is 
surely longer than the time-out 
period of the device), the system 
will wait for a result or until an 
error occurs.

<0 Performs an automatic target 
acquisition (if possible) and then 
tries to measuring in a until a 
valid result or an irrecoverable 
error occurs. The value itself of 
iWaitTime is ignored.

Polar out point in polar co-ordinates 
iReturnCode out see Additional Codes below

See Also TMC_GetCoordinates

Additional Codes in iReturnCode

RC_OK measurement and values are OK
TMC_ACCURACY_GUARANTEE Accuracy is not guaranteed, because the 
results 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.
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.

#### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Measurement and values are OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy. Repeat measurement.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>

#### Example

Start a distance measure, perform measure.

```basic
DIM iRetCode AS Integer
DIM iWaitTime AS Integer
DIM Polar AS TMC_Distance_Type
DIM lError AS Logical
DIM lDone AS Logical
```
'start distance measurement
ON ERROR RESUME ' to get valid angles
TMC_DoMeasure( TMC_DEF_DIST )

iWaitTime = -1
iDone = FALSE
iError = FALSE

DO 'display measured values
   TMC_GetPolar( iWaitTime, Polar, iRetCode )
   SELECT CASE iRetCode
     CASE RC_OK
       'display all data
       'e.g. set lDone here
     CASE else
       'handle error
       iError = TRUE
     END SELECT
   LOOP UNTIL iError OR iDone

'stop distance measurement
TMC_DoMeasure( TMC_CLEAR )

6.3.5 TMC_GetCoordinate

Description Calculate and read co-ordinates.

Declaration TMC_GetCoordinate(
    BYVAL iWaitTime AS Integer,
    Coordinate AS TMC_COORDINATE_Type,
    iReturnCode AS Integer )

Remarks The function calculates and output co-ordinates. Angle and possibly inclination are being measured. The co-ordinates are being corrected. The result is a point in Cartesian co-ordinates. The system calculates co-ordinates and tracking co-ordinates. Simple and multiple measurements (distance-, altitude- and co-ordinate-tracking) are supported. The delay (iWaitTime) just works on the distance measure, not on the measuring of the angle.
As far as no new measure program is started, the results can be read. Additional to the normal return codes iReturnCode delivers also informational return codes which will not interrupt program execution.

**Note**

The measure program must have been started (see TMC_DoMeasure).

**Parameters**

- **iWaitTime**
  - **in** delay time [ms] until a result is available
  - =0 returns already measured values
  - >0 waits the maximal time
  - iWaitTime for a result

- **Coordinate**
  - **out** point in Cartesian co-ordinates (output)

- **iReturnCode**
  - **out** return code, see Additional Codes

**See Also**

TMC_GetPolar

**Additional Codes in iReturnCode**

- **RC_OK**
  - measurement and values are OK
- **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.
- **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.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_ANGLE_NO_FULLCORRECTION</td>
<td>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.</td>
</tr>
<tr>
<td>TMC_DIST_ERROR</td>
<td>No measuring, because of missing target point, co-ordinates are not available. Aim target point and try it again.</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>measurement and values are OK</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>TMC resource is locked respectively TMC task is busy. Repeat measurement.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>Measurement through customer aborted.</td>
</tr>
</tbody>
</table>

**Example**

```basic
Start a distance measure, perform measurement.
DIM iretCode AS Integer
DIM iWaitTime AS Integer
DIM Coord AS TMC_COORDINATE_Type
DIM lError AS Logical
DIM lDone AS Logical
ON ERROR RESUME NEXT ' to get valid angle data
TMC_DoMeasure( TMC_DEF_DIST )
lDone = FALSE
lError = FALSE
```
DO ' display measured values
   TMC_GetCoordinate( 5, Coord, iRetCode )
SELECT CASE iRetCode
   CASE RC_OK
     'display all data
     'e.g. set lDone
   CASE ANGLE_OK
     ' display coordinate
   CASE ELSE
     'handle error
     lError = TRUE
END SELECT
LOOP UNTIL lError OR lDone
TMC_DoMeasure( TMC_CLEAR )

6.3.6 TMC_GetAngle

Description Measure angles.

Declaration TMC_GetAngle( Angles AS TMC_ANGLE_Type,
iReturnCode AS Integer )

Remarks The function measures the horizontal and vertical angle and the possibly belonging inclination, if the inclination compensation is on. If the compensation is off and no valid inclination is present, there may be a delay if the inclination can't be measured immediately. The correction values for the inclination can be calculated with several methods.

As long as no new measure program is started, the results can be read. Additional to the normal return codes iReturnCode delivers also informational return codes which will not interrupt program execution.

Parameters Angles out result of measuring the angle
   iReturnCode out return code, see Additional Codes

See Also TMC_DoMeasure
**Additional Codes in iReturnCode**

- **RC_OK**: Execution successful.
- **TMC_NO_FULL_CORRECTION**: The results are not corrected by all active sensors. Angle data are available. This message is to be considered as a 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 perform a forced incline measurement or switch off the incline. This message is to be considered as info.

**Return Codes**

- **RC_OK**: Angle OK
- **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.

**Example**

Read the currently valid angle.

```basic
DIM Angles AS TMC_ANGLE_Type
DIM RetCode AS Integer

TMC_GetAngle( Angles, RetCode )
```
6.3.7 TMC_GetAngle_WInc

Description
Measure angles with inclination control.

Declaration
TMC_GetAngle_WInc(
    iIncProg AS Integer,
    Angle AS TMC_ANGLE,
    iReturnCode AS Integer
)

Remarks
The function measures the horizontal and vertical angle and in
dependence of the configuration, the inclination.

As far as no new measure program is started, the results can be
read. Additional to the normal return codes iReturnCode
delivers also informational return codes, which will not interrupt
program execution.

Parameters
iIncProg in The manner of incline compensation.
Following settings are possible:

<table>
<thead>
<tr>
<th>Incline Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_MEA_INC</td>
<td>get inclination (apriori sigma)</td>
</tr>
<tr>
<td>TMC_AUTO_INC</td>
<td>get inclination with automatism</td>
</tr>
<tr>
<td>TMC_PLANE_INC</td>
<td>get inclination always with plane</td>
</tr>
</tbody>
</table>

Angle out result of measuring the angle
iReturnCode out return code, see Additional Codes

See Also
TMC_DoMeasure, TMC_GetAngle

Additional Codes in iReturnCode

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Execution successful.</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. Angle data are available.</td>
</tr>
<tr>
<td></td>
<td>This message is to be considered as warning.</td>
</tr>
</tbody>
</table>
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.

Return Codes

- `RC_OK`: angle OK
- `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.

Example

Read the currently valid angle.

```basic
DIM Angles AS TMC_Angle
DIM iRetCode AS Integer

TMC_GetAngle_WInc(TMC_AUTO_INC, Angles, iRetCode)
```

### 6.3.8 TMC_QuickDist

**Description**  
Measure slope distance and angles.

**Declaration**

```basic
TMC_QuickDist (  
    Angle AS TMC_HZ_V_ANG_type,  
    Dist AS Distance,  
    iReturnCode AS Integer )
```

**Remarks**  
The function measures the horizontal and vertical angle and in dependence of the configuration, the inclination. 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 active with the standard tracking measuring program already, the measuring program will not be 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.

<table>
<thead>
<tr>
<th>Note: Due to performance reasons the used inclination will be calculated (only if incline is activated). If 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_WInc(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.).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
</tr>
<tr>
<td>Distance</td>
</tr>
<tr>
<td>iReturnCode</td>
</tr>
</tbody>
</table>

See Also TMC_DoMeasure, TMC_GetAngle
<table>
<thead>
<tr>
<th>Additional Codes in iReturnCode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Execution successful.</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. Angle data are available. This message is to be considered as warning.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>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 force incline measurement perform or switch off the incline. This message is to be considered as info.</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>Problems with angle res. incline sensor. A valid angle could not be measured. Angle data are not available. At repeated occur call service.</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Angle measuring data are valid, but no distance data available. (Possible reasons are: - time out period to short - target out of view) This message is to be considered as warning.</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>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 considered as warning.</td>
</tr>
</tbody>
</table>
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 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.

Return Codes

RC_OK   angle OK
TMC_ANGLE_ERROR   Problems with angle res. incline sensor.  
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.

Example  Fast tracking with QuickDist. See example program TRACKING for more details.
DIM iRetCode AS Integer
DIM HzV AS TMC_HZ_V_ANG_Type
DIM dDist AS Distance
TMC_DoMeasure( TMC_CLEAR ) ' clear distances
' measurement loop  
DO
  ' get measurement values
  TMC_QuickDist( HzV, dDist, iRetCode )
  IF iRetCode = RC_OK OR
      iRetCode = TMC_NO_FULL_CORRECTION OR
      iRetCode = TMC_ACCURACY_GUARANTEE THEN
    ' Angles and distance are valid
    ' ...
    ELSE
      ' only Angles are valid
    ' ...
    END IF
 LOOP UNTIL ....

' terminate
TMC_DoMeasure( TMC_CLEAR ) ' stop measurement

6.3.9 TMC_GetSimpleMea

**Description**  Gets the results of distance and angle measurement.

**Declaration**  TMC_GetSimpleMea(  
  Angles AS TMC_HZ_V_ANG_Type,  
  dSlopeDist AS Double,  
  iReturnCode AS Integer )

**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.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angles out</td>
<td>result of measuring: the angles</td>
</tr>
<tr>
<td>dSlopeDist out</td>
<td>slope distance [m]</td>
</tr>
<tr>
<td>iReturnCode out</td>
<td>return code, see Additional Codes</td>
</tr>
</tbody>
</table>

## See Also

TMC_DoMeasure

## Additional Codes in iReturnCode

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Angle OK</td>
</tr>
<tr>
<td>TMC_NO_FULL_CORRECTION</td>
<td>The results are not corrected by all active sensors. Angle and distance data are available. This message is to be considered as a warning.</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>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 considered as an info.</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>Angle values okay, but no valid distance. Perform a distance measurement.</td>
</tr>
<tr>
<td>TMC_ANGLE_NO_FULL_CORRECTION</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.

Return Codes

RC_OK

Angle OK

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 aborted.

Example

This example measures the slope distance and angles.

```
DIM Angle AS Double
DIM dSlope AS Double
DIM RetCode AS Integer

TMC_GetSimpleMea( Angle, dSlope, RetCode )
```
6.3.10  TMC_Get/SetAngleFaceDef

Description  Gets and sets the current face definition.

Declaration  
TMC_GetAngleFaceDef( eFaceDef AS Integer )
TMC_SetAngleFaceDef( 
    byVal eFaceDef AS Integer )

Remarks  

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>No distance may exist for setting the face definition. Call TMC_DoMeasure(TMC_CLEAR) before this function.</td>
</tr>
</tbody>
</table>

Parameters  

eFaceDef  out/in  TMC_FACE_NORMAL or TMC_FACE_TURN

See Also  -

Return Codes  

RC_OK  Completed successfully.
TMC_BUSY  measurement system is busy (no valid results) or a distance exists

Example  
The example reads the current definition and sets the opposite one.

    DIM face AS TMC_FACE_DEF
    TMC_GetAngelFaceDef(face)
    IF (face = TMC_FACE_NORMAL) THEN
        TMC_SetAngelFaceDef(TMC_FACE_TURN)
    ELSE
        TMC_SetAngelFaceDef(TMC_FACE_NORMAL)
    END IF
6.3.11 TMC_Get/SetHzOffset

**Description**  
Gets and sets the current horizontal offset.

**Declaration**  
```
TMC_GetHzOffset( dHzOffset AS Double )
TMC_SetHzOffset( byVal dHzOffset AS Double )
```

**Remarks**  
**Note**  
No distance may exist for setting the Hz-offset. Call `TMC_DoMeasure(TMC_CLEAR)` before this function.

**Parameters**  
- `dHzOffset` out/in  
  Horizontal offset in radiant.

**See Also**  
-

**Return Codes**  
- `RC_OK`  
  Completed successfully.
- `TMC_BUSY`  
  Measurement system is busy (no valid results) or a distance exists

**Example**  
The example reads the current offsets and sets it to an increased value.

```
DIM off AS Double
TMC_GetHzOffset( off )
TMC_SetHzOffset( off + 1.0 )
```
6.3.12  TMC_Get/SetDistPpm

Description  Gets and sets the correction values for distance measurements.

Declaration  

\[
\begin{align*}
\text{TMC\_GetDistPpm}\left(\text{PpmCorr AS TMC\_PPM\_CORR\_Type}\right) \\
\text{TMC\_SetDistPpm}\left(\text{PpmCorr AS TMC\_PPM\_CORR\_Type}\right)
\end{align*}
\]

Parameters  

PpmCorr  out/in  Correction value for distance measurement.

Return Codes  

RC_OK  Completed successfully.
TMC_BUSY  TMC is in use and can not be changed.

Example  

-  

6.3.13  TMC_Get/SetHeight

Description  Gets and sets the current height of the reflector.

Declaration  

\[
\begin{align*}
\text{TMC\_GetHeight}\left(\text{Height AS Double}\right) \\
\text{TMC\_SetHeight}\left(\text{byVal Height AS Double}\right)
\end{align*}
\]

Parameters  

Height  out/in  Height of reflector in Meters.

Return Codes  

RC_OK  Completed successfully.
TMC_BUSY  measurement system is busy (no valid results)

Example  

The example sets the reflectors height to the value of 1.0 m.

\[
\text{TMC\_SetHeight}\left(1.0\right)
\]
6.3.14  TMC_Get/SetRefractiveCorr

Description   Gets and sets the refractive correction for measuring the distance.

Declaration   TMC_GetRefractiveCorr (  
               Refraction AS TMC_REFRACTION_Type)  
               TMC_SetRefractiveCorr (  
               Refraction AS TMC_REFRACTION_Type)

Parameters   Refraction out/in    Refraction correction value(s).

Return Codes   
               RC_OK  Completed successfully.
               TMC_BUSY  measurement system is busy (no valid results)

Example   -

6.3.15  TMC_Get/SetRefractiveMethod

Description   Gets and sets the method of refractive correction for measuring the distance.

Declaration   TMC_GetRefractiveMethod (  
               Method AS Integer )  
               TMC_SetRefractiveMethod (  
               byVal Method AS Integer )

Parameters   Method out/in    Method of refraction calculation:
               1: method 1
               2: method 2
               else: undefined

Return Codes   
               RC_OK  Completed successfully.
               TMC_BUSY  measurement system is busy (no valid results)
6.3.16  TMC_Get/SetStation

**Description**  Gets and sets station co-ordinates.

**Declaration**  

```basic
TMC_GetStation ( 
    Station AS TMC_STATION_Type )

TMC_SetStation ( 
    Station AS TMC_STATION_Type )
```

**Remarks**

**Note**  No distance may exist for setting a new station. Call TMC_DoMeasure(TMC_CLEAR) before this function.

**Parameters**

- **Station**  out/in  Station co-ordinates.

**Return Codes**

- **RC_OK**  Completed successfully.
- **TMC_BUSY**  measurement system is busy (no valid results) or a distance exists.

**Example**  -

6.3.17  TMC_IfDistTapeMeasured

**Description**  Gets information about manual measurement.

**Declaration**  

```basic
TMC_IfDistTapeMeasured ( 
    bTapeMeasured AS Logical )
```

**Parameters**

- **bTapeMeasured**  out  
  
  TRUE: if measurement has been done by hand.  
  FALSE: if measurement has been done with EDM or if invalid.

**Return Codes**

- **RC_OK**  Completed successfully.

**Example**  -
6.3.18  TMC_SetHandDist

Description  Sets distance manually.

Declaration  

```
TMC_SetHandDist(
    byVal dSlopeDistance AS Double,
    byVal dHgtOffset AS Double
)
```

Parameters  

- `dSlopeDistance` in slope distance [m]  
- `dHgtOffset` in Height to measured point. [m]

See Also  

- 

Return Codes  

- **RC_OK**: Execution successful.
- **TMC_NO_FULL_CORRECTION**: The results are not corrected by all active sensors. This message is to be considered 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 perform a forced incline measurement or switch off the incline. This message is to be considered 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_IVPARAM
Invalid parameter

Example
-

6.3.19 TMC_SetDistSwitch

Description
Defines the distance measurement correction switches.

Declaration
TMC_SetDistSwitch(
  Switches AS TMC_DIST_SWITCH_Type )

Remarks
This procedure sets the distance measurement correction switches.

Parameters
Switches in Distance switches

Return-Codes
RC_OK Successful termination.

See Also
TMC_GetDistSwitch

6.3.20 TMC_GetDistSwitch

Description
Returns the distance measurement correction switches.

Declaration
TMC_GetDistSwitch(
  Switches AS TMC_DIST_SWITCH_Type )

Remarks
This procedure returns the distance measurement correction switches.

Parameters
Switches out Distance switches
### 6.3.21 TMC_SetOffsetDist

#### Description
Defines the distance measurement offset.

#### Declaration
```
TMC_SetOffsetDist(
    Offsets AS TMC_OFFSET_DIST_Type)
```

#### Remarks
This procedure defines the offset to the prism pole. The `dLengthVal` defines the offset away from the prism pole, positive means in the line from instrument to prism. `dCrossVal` means right from the prism pole and `dHeightVal` means higher than prism pole.

#### Remarks
- **Note**: No distance may exist for offset setting. Call `TMC_DoMeasure(TMC_CLEAR)` before this function.

#### Parameters
- `Offsets` in Target point offset

#### Return-Codes
- `RC_OK` Successful termination.
- `TMC_BUSY` measurement system is busy (no valid results) or a distance exists.

#### See Also
- `TMC_GetOffsetDist`, `BAP_Offset`, `TMC_IsOffsetDistMeasured`
6.3.22 TMC_GetOffsetDist

Description
Returns the distance measurement offset.

Declaration
TMC_GetOffsetDist(
    Offsets AS TMC_OFFSET_DIST_Type )

Remarks
This procedure returns the actual offset to the prism pole. The
dLengthVal defines the offset away from the prism pole,
positive means in the line from instrument to prism. dCrossVal
means right from the prism pole and dHeightVal means higher
than prism pole.

Parameters
    Offsets out Target point offset

Return-Codes
    RC_OK Successful termination.

See Also
    TMC_SetOffsetDist, BAP_Offset,
    TMC_IfOffsetDistMeasured

6.3.23 TMC_IfOffsetDistMeasured

Description
Returns the EDM measurement mode.

Declaration
TMC_IfOffsetDistMeasured(
    lOffset AS Logical )

Remarks
This function returns TRUE if an offset is defined.

Parameters
    lOffset out Offset is valid

Return-Codes
    RC_OK Successful termination.

See Also
    TMC_SetOffsetDist, TMC_GetOffsetDist, BAP_Offset
6.3.24 TMC_GetFace1

Description  Get face information of current telescope position.

Declaration  TMC_GetFace1( lFace1 AS Logical )

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_GetFace1 call). Note that the instrument automatically turns into an inactive measurement state after a predefined timeout.

Parameters

<table>
<thead>
<tr>
<th>lFace1 out</th>
<th>TRUE: Face I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE: Face II</td>
<td></td>
</tr>
</tbody>
</table>

Return-Codes

| RC_OK | Successful termination. |

6.3.25 TMC_SetAngSwitch

Description  Defines the angle measurement correction switches.

Declaration  TMC_SetAngSwitch(
  Switches AS TMC_ANG_SWITCH_Type )

Remarks  This procedure sets the angle measurement correction switches. 

Note  No distance may exist for setting the angle switches. Call TMC_DoMeasure( TMC_CLEAR ) before this function.

Parameters

| Switches in | angular switches |

Return-Codes

| RC_OK | Successful termination. |
| TMC_BUSY | A distance exists |

See Also  TMC_GetAngSwitch
Example  Change switches

```basic
DIM AngSwitches AS TMC_ANG_SWITCH_Type

TMC_DoMeasure(TMC_CLEAR) ' clear distances
TMC_GetAngSwitch(AngSwitches)
AngSwitches.InclineCorr = TRUE
AngSwitches.CollimationCorr = FALSE
TMC_SetAngSwitch(AngSwitches)
```

6.3.26  TMC_GetAngSwitch

<table>
<thead>
<tr>
<th>Description</th>
<th>Returns the angle measurement correction switches.</th>
</tr>
</thead>
</table>
| Declaration | TMC_GetAngSwitch(
|             | Switches AS TMC_ANG_SWITCH_Type ) |
| Remarks     | This procedure returns the actual angle measurement correction switches. |

Parameters

- `Switches in` Angular switches

Return-Codes

- `RC_OK` Successful termination.

See Also

- TMC_SetAngSwitch

6.3.27  TMC_SetInclineSwitch

<table>
<thead>
<tr>
<th>Description</th>
<th>Defines the compensator switch.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declaration</td>
<td>TMC_SetAngSwitches( lOn AS Logical )</td>
</tr>
<tr>
<td>Remarks</td>
<td>This procedure enables or disables the dual axis compensator correction.</td>
</tr>
</tbody>
</table>

Note: No distance may exist for a switch setting. Call `TMC_DoMeasure(TMC_CLEAR)` before this function.

Parameters

- `lOn in Switch`
Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>A distance exists</td>
</tr>
</tbody>
</table>

See Also TMC_GetInclineSwitch

---

6.3.28 TMC_GetInclineSwitch

Description Returns the compensator switch.

Declaration TMC_GetInclineSwitches( lOn AS Logical )

Remarks This procedure returns the dual axis compensator correction state.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lOn</td>
<td>out Switch</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See Also TMC_SetInclineSwitch

---

6.3.29 TMC_GetInclineStatus

Description Returns the inclination compensator status.

Declaration TMC_GetInclineStatus( iStatus AS Integer )

Remarks This procedure returns status of the inclination sensor.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iStatus</td>
<td>out TMC_INC_OFF</td>
</tr>
<tr>
<td></td>
<td>TMC_INC_OK</td>
</tr>
<tr>
<td></td>
<td>Incline-sensor is switched off</td>
</tr>
<tr>
<td></td>
<td>Inclination is ok, recording is allowed</td>
</tr>
<tr>
<td></td>
<td>TMC_INC_TILT</td>
</tr>
<tr>
<td></td>
<td>Incline-sensor is out of working area</td>
</tr>
<tr>
<td></td>
<td>TMC_INC_OLD</td>
</tr>
<tr>
<td></td>
<td>Incline-values are not yet updated</td>
</tr>
</tbody>
</table>

6-128 TPS1100-Version 2.10
Inclination measurement fails

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See Also TMC_SetInclineSwitch

Example See example file „meas.gbs“.
6.4 FUNCTIONS FOR GSI

6.4.1 Summarizing Lists of GSI Types and Procedures

### 6.4.1.1 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wi_List</td>
<td>Array of GSI_WiDlg_Entry_Type.</td>
</tr>
<tr>
<td>GSI_Point_Coord_Type</td>
<td>Point co-ordinate data.</td>
</tr>
<tr>
<td>GSI_Rec_Id_List</td>
<td>Record mask array of integers (indicating WI-identifications)</td>
</tr>
<tr>
<td>GSI_WiDlg_Entry_Type</td>
<td>Dialog entry information.</td>
</tr>
</tbody>
</table>

### 6.4.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_Coding</td>
<td>Starts the active coding function of the TPS system.</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>Returns if distance tracking is running.</td>
</tr>
<tr>
<td>GSI_CreateMDlg</td>
<td>Creates and shows the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_DefineMDlg</td>
<td>Defines the entries of the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_DefineRecMaskDlg</td>
<td>Defines the recording mask dialog.</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>Executes an automatic distance measurement.</td>
</tr>
<tr>
<td>GSI_ExecQCoding</td>
<td>Executes the Quick-Coding.</td>
</tr>
<tr>
<td>GSI_GetDataPath</td>
<td>Get the name of the file with the import data.</td>
</tr>
<tr>
<td>GSI_GetIndivNr</td>
<td>Fetches the individual point number.</td>
</tr>
<tr>
<td>GSI_GetLineSysMDlg</td>
<td>Gets the definition of a line in the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>Returns the number of the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>This routine returns the status for Quick-</td>
</tr>
</tbody>
</table>

TPS1100-Version 2.10
<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_GetRecMask</td>
<td>Coding. Get the definition and the format of a recording mask.</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>Returns the used recording mask.</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>Returns the recording order for Quick-Coding.</td>
</tr>
<tr>
<td>GSI_GetRecPath</td>
<td>Returns the recording path.</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>Fetches the running point number and the increment.</td>
</tr>
<tr>
<td>GSI_GetWiEntry</td>
<td>Get data from the Theodolite data pool.</td>
</tr>
<tr>
<td>GSI_ImportCoordDlg</td>
<td>Show the co-ordinate import dialog.</td>
</tr>
<tr>
<td>GSI_IncPNumber</td>
<td>Automatically point number increment.</td>
</tr>
<tr>
<td>GSI_IsRunningNr</td>
<td>Queries if running number is being used.</td>
</tr>
<tr>
<td>GSI_ManCoordDlg</td>
<td>Show the manual co-ordinate input dialog.</td>
</tr>
<tr>
<td>GSI_Measure</td>
<td>Entry point for measure and registration dialog (measure and registration).</td>
</tr>
<tr>
<td>GSI_QuickSet</td>
<td>Show the Quickset dialog.</td>
</tr>
<tr>
<td>GSI_RecordRecMask</td>
<td>Recording the given wi mask.</td>
</tr>
<tr>
<td>GSI_SelectCode</td>
<td>This routine shows the codelist-coding dialog.</td>
</tr>
<tr>
<td>GSI_SetDataPath</td>
<td>Set the file with the import data.</td>
</tr>
<tr>
<td>GSI_SetIndivNr</td>
<td>Sets the individual point number.</td>
</tr>
<tr>
<td>GSI_SetIvPtNrStatus</td>
<td>Switches the individual point number mode on/off.</td>
</tr>
<tr>
<td>GSI_SetLineMDlg</td>
<td>Sets one line in the user definable measurement dialog to system parameter.</td>
</tr>
<tr>
<td>GSI_SetLineMDlgPar</td>
<td>Sets a line in the user definable measurement dialog to an application parameter.</td>
</tr>
<tr>
<td>GSI_SetLineMDlgText</td>
<td>Puts a textline into the user definable measurement dialog.</td>
</tr>
<tr>
<td>GSI_SetLineSysMDlg</td>
<td>Sets a line in the system measurement dialog.</td>
</tr>
<tr>
<td>GSI_SetMDlgNr</td>
<td>Sets the number of the system measurement dialog.</td>
</tr>
</tbody>
</table>
### 6. System Functions

#### 6.4.2 Constants for WI values

Definitions for WI values:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_PTNR</td>
<td>String</td>
<td>Point number</td>
</tr>
<tr>
<td>GSI_ID_FNR</td>
<td>Double</td>
<td>Serial number</td>
</tr>
<tr>
<td>GSI_ID_TYPE</td>
<td>String</td>
<td>Device type</td>
</tr>
<tr>
<td>GSI_ID_TIME_1</td>
<td>String</td>
<td>First time art</td>
</tr>
<tr>
<td>GSI_ID_TIME_2</td>
<td>String</td>
<td>Second time art</td>
</tr>
<tr>
<td>GSI_ID_HZ</td>
<td>Double</td>
<td>Horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_V</td>
<td>Double</td>
<td>Vertical angle</td>
</tr>
<tr>
<td>GSI_ID_NHZ</td>
<td>Double</td>
<td>Nominal horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_DHZ</td>
<td>Double</td>
<td>Difference horizontal angle</td>
</tr>
<tr>
<td>GSI_ID_NV</td>
<td>Double</td>
<td>Nominal vertical angle</td>
</tr>
<tr>
<td>GSI_ID_DV</td>
<td>Double</td>
<td>Difference vertical angle</td>
</tr>
<tr>
<td>GSI_ID_SLOPE</td>
<td>Double</td>
<td>Slope distance</td>
</tr>
<tr>
<td>Name</td>
<td>Data Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>GSI_ID_HOR</td>
<td>Double</td>
<td>Horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_HGT</td>
<td>Double</td>
<td>Height difference</td>
</tr>
<tr>
<td>GSI_ID_NHOR</td>
<td>Double</td>
<td>Nominal horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_DHOR</td>
<td>Double</td>
<td>Difference horizontal distance</td>
</tr>
<tr>
<td>GSI_ID_NHGT</td>
<td>Double</td>
<td>Nominal height difference</td>
</tr>
<tr>
<td>GSI_ID_DHGT</td>
<td>Double</td>
<td>Difference height difference</td>
</tr>
<tr>
<td>GSI_ID_NSLOPE</td>
<td>Double</td>
<td>Nominal slope distance</td>
</tr>
<tr>
<td>GSI_ID_DSLOPE</td>
<td>Double</td>
<td>Difference slope distance</td>
</tr>
<tr>
<td>GSI_ID_CODE</td>
<td>String</td>
<td>Code information</td>
</tr>
<tr>
<td>GSI_ID_CODE_1</td>
<td>String</td>
<td>Information 1</td>
</tr>
<tr>
<td>GSI_ID_CODE_2</td>
<td>String</td>
<td>Information 2</td>
</tr>
<tr>
<td>GSI_ID_CODE_3</td>
<td>String</td>
<td>Information 3</td>
</tr>
<tr>
<td>GSI_ID_CODE_4</td>
<td>String</td>
<td>Information 4</td>
</tr>
<tr>
<td>GSI_ID_CODE_5</td>
<td>String</td>
<td>Information 5</td>
</tr>
<tr>
<td>GSI_ID_CODE_6</td>
<td>String</td>
<td>Information 6</td>
</tr>
<tr>
<td>GSI_ID_CODE_7</td>
<td>String</td>
<td>Information 7</td>
</tr>
<tr>
<td>GSI_ID_CODE_8</td>
<td>String</td>
<td>Information 8</td>
</tr>
<tr>
<td>GSI_ID_PPMM</td>
<td>String</td>
<td>mm and ppm</td>
</tr>
<tr>
<td>GSI_ID_SIGMA</td>
<td>String</td>
<td>Distance count and deviation</td>
</tr>
<tr>
<td>GSI_ID_MM</td>
<td>Double</td>
<td>mm</td>
</tr>
<tr>
<td>GSI_ID_PPM</td>
<td>Double</td>
<td>ppm</td>
</tr>
<tr>
<td>GSI_ID_REM_1</td>
<td>String</td>
<td>Remark 1</td>
</tr>
<tr>
<td>GSI_ID_REM_2</td>
<td>String</td>
<td>Remark 2</td>
</tr>
<tr>
<td>GSI_ID_REM_3</td>
<td>String</td>
<td>Remark 3</td>
</tr>
<tr>
<td>GSI_ID_REM_4</td>
<td>String</td>
<td>Remark 4</td>
</tr>
<tr>
<td>GSI_ID_REM_5</td>
<td>String</td>
<td>Remark 5</td>
</tr>
<tr>
<td>GSI_ID_REM_6</td>
<td>String</td>
<td>Remark 6</td>
</tr>
<tr>
<td>GSI_ID_REM_7</td>
<td>String</td>
<td>Remark 7</td>
</tr>
<tr>
<td>GSI_ID_REM_8</td>
<td>String</td>
<td>Remark 8</td>
</tr>
<tr>
<td>GSI_ID_REM_9</td>
<td>String</td>
<td>Remark 9</td>
</tr>
<tr>
<td>GSI_ID_E</td>
<td>Double</td>
<td>East co-ordinate</td>
</tr>
<tr>
<td>Name</td>
<td>Data Type</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>GSI_ID_N</td>
<td>Double</td>
<td>North co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_H</td>
<td>Double</td>
<td>Height</td>
</tr>
<tr>
<td>GSI_ID_E0</td>
<td>Double</td>
<td>East station co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_N0</td>
<td>Double</td>
<td>North station co-ordinate</td>
</tr>
<tr>
<td>GSI_ID_H0</td>
<td>Double</td>
<td>Station height</td>
</tr>
<tr>
<td>GSI_ID_HR</td>
<td>Double</td>
<td>Reflector height</td>
</tr>
<tr>
<td>GSI_ID_HI</td>
<td>Double</td>
<td>Instrument height</td>
</tr>
<tr>
<td>GSI_ID_INDIV</td>
<td>String</td>
<td>Individual point number</td>
</tr>
<tr>
<td>GSI_ID_PTLA</td>
<td>String</td>
<td>Number of the last recorded point</td>
</tr>
<tr>
<td>GSI_ID_STEP</td>
<td>Double</td>
<td>Increment of the running point number</td>
</tr>
<tr>
<td>GSI_ID_SPTNR</td>
<td>String</td>
<td>Station point number</td>
</tr>
<tr>
<td>GSI_ID_SHZ</td>
<td>Double</td>
<td>Hz angle with no sign change</td>
</tr>
<tr>
<td>GSI_ID_CD_DSC</td>
<td>String</td>
<td>Code description</td>
</tr>
<tr>
<td>GSI_ID_PTCD_DSC</td>
<td>String</td>
<td>Point code description</td>
</tr>
<tr>
<td>GSI_ID_PV_CD</td>
<td>String</td>
<td>Preview code</td>
</tr>
<tr>
<td>GSI_ID_PV_PTCD</td>
<td>String</td>
<td>Preview point code</td>
</tr>
<tr>
<td>GSI_ID_ACT_PTID</td>
<td>String</td>
<td>Actual point ID</td>
</tr>
<tr>
<td>GSI_ID_BACKID</td>
<td>String</td>
<td>Backside ID</td>
</tr>
<tr>
<td>GSI_ID_APPDATA0</td>
<td>String/Double</td>
<td>Application data 0</td>
</tr>
<tr>
<td>GSI_ID_APPDATA1</td>
<td>String/Double</td>
<td>Application data 1</td>
</tr>
<tr>
<td>GSI_ID_APPDATA2</td>
<td>String/Double</td>
<td>Application data 2</td>
</tr>
<tr>
<td>GSI_ID_APPDATA3</td>
<td>String/Double</td>
<td>Application data 3</td>
</tr>
<tr>
<td>GSI_ID_APPDATA4</td>
<td>String/Double</td>
<td>Application data 4</td>
</tr>
<tr>
<td>GSI_ID_APPDATA5</td>
<td>String/Double</td>
<td>Application data 5</td>
</tr>
<tr>
<td>GSI_ID_APPDATA6</td>
<td>String/Double</td>
<td>Application data 6</td>
</tr>
<tr>
<td>GSI_ID_APPDATA7</td>
<td>String/Double</td>
<td>Application data 7</td>
</tr>
<tr>
<td>GSI_ID_APPDATA8</td>
<td>String/Double</td>
<td>Application data 8</td>
</tr>
<tr>
<td>GSI_ID_APPDATA9</td>
<td>String/Double</td>
<td>Application data 9</td>
</tr>
<tr>
<td>GSI_ID_APPDATA10</td>
<td>String/Double</td>
<td>Application data 10</td>
</tr>
<tr>
<td>GSI_ID_APPDATA11</td>
<td>String/Double</td>
<td>Application data 11</td>
</tr>
<tr>
<td>GSI_ID_FS_SCALE</td>
<td>Double</td>
<td>Free station scale</td>
</tr>
</tbody>
</table>
### 6.4.3 Constants for Measurement Dialog Definition

Definition of (user definable) application parameters for measurement dialogs, either Double or String. See also GSI_SetLineMDlgPar and GSI_SetLineMDlgText.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_AppData0</td>
<td>Application parameter 0</td>
</tr>
<tr>
<td>GSI_PAR_AppData1</td>
<td>Application parameter 1</td>
</tr>
<tr>
<td>GSI_PAR_AppData2</td>
<td>Application parameter 2</td>
</tr>
<tr>
<td>GSI_PAR_AppData3</td>
<td>Application parameter 3</td>
</tr>
<tr>
<td>GSI_PAR_AppData4</td>
<td>Application parameter 4</td>
</tr>
<tr>
<td>GSI_PAR_AppData5</td>
<td>Application parameter 5</td>
</tr>
<tr>
<td>GSI_PAR_AppData6</td>
<td>Application parameter 6</td>
</tr>
<tr>
<td>GSI_PAR_AppData7</td>
<td>Application parameter 7</td>
</tr>
<tr>
<td>GSI_PAR_AppData8</td>
<td>Application parameter 8</td>
</tr>
<tr>
<td>GSI_PAR_AppData9</td>
<td>Application parameter 9</td>
</tr>
<tr>
<td>GSI_PAR_AppData10</td>
<td>Application parameter 10</td>
</tr>
<tr>
<td>Name</td>
<td>Meaning</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_PAR_AppData11</td>
<td>Application parameter 11</td>
</tr>
</tbody>
</table>

Definition of system (defined) parameters for measurement dialogs. See also GSI_SetLineSysMDlg and GSI_SetLineMDlg.

<table>
<thead>
<tr>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_PAR_AddConst</td>
<td>Prism constant</td>
</tr>
<tr>
<td>GSI_PAR_Attrib1</td>
<td>Point Code Attribute 1</td>
</tr>
<tr>
<td>GSI_PAR_Attrib2</td>
<td>Point Code Attribute 2</td>
</tr>
<tr>
<td>GSI_PAR_Attrib3</td>
<td>Point Code Attribute 3</td>
</tr>
<tr>
<td>GSI_PAR_Attrib4</td>
<td>Point Code Attribute 4</td>
</tr>
<tr>
<td>GSI_PAR_Attrib5</td>
<td>Point Code Attribute 5</td>
</tr>
<tr>
<td>GSI_PAR_Attrib6</td>
<td>Point Code Attribute 6</td>
</tr>
<tr>
<td>GSI_PAR_Attrib7</td>
<td>Point Code Attribute 7</td>
</tr>
<tr>
<td>GSI_PAR_Attrib8</td>
<td>Point Code Attribute 8</td>
</tr>
<tr>
<td>GSI_PAR_AvgMeasNo</td>
<td>Maximal number of distance measurements</td>
</tr>
<tr>
<td></td>
<td>of the average mode</td>
</tr>
<tr>
<td>GSI_PAR_BacksideId</td>
<td>Last used Backside</td>
</tr>
<tr>
<td>GSI_PAR_Code</td>
<td>Last used Code</td>
</tr>
<tr>
<td>GSI_PAR_CodeDescr</td>
<td>Last used free Code Description</td>
</tr>
<tr>
<td>GSI_PAR_CodeList</td>
<td>Codelist management (select, create etc)</td>
</tr>
<tr>
<td>GSI_PAR_CodeListSelect</td>
<td>Codelist selection (of an existing codelist)</td>
</tr>
<tr>
<td>GSI_PAR_DataJobSelect</td>
<td>Data job selection (of an existing job)</td>
</tr>
<tr>
<td>GSI_PAR_Date</td>
<td>Current date of the instrument. The displayed</td>
</tr>
<tr>
<td></td>
<td>format depends on the setting of the</td>
</tr>
<tr>
<td></td>
<td>parameter &quot;Date form.&quot;</td>
</tr>
<tr>
<td>GSI_PAR_DisplayMask</td>
<td>Select display mask for standard measuring</td>
</tr>
<tr>
<td></td>
<td>dialog. Max. 3 displaymasks can be defined</td>
</tr>
<tr>
<td></td>
<td>for this dialog. The displaymasks can also</td>
</tr>
<tr>
<td></td>
<td>be changed with the system function &quot;Next</td>
</tr>
<tr>
<td></td>
<td>Displaymask&quot;.</td>
</tr>
<tr>
<td>GSI_PAR_DataJob</td>
<td>Data job management (select, create etc)</td>
</tr>
<tr>
<td>GSI_PAR_TargetEast</td>
<td>Target point Easting</td>
</tr>
<tr>
<td>GSI_PAR_DistMeasProg</td>
<td>EDM measurement program selection.</td>
</tr>
<tr>
<td>Name</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_PAR_TargetElev</td>
<td>Target point Elevation</td>
</tr>
<tr>
<td>GSI_PAR_ElevDiff</td>
<td>Elevation difference</td>
</tr>
<tr>
<td>GSI_PAR_HalfLineSpace</td>
<td>This item can be used to display a half line space in order to separate or group lines on instrument screen.</td>
</tr>
<tr>
<td>GSI_PAR_DistHoriz</td>
<td>Horizontal distance</td>
</tr>
<tr>
<td>GSI_PAR_AngleHz</td>
<td>Hz-Angle</td>
</tr>
<tr>
<td>GSI_PAR_PointIdIncr</td>
<td>defines the increment step. It is used to increment the Target Point Id after recording a target point.</td>
</tr>
<tr>
<td>GSI_PAR_IndivPointId</td>
<td>Individual point identifier</td>
</tr>
<tr>
<td>GSI_PAR_Info1</td>
<td>Shows the Free Code Info 1</td>
</tr>
<tr>
<td>GSI_PAR_Info2</td>
<td>Shows the Free Code Info 2</td>
</tr>
<tr>
<td>GSI_PAR_Info3</td>
<td>Shows the Free Code Info 3</td>
</tr>
<tr>
<td>GSI_PAR_Info4</td>
<td>Shows the Free Code Info 4</td>
</tr>
<tr>
<td>GSI_PAR_Info5</td>
<td>Shows the Free Code Info 5</td>
</tr>
<tr>
<td>GSI_PAR_Info6</td>
<td>Shows the Free Code Info 6</td>
</tr>
<tr>
<td>GSI_PAR_Info7</td>
<td>Shows the Free Code Info 7</td>
</tr>
<tr>
<td>GSI_PAR_Info8</td>
<td>Shows the Free Code Info 8</td>
</tr>
<tr>
<td>GSI_PAR_InstrHeight</td>
<td>Instrument Height (hi)</td>
</tr>
<tr>
<td>GSI_PAR_LastPointId</td>
<td>Last recorded target point identifier</td>
</tr>
<tr>
<td>GSI_PAR_MeasJobSelect</td>
<td>Measurement Job selection (of an existing Job or RS232 for online recording)</td>
</tr>
<tr>
<td>GSI_PAR_MeasJob</td>
<td>Measurement Job management (select, create, etc.)</td>
</tr>
<tr>
<td>GSI_PAR_NS</td>
<td>Number of measurements and standard deviation</td>
</tr>
<tr>
<td>GSI_PAR_TargetNorth</td>
<td>Target point Northing</td>
</tr>
<tr>
<td>GSI_PAR_OffsetCross</td>
<td>Cross Offset</td>
</tr>
<tr>
<td>GSI_PAR_OffsetElev</td>
<td>Offset Elevation</td>
</tr>
<tr>
<td>Name</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GSI_PAR_OffsetLength</td>
<td>Offset Length</td>
</tr>
<tr>
<td>GSI_PAR_OffsetMode</td>
<td>Defines the resetting of the offset</td>
</tr>
<tr>
<td>GSI_PAR_PointCode</td>
<td>Actual Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_PointId</td>
<td>Actual Target point identifier, running or individual. The Value and the display text changes if an individual number is set.</td>
</tr>
<tr>
<td>GSI_PAR_PpmAtm</td>
<td>ppm atmospheric</td>
</tr>
<tr>
<td>GSI_PAR_PpmGeom</td>
<td>ppm geometric</td>
</tr>
<tr>
<td>GSI_PAR_PpmTotal</td>
<td>Total ppm</td>
</tr>
<tr>
<td>GSI_PAR_PpmMm</td>
<td>Total ppm and prism constant</td>
</tr>
<tr>
<td>GSI_PAR_PrevCode</td>
<td>Shows the second last used Code</td>
</tr>
<tr>
<td>GSI_PAR_PrevPointCode</td>
<td>Last used Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_PointCodeDescr</td>
<td>Shows the Point Code Description of the actual Feature Code</td>
</tr>
<tr>
<td>GSI_PAR_RecMask</td>
<td>Selected Recording mask for target point measurements</td>
</tr>
<tr>
<td>GSI_PAR_ReflHeight</td>
<td>Reflector height (hr)</td>
</tr>
<tr>
<td>GSI_PAR_ReflName</td>
<td>Used reflector type</td>
</tr>
<tr>
<td>GSI_PAR_ReflSelection</td>
<td>reflector type selection. If there are user defined prism, then they will be added to this list. The User Refl1..User Refl3 are only valid, if these user definable prisms are defined.</td>
</tr>
<tr>
<td>GSI_PAR_RunningPointId</td>
<td>Running target point identifier</td>
</tr>
<tr>
<td>GSI_PAR_DistSlope</td>
<td>Slope distance</td>
</tr>
<tr>
<td>GSI_PAR_StationId</td>
<td>Identifies the Station</td>
</tr>
<tr>
<td>GSI_PAR_StationEast</td>
<td>Station Easting</td>
</tr>
<tr>
<td>GSI_PAR_StationElev</td>
<td>Station Elevation</td>
</tr>
<tr>
<td>GSI_PAR_StationNorth</td>
<td>Station Northing</td>
</tr>
<tr>
<td>GSI_PAR_TargetType</td>
<td>Definition of the target type (Reflector / reflectorless)</td>
</tr>
<tr>
<td>GSI_PAR_Time</td>
<td>Current time of the instrument. The displayed format depends on the setting of the parameter &quot;Time form.&quot;</td>
</tr>
</tbody>
</table>
Name: GSI_PAR_AngleV
Meaning: V-Angle

V-Angle display format: Zenith angle = 0gon for zenith, angles are positive. Elev. angle = 0gon for horizontal, (+) above horizon and (-) below horizon. Elev. angle% = 0% for horizon, 100% for 50gon. V-angle is displayed (+) above and (-) below horizon but as percentage of the gradient.

Name: GSI_PAR_VangleFormat
Meaning: Designates a line that is unused.

6.4.4 Relationship of GSI_ID’s to GSI_PAR’s

In general we can distinguish between two data value pools who are able to store values in it. Some of these values are shared between the two pools. GSI_ID_-Ids describe the values which can be stored and requested in the (W1) data value pool. GSI_PAR_-Ids describe the values which can be used for displaying in a measurement dialog. Their sets of Id’s are not associated directly in all cases. Moreover their sets of Id’s can be distinguished in their meaning. Association in this context means that both pools, the data value pool and the data display pool, share their values directly. Nonassociated values are unique to either the data value pool or the data display pool.

Many of the GSI_IDs are record-able. Two types of record-able Ids can be distinguished:

a) Measurement block (“Meas”) (has to start with a GSI_ID_PTNR)
b) Code block (“Code”) (has to start with a GSI_ID_CODE)

They may not be mixed.

<table>
<thead>
<tr>
<th>Record-able</th>
<th>GSI_ID_-Ids</th>
<th>GSI_PAR_-Ids</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GSI_ID_NHZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSI_ID_DHZ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GSI_ID_NV</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_DV</td>
<td>GSI_PAR_IndivPointId</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_NHOR</td>
<td>GSI_PAR_LastPointId</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_DHOR</td>
<td>GSI_PAR_PointIdIncr</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_NMGT</td>
<td>GSI_PAR_StationId</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_DHGT</td>
<td>GSI_PAR_BackSideId</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_NSLOPE</td>
<td>GSI_PAR_PointId</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_DDSLOPE</td>
<td>GSI_PAR_SerialNr</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_INDIV</td>
<td>GSI_PAR_RunningPointId</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PTLA</td>
<td>GSI_PAR_CodeDescr</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_STEP</td>
<td>GSI_PAR_PointCodeDescr</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_SPTNR</td>
<td>GSI_PAR_InstrType</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_SHZ</td>
<td>GSI_PAR_DistSlope</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_CD_DSC</td>
<td>GSI_PAR_Date</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PTCD_DSC</td>
<td>GSI_PAR_Time</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PV_CD</td>
<td>GSI_PAR_AngleHz</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_PV_PTCD</td>
<td>GSI_PAR_AngleV</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_ACT_PTID</td>
<td>GSI_PAR_ElevDiff</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_BACKID</td>
<td>GSI_PAR_PpmMm</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_FNR</td>
<td>GSI_PAR_PpmMm</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_TYPE</td>
<td>GSI_PAR_NS</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_TIME_1</td>
<td>(undefined)</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_TIME_2</td>
<td>(undefined)</td>
<td></td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_MM</td>
<td>GSI_PAR_AddConst</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_PPM</td>
<td>GSI_PAR_PpmTotal</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_1</td>
<td>GSI_PAR_Info1</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_2</td>
<td>GSI_PAR_Info2</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_3</td>
<td>GSI_PAR_Info3</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_4</td>
<td>GSI_PAR_Info4</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_5</td>
<td>GSI_PAR_Info5</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_6</td>
<td>GSI_PAR_Info6</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_7</td>
<td>GSI_PAR_Info7</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_8</td>
<td>GSI_PAR_Info8</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_REM_9</td>
<td>GSI_PAR_Info9</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_E</td>
<td>GSI_PAR_TargetEast</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_N</td>
<td>GSI_PAR_TargetNorth</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_H</td>
<td>GSI_PAR_TargetElev</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_E0</td>
<td>GSI_PAR_StationEast</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_N0</td>
<td>GSI_PAR_StationNorth</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_H0</td>
<td>GSI_PAR_StationElev</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HR</td>
<td>GSI_PAR_ReflHeight</td>
</tr>
<tr>
<td>Meas</td>
<td>GSI_ID_HI</td>
<td>GSI_PAR_InstrHeight</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE</td>
<td>GSI_PAR_Attrib1</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_1</td>
<td>GSI_PAR_Attrib2</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_2</td>
<td>GSI_PAR_Attrib3</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_3</td>
<td>GSI_PAR_Attrib4</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_4</td>
<td>GSI_PAR_Attrib5</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_5</td>
<td>GSI_PAR_Attrib6</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_6</td>
<td>GSI_PAR_Attrib7</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_7</td>
<td>GSI_PAR_Attrib8</td>
</tr>
<tr>
<td>Code</td>
<td>GSI_ID_CODE_8</td>
<td>GSI_PAR_Attrib9</td>
</tr>
</tbody>
</table>

GSI_ID_APPDATA0 are for the purpose of exchanging data between applications and between application and MDlg. They cannot be recorded. Both can be of the form GSI_ASCII or GSI_DOUBLE.
<table>
<thead>
<tr>
<th>GSI_ID_APPDATA0</th>
<th>GSI_PAR_APPDATA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_APPDATA1</td>
<td>GSI_PAR_APPDATA1</td>
</tr>
<tr>
<td>GSI_ID_APPDATA2</td>
<td>GSI_PAR_APPDATA2</td>
</tr>
<tr>
<td>GSI_ID_APPDATA3</td>
<td>GSI_PAR_APPDATA3</td>
</tr>
<tr>
<td>GSI_ID_APPDATA4</td>
<td>GSI_PAR_APPDATA4</td>
</tr>
<tr>
<td>GSI_ID_APPDATA5</td>
<td>GSI_PAR_APPDATA5</td>
</tr>
<tr>
<td>GSI_ID_APPDATA6</td>
<td>GSI_PAR_APPDATA6</td>
</tr>
<tr>
<td>GSI_ID_APPDATA7</td>
<td>GSI_PAR_APPDATA7</td>
</tr>
<tr>
<td>GSI_ID_APPDATA8</td>
<td>GSI_PAR_APPDATA8</td>
</tr>
<tr>
<td>GSI_ID_APPDATA9</td>
<td>GSI_PAR_APPDATA9</td>
</tr>
<tr>
<td>GSI_ID_APPDATA10</td>
<td>GSI_PAR_APPDATA10</td>
</tr>
<tr>
<td>GSI_ID_APPDATA11</td>
<td>GSI_PAR_APPDATA11</td>
</tr>
</tbody>
</table>

Special Ids

<table>
<thead>
<tr>
<th>GSI_ID_NONE</th>
<th>GSI_PAR_NONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ID_EMPTY</td>
<td></td>
</tr>
<tr>
<td>GSI_ID_UNKNOWN</td>
<td></td>
</tr>
</tbody>
</table>

The set of GSI_PAR-ids is not complete in this table. There exist several more Ids, which can be used for displaying.

6.4.5 Data Structures for GSI Functions

**GSI_WiDlg_Entry_Type**: Dialog entry information

**Description**: This data structure is used to store information about the entries (data fields) of the WI dialog.

```plaintext
TYPE GSI_WiDlg_Entry_Type
iId AS Integer
    The identifier of the dialog entry. For possible value see WI constants.
```

6-142
The type of the date stored in dValue or sValue. For possible value see table below.

<table>
<thead>
<tr>
<th>iDataType</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ASCII</td>
<td>ASCII data (stored in sValue)</td>
</tr>
<tr>
<td>GSI_ASCII_SIGN</td>
<td>Signed ASCII data (stored in sValue)</td>
</tr>
<tr>
<td>GSI_DOUBLE</td>
<td>Double data (stored in dValue)</td>
</tr>
</tbody>
</table>

TRUE if the value is valid.

Data if value is of type Double.

Data if value is of type String.

Wi_List: An array of GSI_WiDlg_Entry_Type

Description: This array consists of GSI_MAX_REC_WI elements of the type GSI_WiDlg_Entry_Type.

GSI_Rec_Id_List: An array of integers (indicating WI-identifications)

Description: This array consists of GSI_MAX_REC_WI elements of the type Integer. It is used to define the recorded values (recmask).

GSI_Point_Coord_Type: Point co-ordinate data

Description: This data structure is used to store a point name and its co-ordinates.

<table>
<thead>
<tr>
<th>Description</th>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>sPtNr Valid</td>
<td>String10</td>
<td>TRUE if point number is valid</td>
</tr>
<tr>
<td>dEast Valid</td>
<td>Double</td>
<td>TRUE if east co-ordinate is valid</td>
</tr>
<tr>
<td>dNorth Valid</td>
<td>Double</td>
<td>TRUE if north co-ordinate is valid</td>
</tr>
<tr>
<td>dHeight Valid</td>
<td>Double</td>
<td>TRUE if height co-ordinate is valid</td>
</tr>
<tr>
<td>lPtNrValid</td>
<td>Logical</td>
<td>TRUE if point number is valid</td>
</tr>
<tr>
<td>lEValid</td>
<td>Logical</td>
<td>TRUE if east co-ordinate is valid</td>
</tr>
<tr>
<td>lNValid</td>
<td>Logical</td>
<td>TRUE if north co-ordinate is valid</td>
</tr>
</tbody>
</table>
lHValid AS Logical TRUE if height coordinate is valid

END GSI_Point_Coord_Type

6.4.6 GSI_GetRunningNr

Description
Fetched the running point number and the increment.

Declaration
GSI_GetRunningNr( sPntId AS String20, sPntIncr AS String20 )

Remarks
Fetched the running point number and increment for it.

Parameters
sPntId out the running point number
sPntIncr out the increment for the running point number

See Also
GSI_SetRunningNr, GSI_GetIndivNr,
GSI_SetIndivNr, GSI_IsRunningNr

Return-Codes
RC_OK successful

Example
DIM sPntId AS String20
DIM sPntIncr AS String20
GSI_GetRunningNr( sPntId, sPntIncr )
6.4.7 GSI_SetRunningNr

**Description**
Sets the running point number and increment.

**Declaration**

```basic
GSI_SetRunningNr( 
    BYVAL sPntId AS String20, 
    BYVAL sPntIncr AS String20 )
```

**Remarks**
Sets the running point number and the increment for it. The running point number mode is switched on.

**Parameters**

- `sPntId` **in** The user running point number.
- `sPntIncr` **in** The increment for the user point running number.

**See Also**
GSI_GetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr, GSI_IsRunningNr

**Return-Codes**

- `RC_OK` successful

**Example**

```basic
DIM sPntId AS String20
DIM sPntInc AS String20
GSI_SetRunningNr( sPntId, sPntInc )
```

6.4.8 GSI_GetIndivNr

**Description**
Fetches the individual point number.

**Declaration**

```basic
GSI_GetIndivNr( sPntId AS String20 )
```

**Remarks**
Fetches the individual point number.

**Parameters**

- `sPntId` **out** The user-defined individual point number.

**See Also**
GSI_GetRunningNr, GSI_SetRunningNr, GSI_SetIndivNr, GSI_IsRunningNr

**Return-Codes**

- `RC_OK` successful

**Example**

```basic
DIM sPntId AS String20
DIM sPntInc AS String20
GSI_SetRunningNr( sPntId, sPntInc )
```
### 6.4.9 GSI_SetIndivNr

**Description**: Sets the individual point number.

**Declaration**: 
```
GSI_SetIndivNr( BYVAL sPntId AS String20 )
```

**Remarks**: Sets the individual point number. After this call, the running point number mode is switched to the individual point number. This mode will be active until replaced by a running number or until the next save.

**Parameters**
- `sPntId` in: The user-defined individual point number.

**See Also**
- `GSI_GetRunningNr`, `GSI_SetRunningNr`, `GSI_GetIndivNr`, `GSI_IsRunningNr`

**Return-Codes**
- `RC_OK` successful

**Example**
```
DIM sPntId AS String20
GSI_SetIndivNr( sPntId )
```

### 6.4.10 GSI_IsRunningNr

**Description**: Queries if running number is being used.

**Declaration**: 
```
GSI_IsRunningNr( lRunningOn AS Logical )
```

**Remarks**: If the running number is active the parameter will forced to TRUE otherwise to FALSE.
6. System Functions

### Parameters

```plaintext
lRunningOn  out  information about the running point number
```

### See Also

GSI_GetRunningNr, GSI_SetRunningNr,
GSI_GetIndivNr, GSI_SetIndivNr

### Return-Codes

- **RC_OK**  successful

### Example

```plaintext
DIM lRunningOn AS Logical
GSI_IsRunningNr( lRunningOn )
```

---

### 6.4.11 GSI_SetIvPtNrStatus

#### Description

Switches the individual point number mode on/off.

#### Declaration

```plaintext
GSI_SetIvPtNrStatus(  
  BYVAL lSwitch AS Logical )
```

#### Remarks

Switch the individual point number on or off. When point number is shown in the display the number will change.

#### Parameters

- **lSwitch**  in  switch for the individual point-number  
  (TRUE = on, FALSE = off)

#### See Also

GSI_GetRunningNr, GSI_SetRunningNr,
GSI_GetIndivNr, GSI_SetIndivNr,
GSI_IsRunningNr

#### Return-Codes

- **RC_OK**  successful

#### Example

```plaintext
GSI_SetIvPtNrStatus( FALSE )
```
6.4.12 GSI_IncPNumber

Description: Automatically point number increment.

Declaration: GSI_IncPNumber()

Remarks: This function increments the running alphanumeric point number.

Parameters: none

See Also: GSI_GetRunningNr, GSI_SetRunningNr, GSI_GetIndivNr, GSI_SetIndivNr

Return Codes:

RC_IVRESULT: Point number is not incremented, possible reasons could be:
- wrong alphanumeric chars in point number
- alphanumeric chars in step
- overflow on a alphanumeric char step
- step is longer as the point number

Example:

GSI_IncPNumber()

6.4.13 GSI_Coding

Description: Starts the active coding function of the TPS system.

Declaration: GSI_Coding( BYVAL Caption AS _Token)

Remarks: This routine starts the active coding function of the TPS system. Since there exist three possible locations, the TPS system follows a default ordering rule to invoke one of the programs. First it checks if there is an appropriate set up GeoBASIC coding program. If yes it will be executed, otherwise it examines the codelist management if a codelist is selected. If yes then the codelist will be opened, otherwise the standard coding will be activated.

Parameters:

Caption: The left caption string of the dialog.
6.4.14 GSI_SelectCode

**Description**
This routine shows the codelist-coding dialog.

**Declaration**

```plaintext
GSI_SelectCode( BYVAL Caption AS _Token)
```

**Remarks**
This routine starts the codelist-coding function of the TPS system. It will be executed only if a valid codelist is selected.

**Parameters**

- **Caption** in The left caption string of the dialog.

**Return-Codes**

- **RC_OK** successful
- **RC_ABORT** Coding was aborted by pressing of the ESC-button
- **RC_ABORT_APPL** Coding was aborted by pressing of the QUIT-button
- **COD_RC_LIST_NOT_VALID** No valid codelist selected

**Example**
See example file "meas.gbs".

6.4.15 GSI_GetQCodeAvailable

**Description**
This routine returns the status for Quick-Coding.

**Declaration**

```plaintext
GSI_GetQCodeAvailable(lAvailable As Logical, lEnabled As Logical)
```
Remarks
This routine returns if a valid codelist is selected and if Quick-Coding is enabled or not.

Parameters

lAvailable out TRUE: a valid codelist is selected.
lEnabled out TRUE: Quick-Coding is activated

See Also
GSI_SetQCodeMode, GSI_ExecQCoding

Return-Codes

RC_OK successful

Example
See example file „meas_od.gbs“.

6.4.16 GSI_SetQCodeMode

Description
Sets the Quick-Coding mode.

Declaration
GSI_SetQCodeMode(BYVAL lEnabled As Logical)

Remarks
This routine enables or disables the Quick-Coding. It can be only activated if a valid codelist is selected (see GSI_GetQCodeAvailable)

Parameters

lEnabled in TRUE: enable Quick-Coding

See Also
GSI_GetQCodeAvailable, GSI_ExecQCoding

Return-Codes

RC_OK successful

Example
See example file „meas.gbs“.

6.4.17 GSI_ExecQCoding

Description
Executes the Quick-Coding.
Declaration

GSI_ExecQCoding(
    BYVAL lRecEnable AS Logical
    iButtonId AS Integer,
    lNewCode AS Logical)

Remarks

This routine executes the Quick-Coding. If Quick-Coding is enabled, it checks the button iButtonId and searches the corresponding code. If the selected code needs mandatory attributes, it shows the coding dialog. As successful coding is indicated by lNewCode=TRUE. The results are stored in the Theodolite data pool (see GSI_GetWiEntry).

If lRecEnable=TRUE, this routine executes the ALL-button functionality too, it measures a distance and records the results. The recording order (measurement block – code block or vice versa) depends on the system setting (see GSI_GetRecOrder).

If lRecEnable=FALSE, this routine forces no new distance measurement and there is no recording.

Parameters

lRecEnable in TRUE: Quick-Coding including distance measurement. It records a code- and a measurement-block in the correct order.
FALSE: Quick-Coding without measurement and without recording

iButtonId inout In: Pressed button.
Out: If a Quick-Coding was possible, iButtonId is changed to MMI_NO_KEY, otherwise it is unchanged

lNewCode out TRUE: Quick-Coding was successful

See Also

GSI_GetQCodeAvailable, GSI_SetQCodeMode, GSI_SetRecOrder

Return-Codes

RC_OK successful

Example

See example files „meas.gbs“ and „meas_od.gbs“.
6.4.18 GSI_SetRecOrder

Description  Sets the recording order for Quick-Coding.

Declaration  GSI_SetRecOrder(BYVAL lCodeFirst As Logical)

Remarks  This routine defines the recording order for Quick-Coding.

If lCodeFirst=TRUE, then the code-block will be recorded before the measurement block.

Parameters  

lCodeFirst in  TRUE: code-block before measurement block

See Also  GSI_GetRecOrder, GSI_ExecQCoding

Return-Codes  

RC_OK  successful

Example  See example file „meas_od.gbs“.

6.4.19 GSI_GetRecOrder

Description  Returns the recording order for Quick-Coding.

Declaration  GSI_GetRecOrder(lCodeFirst As Logical)

Remarks  This routine returns the recording order for Quick-Coding.

If lCodeFirst=TRUE, then the code-block will be recorded before the measurement block.

Parameters  

lCodeFirst out  TRUE: code-block before measurement block

See Also  GSI_SetRecOrder, GSI_ExecQCoding

Return-Codes  

RC_OK  successful

Example  See example file „meas_od.gbs“.
6.4.20 GSI_QuickSet

**Description**  Shows the Quickset dialog.

**Declaration**  

GSI_QuickSet(BYVAL sCaptionLeft AS _Token)

**Remarks**  This procedure shows Quickset for station setting.

**Parameters**  

- **sCaptionLeft** in  Left caption for the Quickset dialog

**Return-Codes**

- **RC_OK**  Successful termination.

**Example**  Show the dialog:

```
GSI_QuickSet( "BASIC" )
```

6.4.21 GSI_SetRecPath

**Description**  Defines the recording path for the measurements.

**Declaration**  

GSI_SetRecPath(  
    BYVAL iPathInfo AS Integer,  
    BYVAL sFileName AS FileName,  
    BYVAL sFilePath AS FilePath )

**Remarks**  This procedure defines where the measurements will be recorded. If **iPathInfo** is set to GSI_INTERFACE, then the measurements will be sent to the RS232 line and the other parameters are not be interpreted. If **iPathInfo** is set to GSI_EXTERNAL, then **sFileName** defines the filename i.e. "MeasJob.GSI" and **sFilePath** defines the file-path, i.e. "A:\\GSI".

**Parameters**  

- **iPathInfo** in  Defines where the data are recorded
- **sFileName** in  Valid Filename (8+3 format)
- **sFilePath** in  file-path
6.4.22 GSI_GetRecPath

**Description**
Returns the recording path for the measurements.

**Declaration**

```plaintext
GSI_GetRecPath( 
  iPathInfo AS Integer, 
  sFileName AS FileName, 
  sFilePath AS FilePath )
```

**Remarks**
This procedure returns where the measurements will be recorded. If `iPathInfo = GSI_INTERFACE`, then the measurements will be sent to the RS232 line and the other parameters are not valid. If `iPathInfo = GSI_EXTERNAL`, then `sFileName` defines the filename i.e. "MeasJob.GSI" and `sFilePath` defines the filepath, i.e. "A:\\GSI".

Example
This example shows the actual recording path and set it to the RS232 line:

```plaintext
DIM sFile As FileName
DIM sPath As FilePath
DIM iPathInfo As Integer

GSI_GetRecPath(iPathInfo, sFile, sPath)
IF iPathInfo = GSI_EXTERNAL THEN
  MMI_PrintStr(0, 1, "RecFile-CARD: " + sFile, TRUE)
  MMI_PrintStr(0, 2, "Path: " + sPath, TRUE)
ELSE
  MMI_PrintStr(0, 1, "RecPath - serial line", TRUE)
END IF
GSI_SetRecPath( GSI_INTERFACE, sFile, sPath)
```
6.4.23 GSI_SetDataPath

Description
Set the file with the import data.

Declaration
GSI_SetDataPath(
   BYVAL iPathInfo AS Integer,
   BYVAL sFileName AS FileName,
   BYVAL sFilePath AS FilePath )

Remarks
This procedure sets the file from which data will be imported.
Only GSI_EXTERNAL is valid for the iPathInfo. sFileName defines the filename i.e. "DataJob.GSI" and sFilePath defines the file-path, i.e. "A:\GSI".

Parameters
iPathInfo         in  Device info (Only GSI_EXTERNAL is valid)
sFileName         in  Valid Filename (8+3 format)
sFilePath         in  File-path

Return-Codes
RC_OK             Successful termination.

See Also
GSI_GetDataPath

Example
The example defines the file “A:\GSI\DataJob.GSI” as new import file.
6.4.24 GSI_GetDataPath

Description  Get the name of the file with the import data.

Declaration  

GSI_GetDataPath(
    iPathInfo AS Integer,
    sFileName AS FileName,
    sFilePath AS FilePath
)

Remarks  This procedure fetches the name and the path of the file from which data will be imported. If `iPathInfo = GSI_EXTERNAL`, then `sFileName` defines the filename i.e. "DataJob.GSI" and `sFilePath` defines the file-path, i.e. "A:\GSI".

Parameters

- `iPathInfo` out  Device info
- `sFileName` out  Filename (8+3 format)
- `sFilePath` out  File-path

Return-Codes

- `RC_OK`  Successful termination.

See Also  GSI_SetDataPath

Example  The example fetches the name and the path of the standard import data file:

```basic
DIM iPathInfo AS Integer
DIM sFileName AS FileName
DIM sFilePath AS FilePath
GSI_GetDataPath(iPathInfo, sFileName, sFilePath)
```

6.4.25 GSI_GetWiEntry

Description  Get data from the Theodolite data pool.
GSI_GetWiEntry

**Declaration**
```
GSI_GetWiEntry(
    WiIdentification AS Integer,
    WiEntry AS GSI_Widg_Entry_Type )
```

**Remarks**
This routine is used to fetch data from the Theodolite data pool. All existing wi’s can be fetched (see the description of the WI constants for possible values).

**Parameters**
- **WiIdentification** in
  The identification of the WI.
- **WiEntry** out
  The WI entry data. See the description of GSI_Widg_Entry_Type for further information.

**See Also**
GSI_SetWiEntry

**Example**
See example GSI_SetWiEntry.

---

GSI_SetWiEntry

**Description**
Put data to the Theodolite data pool.

**Declaration**
```
GSI_SetWiEntry(
    WiIdentification AS Integer,
    WiEntry AS GSI_Widg_Entry_Type )
```

**Remarks**
This routine is used to put data to the Theodolite data pool. See the description of the WI constants.

**Parameters**
- **WiIdentification** in
  The identification of the WI.
- **WiEntry** in
  The WI entry data. See the description of GSI_Widg_Entry_Type for further information.

**See Also**
GSI_GetWiEntry

**Example**
GSI_SetWiEntry does not set WI.iId according to the first parameter, instead it will just use the value stored in WI.iId. If that value is unequal to the first parameter value, then it comes to a conflict. Use a GSI_GetWiEntry() first, to be sure that all values...
of the GSI_WiDlg_Entry_Type are initialized correctly. See also the example for the definition of a measurement dialog. Save way:

```plaintext
GSI_GetWiEntry ( GSI_ID_HR, Wi )
Wi.lValid = TRUE
Wi.dValue = 2.12
GSI_SetWiEntry ( GSI_ID_HR, Wi )
```

### 6.4.27 GSI_GetRecMask

**Description**
Get the definition and the format of a recording mask.

**Declaration**

```plaintext
GSI_GetRecMask(
    BYVAL iMaskNr AS Integer,
    sMaskName AS String18,
    RecWiMask AS GSI_Rec_Id_List,
    iRecFormat AS Integer,
    lEditMask AS Logical
)
```

**Remarks**
This routine fetches the definition and the format of the recording mask with the number iMaskNr. Valid formats are GSI_RECFORMAT_GSI and GSI_RECFORMAT_GSI16. A recording mask can be set with GSI_SetRecMask. If lEditMask is TRUE the elements of the recording mask can be changed in GSI_DefineRecMaskDlg. All unused elements of the recording list are set to GSI_ID_NONE. All values from 0 to 5 are valid for the mask number. Mask number 0 is predefined for the station recording mask.

**Note**
Only the first 16 characters of sMaskName are valid.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMaskNr</td>
<td>Number of the recording mask. GSI_ACTUAL_RECMASK can be used to retrieve settings of the actual mask</td>
</tr>
<tr>
<td>sMaskName</td>
<td>Name of the recording mask</td>
</tr>
<tr>
<td>RecWiMask</td>
<td>The definition of the recording mask. The elements of the array are the identification numbers of the WI’s. See the description</td>
</tr>
</tbody>
</table>
of the WI constants.

\[ \text{iRec} \quad \text{out} \quad \text{Recording format} \]
\[ \text{Format} \quad \text{(GSI\_RECFORMAT\_GSI, GSI\_RECFORMAT\_GSI16)} \]
\[ \text{lEditMask} \quad \text{out} \quad \text{Mask editable flag} \]

**See Also**
GSI\_SetRecMask, GSI\_DefineRecMaskDlg

**Example**
The example uses the GSI\_GetRecMask routine to fetch the definition and the format of the recording mask number 2.

```
DIM sMaskName AS String18
DIM RecWiMask AS GSI\_Rec\_Id\_List
DIM iRecFormat AS Integer
DIM lEditMask AS Logical

GSI\_GetRecMask(2, sMaskName, RecWiMask, iRecFormat, lEditMask)
```

### 6.4.28 GSI\_SetRecMask

**Description**
Set the definition and the format of a recording mask.

**Declaration**

```
GSI\_SetRecMask(
    BYVAL iMaskNr AS Integer,
    BYVAL sMaskName AS String18,
    BYVAL RecWiMask AS GSI\_Rec\_Id\_List,
    BYVAL iRecFormat AS Integer,
    BYVAL lEditMask AS Logical)
```

**Remarks**
This routine sets the definition and the format of the recording mask with the number iMaskNr. Valid formats are GSI\_RECFORMAT\_GSI and GSI\_RECFORMAT\_GSI16. If lEditMask is TRUE the elements of the recording mask can be changed in GSI\_DefineRecMaskDlg. All unused elements should be set to GSI\_ID\_NONE. All values from 0 to 5 are valid for the mask number. Mask number 0 is predefined for the station recording mask.
WiEntries must be unique, hence may not appear doubly.
2) Only GSI_MAX_REC_WI number of entries may be defined.
3) Only the first 16 characters of sMaskName are valid.

Parameters

iMaskNr in Number of the recording mask. GSI_ACTUAL_RECMASK can be used to set the values of the currently active mask.

sMaskName in Name of the recording mask.

RecWiMask in The definition of the recording mask. The elements of the array are the identification numbers of the WI’s. See the description of the WI constants.

iRec in Recording format (GSI_RECFORMAT_GSI, GSI_RECFORMAT_GSI16)

lEditMask in Mask editable flag

See Also GSI_GetRecMask, GSI_DefineRecMaskDlg
Example
The example sets the 4th element of the currently active recording mask on GSI_ID_HZ.

```basic
DIM sMaskName AS String18
DIM RecWiMask AS GSI_Rec_Id_List
DIM iRecFormat AS Integer
DIM lEditMask AS Logical

GSI_GetRecMask(GSI_ACTUAL_RECMASK, sMaskName,
RecWiMask, iRecFormat, lEditMask)
RecWiMask(4) = GSI_ID_HZ
GSI_SetRecMask(GSI_ACTUAL_RECMASK, sMaskName,
RecWiMask, iRecFormat, lEditMask)
```

6.4.29 GSI_SetRecMaskNr

Description
Set the used recording mask.

Declaration
GSI_SetRecMaskNr(BYVAL iMaskNr AS Integer)

Parameters
iMaskNr in Number of the recording mask. Number must be in the range 1.. GSI_MAX_REC_MASKS.

See Also
GSI_GetRecMaskNr
Example

The example sets the next recording mask.

```basic
DIM i AS Integer
GSI_GetRecMaskNr(i)
i = i + 1 'take next mask
i = ((i - 1) MOD GSI_MAX_REC_MASKS) + 1
GSI_SetRecMaskNr(i)
```

6.4.30 GSI_GetRecMaskNr

Description
Returns the used recording mask.

Declaration
GSI_GetRecMaskNr(iMaskNr AS Integer)

Parameters
- iMaskNr out Number of the recording mask.

See Also
- GSI_SetRecMaskNr

6.4.31 GSI_DefineRecMaskDlg

Description
Defines the recording mask dialog.

Declaration
GSI_DefineRecMaskDlg()

Remarks
Defines the contents of the recording mask. Using a dialog with
list-fields, the user can select the items for the user registration
mask. This routine is an interactive equivalent to the routines
GSI_GetRecMask and GSI_SetRecMask.

See Also
- GSI_GetRecMask, GSI_SetRecMask,

Example

GSI_DefineRecMaskDlg()
Declaration

GSI_ManCoordDlg(
    BYVAL sCaption AS _Token,
    BYVAL iPointType AS Integer,
    Point AS GSI_Point_Coord_Type,
    BYVAL iFlags AS Integer,
    BYVAL sHelpText AS _Token )

Remarks

This routine shows the manual co-ordinates input dialog and allows editing, coding and recording. The type of co-ordinates (station or target) can be selected using iPointType. Recording to the current data-file (defined in GSI_ImportCoordDlg) with REC or leaving this function with CONT is only possible if the point number is valid, and at least E- and N-co-ordinates are valid. If GSI_HEIGHT_MUST is included in iFlags the Height / Elevation-co-ordinate must be valid too. Leaving using ESC or QUIT (Shift-F6) is always possible. Recording and coding sets the according values in the Theodolite data-pool too.

Parameters

sCaption in
The maximal five-character long left part of the title bar.

iPointType in
station or target point. For the values for PointType see table below

<table>
<thead>
<tr>
<th>Point Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_STATION</td>
<td>station point number</td>
</tr>
<tr>
<td>GSI_INDIV_TG</td>
<td>individual target number</td>
</tr>
<tr>
<td>GSI_RUN_TG</td>
<td>running target</td>
</tr>
<tr>
<td>GSI_BACKSIGHT</td>
<td>backside number (analog target, only changed prompts)</td>
</tr>
</tbody>
</table>
### `GSI_POINT_CODE`

PointId / CodeId (analog target, only changed prompts)

<table>
<thead>
<tr>
<th>Point</th>
<th>in</th>
<th>only point number, co-ordinates will be set to 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point</td>
<td>out</td>
<td>point number and -co-ordinates. For further information see the description of GSI_Point_Coord_Type</td>
</tr>
<tr>
<td>iFlags</td>
<td>in</td>
<td>defines functionality</td>
</tr>
</tbody>
</table>

#### Valid Flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ALLOW_REC</td>
<td>allows recording and coding</td>
</tr>
<tr>
<td>GSI_HEIGHT_MUST</td>
<td>height must be entered</td>
</tr>
<tr>
<td>GSI_NE_OPTIONAL</td>
<td>only height must be entered, north &amp; east are optional</td>
</tr>
<tr>
<td>GSI_MULTI_REC</td>
<td>Allows entering and recording of more than one data-set, without leaving this routine</td>
</tr>
<tr>
<td>GSI_NO_FILE_CHANGE</td>
<td>File changing is disabled</td>
</tr>
</tbody>
</table>

Flag values can be combined with `+` operator

- `sHelpText` in: This text is shown, when the help button `SHIFT-F1` is pressed and the help functionality of the theodolite is enabled.

#### See Also

- `GSI_ImportCoordDlg`

#### Example

- `GSI_ImportCoordDlg`
6.4.33 GSI_ImportCoordDlg

Description
Show the co-ordinate import dialog.

Declaration
GSI_ImportCoordDlg(
    BYVAL sCaption AS _Token,
    BYVAL iPointType AS Integer,
    Point AS GSI_Point_Coord_Type,
    BYVAL iFlags AS Integer,
    BYVAL iImportFile AS Integer,
    BYVAL sImportHelp AS _Token,
    BYVAL sInputHelp AS _Token,
    BYVAL sF2Button AS _Token,
    BYVAL sF4Button AS _Token)

Remarks
This routine contains three dialogues, the search-, the view- and
the manual-input dialog. The type of co-ordinates (station or
target) can be selected using iPointType. The search dialog
allows selecting the data- or the measure file and editing a point-
number. Depending on the pressed button, the manual co-ordinate
input function (only if GSI_ALLOW_MAN is included in iFlags,
see GSI_ManCoordDlg) or the view-co-ordinates dialog will be
called.

The start of searching is always at the top of the file. With the two
search keys, the user can step from one valid point to the next in
both directions.

Rules for a valid point:
- point number found
- E- and N-coordinates (target or station) exists and are valid
- if GSI_HEIGHT_MUST is included in iFlags, a valid
height / elevation-coordinate must exist to within the file too.

If no valid point exists or no more valid points are in the desired search direction, a warning message will be displayed.

Parameters

- **sCaption** in
  The maximal five-character long left part of the title bar.

- **iPointType** in
  station or target point. For the values for **PointType** see table below

### Point Type

<table>
<thead>
<tr>
<th>Point Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_STATION</td>
<td>station point number</td>
</tr>
<tr>
<td>GSI_INDIV_TG</td>
<td>individual target number</td>
</tr>
<tr>
<td>GSI_RUN_TG</td>
<td>running target</td>
</tr>
<tr>
<td>GSI_BACKSIGHT</td>
<td>backside number (analog target, only changed prompts)</td>
</tr>
<tr>
<td>GSI_POINT_CODE</td>
<td>PointId / CodeId (analog target, only changed prompts)</td>
</tr>
</tbody>
</table>

- **Point** in
  Only point number, the co-ordinates will be set to 0.

- **Point** out
  point number and -co-ordinates. For further information see the description of **GSI_Point_Coord_Type**.

- **iFlags** in
  defines functionality

### Valid Flags

<table>
<thead>
<tr>
<th>Valid Flags</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_ALLOW_REC</td>
<td>allows recording and coding</td>
</tr>
<tr>
<td>GSI_MULTI_REC</td>
<td>Allows multiple manual coord. entering</td>
</tr>
</tbody>
</table>

GSI_ALLOW_MAN allows manual coord. entering
GSI_HEIGHT_MUST height must be entered
GSI_DIRECT_SEARCH direct searching without dialog
GSI_NO_VIEW no coord view if found
GSI_NE_OPTIONAL only height must be entered, north & east are optional
GSI_SEARCH_FROM_END Starts searching from end of file
GSI_NO_FILE_CHANGE Changing of file is disabled
GSI_GET_NEXT Return the next valid data-set, ignore sPtNr

Flags can be combined with ‘+’ - operator (iFlags = iFlag1 + iFlag2)

iImportFile in defines the source file for importing

Valid Import File Meaning
GSI_FILE_MEAS MEAS file
GSI_FILE_DATA DATA file
GSI_FILE_LAST last used file

sImportHelp in Help text for import dialog. Only visible if the help functionality of the theodolite is enabled.

sInputHelp in Help text for manual input dialog. Only visible if the help functionality of the theodolite is enabled.

sF2Button in Text for activating F2 button.

sF4Button in Text for activating F4 button

See Also GSI_ManCoordDlg
Example

```
DIM Point AS GSI_Point_Coord_Type
GSI_ImportCoordDlg( "IMP", GSI_INDIV_TG, 
Point, GSI_ALLOW_REC + GSI_ALLOW_MAN, 
GSI_FILE_DATA, "Import Help Text", 
"Input Help Text", "F2", "F4" )
```

### 6.4.34 GSI_SetLineSysMDlg

**Description**
Sets a line in the system measurement dialog.

**Declaration**
```
GSI_SetLineSysMDlg( 
  BYVAL iDlgNr AS Integer 
  BYVAL iLineNr AS Integer 
  BYVAL iSysParamId AS Integer 
)
```

**Remarks**
This routine sets one line in the system measurement dialog. To fetch information about a line, `GSI_GetLineSysMDlg` can be used. Unused lines should be set to `GSI_PAR_NONE`.

**Note**
1) Parameters are identified by `GSI_PAR_*` values and not by `GSI_ID_*` values.  
2) A line in the system measurement dialog can only be set to a system parameter not to an application parameter.

**Parameters**

- `iDlgNr` in: The number of the system measurement dialog where the line should be set. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_SYS_MDLG_1</td>
<td>Dialog 1</td>
</tr>
<tr>
<td>GSI_SYS_MDLG_2</td>
<td>Dialog 2</td>
</tr>
<tr>
<td>GSI_SYS_MDLG_3</td>
<td>Dialog 3</td>
</tr>
</tbody>
</table>

- `iLineNr` in: The number of the line to set. Valid numbers: 1.. `GSI_MAX_DLG_LINES`

- `iSysParamId` in: Identification of the system parameter. Refer to the chapter
“Constants for Measurement Dialog Definition”

See Also
- GSI_GetLineSysMDlg
- GSI_DefineMDlg

Example
See sample program “meas.gbs”.
This example uses GSI_SetLineSysMDlg to configure the first two lines of the first system measurement dialog.

GSI_SetLineSysMDlg( GSI_SYS_MDLG_1, 1, GSI_PAR_Date )
GSI_SetLineSysMDlg( GSI_SYS_MDLG_1, 2, GSI_PAR_Time )

6.4.35 GSI_GetLineSysMDlg

Description
Gets the definition of a line in the system measurement dialog.

Declaration
GSI_GetLineSysMDlg( BYVAL iDlgNr AS Integer, BYVAL iLineNr AS Integer, iSysParamId AS Integer )

Remarks
This routine fetches the information about the setting of one line in the system measurement dialog. To set a line in the system measurement dialog the routine GSI_SetLineSysMDlg can be used.
Parameters

iDlgNr in  The number of the system measurement dialog where the line should be fetched. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_SYS_MDLG_1</td>
<td>Dialog 1</td>
</tr>
<tr>
<td>GSI_SYS_MDLG_2</td>
<td>Dialog 2</td>
</tr>
<tr>
<td>GSI_SYS_MDLG_3</td>
<td>Dialog 3</td>
</tr>
</tbody>
</table>

iLineNr in  The number of the line to fetch.

iSysParamId out  Identification of the system parameter. Refer to the chapter “Constants for Measurement Dialog Definition”

See Also  GSI_SetLineSysMDlg
          GSI_DefineMDlg

Example  See sample program “meas.gbs”.
This example uses GSI_GetLineSysMDlg to get information about the configuration of the first system measurement dialog’s first two lines.

```
DIM iParLine1 AS Integer
DIM iParLine2 AS Integer

GSI_GetLineSysMDlg( GSI_SYS_MDLG_1, 1, iParLine1)
GSI_GetLineSysMDlg( GSI_SYS_MDLG_1, 2, iParLine2)
```

6.4.36  GSI_SetMDlgNr

Description  Sets the number of the system measurement dialog.

Declaration  GSI_SetMDlgNr( BYVAL iMDlgNr AS Integer)

Remarks  Sets the number of the system measurement dialog. The content of these dialogs can be changed by using of DefineMDlg.

Parameters
6.4.37 GSI_GetMDlgNr

**Description**
Returns the number of the system measurement dialog.

**Declaration**

```basic
GSI_GetMDlgNr(iMDlgNr AS Integer)
```

**Remarks**
Returns the number of the system measurement dialog.

**Parameters**
- `iMDlgNr` out Number of the actual measurement dialog

**See Also**
- GSI_SetMDlgNr

---

6.4.38 GSI_CreateMDlg

**Description**
Create and show the user definable measurement dialog.

**Declaration**

```basic
GSI_CreateMDlg(
  BYVAL iFixLines AS Integer,
  BYVAL sCaptionLeft AS _Token,
  BYVAL sCaptionRight AS _Token,
  BYVAL sHelpText AS _Token
)
```

**Remarks**
This routine creates and shows the user definable measurement dialog with `iFixLines` fix lines, the left part of the title bar `sCaptionLeft`, the caption `sCaptionRight`, and the help text `sHelpText`.

---

Example
See sample program “meas_od.gbs”. This example sets the next dialog mask

```basic
GSI_GetMDlgNr( i )
i = (i + 1) MOD GSI_MAX_MDLG_MASKS
GSI_SetMDlgNr( i )
```
Only one measurement dialog can exist at the same time. If GSI_CreateMDlg is called and there already exists a measurement dialog, the existing dialog (together with all attached buttons) is deleted and the new dialog is created.

**Note**  
If a graphics dialog or a text dialog exist together with a measurement dialog, all button routines (MMI_AddButton, MMI_GetButton, MMI_DeleteButton) are related to the measurement dialog.

The shown parameters used in the dialog are defined in the user display mask (see GSI_DefineMDlg).

**Parameters**

- **iFixLines in**  
The number of fix lines. (These lines are not scrolled.)

- **sCaptionLeft in**  
The part of the title bar displayed on the left border (up to five characters wide)

- **sCaptionRight in**  
The caption of the dialog.

- **sHelpText in**  
This text is shown, when the help button SHIFT-F1 is pressed and the help functionality of the theodolite is enabled.

**See Also**

- GSI_UpdateMDlg
- GSI_UpdateMeasurement

**Example**

See example file "meas.gbs" too.

This example uses the measure dialog routines GSI_CreateMDlg, GSI_UpdateMDlg and GSI_UpdateMeasurement to execute a measure process.

```
DIM ValidForRec AS Logical
DIM RetCodeForMsg AS Integer
DIM WaitForMsg AS Integer
DIM iButton AS Integer

WaitTime = 10 'ms

' user definition of measurement dialog
' can be placed here
```
GSI_CreateMDlg( 1, "WIR", "Measure Dialog", "This is the Helpertext")

DO
  GSI_UpdateMeasurement( TMC_MEA_INC, WaitTime, ValidForRec, RetCodeForMsg, FALSE )
  GSI_UpdateMDlg (iButton)
LOOP UNTIL iButton = MMI_ESC_KEY
GSI_DeleteDialog()

6.4.39 GSI_SetLineMDlg

Description
Sets one line in the user definable measurement dialog to system parameter.

Declaration
GSI_SetLineMDlg(  
  BYVAL iLineNr AS Integer  
  BYVAL iSysParamId AS Integer )

Remarks
This routine sets the configuration of a line in the user definable measurement dialog to a system parameter. This measurement dialog is initialized automatically with the actual settings of the first system measurement dialog. Modifications of the user definable dialog have no effects on the system measurement dialog and will be lost after termination of the program. An unused line should be set to GSI_PAR_NONE. To add a user definable application parameter to the dialog use GSI_SetLineMDlgPar. To add a line of text (e.g. separator line) to the dialog use GSI_SetLineMDlgText.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iLineNr in</td>
<td>The number of the line to set. Valid numbers: 1..GSI_MAX_DLG_LINES</td>
</tr>
<tr>
<td>iSysParamId in</td>
<td>Identification of the system parameter. Refer to the chapter “Constants for Measurement Dialog Definition”</td>
</tr>
</tbody>
</table>
6.4.40  GSI_SetLineMDlgText

Description  Puts a text line into the user definable measurement dialog.

Declaration  

GSI_SetLineMDlgText(  
BYVAL iLineNr AS Integer,  
BYVAL iParamId AS Integer,  
BYVAL sText AS _Token  
)

Remarks  This routine inserts a pure text line into the user definable measurement dialog. To add an user definable application parameter to the dialog use GSI_SetLineMDlgPar. To add a system parameter to the dialog use GSI_SetLineMDlg.

Parameters  

iLineNr  in  The number of the line to set.  
Valid numbers:  
1.. GSI_MAX_DLG_LINES

iParamId  in  Id of the system parameter.

sText  in  Contents of the line.

See Also  GSI_SetLineMDlg  
GSI_SetLineMDlgPar  
GSI_CreateMDlg
Example

This example uses GSI_SetLineMDlg and GSI_SetLineMDlgText to configure the user definable measurement dialog.

```
GSI_SetLineMDlg( 1, GSI_PAR_Date )
GSI_SetLineMDlg( 2, GSI_PAR_Time )
GSI_SetLineMDlgText( 3, GSI_PAR_APPDATA0,
                     "-----------------" )
GSI_SetLineMDlg( 4, GSI_PAR_Info1 )
GSI_SetLineMDlg( 5, GSI_PAR_Info2 )
```

6.4.41 GSI_SetLineMDlgPar

Description

Sets one line in the user definable measurement dialog to an application parameter.

Declaration

```
GSI_SetLineMDlgPar(
    BYVAL iLineNr AS Integer
    BYVAL iApplParamId AS Integer
    BYVAL sLabel AS _Token
    BYVAL lEditable AS Logical
    BYVAL iFormat AS Integer )
```

Remarks

This routine sets the configuration of a line in the user definable measurement dialog to an application parameter. The style of the application parameter is also defined in this routine. Any floating point format and strings are valid formats. The starting values of every application parameter is not predefined and hence has to be set explicitly. To initialize an application parameter the routine GSI_SetWiEntry can be used. To add a line of text to the dialog use GSI_SetLineMDlgText. To add a system parameter to the dialog use GSI_SetLineMDlg.

Parameters

- **iLineNr** in The number of the line to set. Valid numbers:
  1.. GSI_MAX_DLG_LINES
- **iApplParamId** in Id of the application parameter.
- **sLabel** in Description of parameter on display.
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI_FFORMAT_STRING</td>
<td>string</td>
</tr>
<tr>
<td>MMI_FFORMAT_DOUBLE</td>
<td>double</td>
</tr>
<tr>
<td>MMI_FFORMAT_DISTANCE</td>
<td>distance</td>
</tr>
<tr>
<td>MMI_FFORMAT_SUBDISTANCE</td>
<td>sub-distance [mm]</td>
</tr>
<tr>
<td>MMI_FFORMAT_ANGLE</td>
<td>angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_VANGLE</td>
<td>vertical angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_HZANGLE</td>
<td>horizontal angle</td>
</tr>
<tr>
<td>MMI_FFORMAT_TEMPERATURE</td>
<td>temperature</td>
</tr>
</tbody>
</table>

See Also
- GSI_SetLineMDlg
- GSI_SetLineMDlgText
- GSI_CreateMDlg

Example
See also sample file “meas.gbs”. This example uses GSI_SetLineMDlgPar and GSI_SetWiEntry to configure the user definable measurement dialog.
DIM WI AS GSI_WIDLG_ENTRY_TYPE
WI.lValid = FALSE
WI.iDataType = GSI_ASCII
GSI_SetWiEntry(GSI_ID_APPDATA0, WI)
GSI_SetLineMDlgPar(1, GSI_PAR_AppData0, "Stat. Name:", TRUE, MMI_FFORMAT_STRING)
WI.lValid = TRUE
WI.iDataType = GSI_DOUBLE
WI.dValue = 2.2
GSI_SetWiEntry(GSI_ID_APPDATA3, WI)
GSI_SetLineMDlgPar(8, GSI_PAR_AppData3, "Distance : ", TRUE, MMI_FFORMAT_DISTANCE)

6.4.42 GSI_UpdateMDlg

**Description**
Updates the user definable measurement dialog.

**Declaration**
GSI_UpdateMDlg( iButton As Integer)

**Remarks**
This procedure updates the user definable measurement dialog with the actual values from the Theodolite data pool and returns pressed buttons.

**Parameters**

- **iButton out**
  Contains pressed button identifier. For details see MMI_GetButton (lAllKeys = TRUE).

**See Also**
GSI_CreateMDlg
GSI_UpdateMeasurement

**Example**
See example GSI_CreateMDlg and example file „meas.gbs“. 
6.4.43 GSI_DefineMDlg

Description
Defines the entries of the user definable measurement dialog.

Declaration
GSI_DefineMDlg( BYVAL sCaption AS _Token)

Remarks
Interactively defines the contents of the user definable measurement dialog. Using a dialog with list fields, the user can select the items for the measurement dialog. This routine is an interactive equivalent to the routines GSI_SetLineSysMDlg and GSI_GetLineSysMDlg.

Parameters
sCaption in The left caption of the title bar. (Up to 5 characters wide.)

See Also
GSI_GetDlgMask
GSI_SetDlgMask

Example
GSI_DefineMDlg( "DEF" )

6.4.44 GSI_UpdateMeasurement

Description
Update the measurement data.

Declaration
GSI_UpdateMeasurement(
    iInclinePrg AS Integer,
    iWaitTime AS Integer,
    lValidForRec AS Logical,
    iRetCodeForMsg AS Integer,
    lChkIncRangeNow AS Logical )

Remarks
This function updates the measurement values in the Theodolite data pool. The data are the incline program, angles, distances, time, reflector height.
Parameters

iInclinePrg in
The manner of incline compensation. Following settings are possible:

<table>
<thead>
<tr>
<th>Incline Program</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_MEA_INC</td>
<td>get inclination</td>
</tr>
<tr>
<td>TMC_AUTO_INC</td>
<td>get inclination with automatism</td>
</tr>
<tr>
<td>TMC_PLANE_INC</td>
<td>get inclination always with plane</td>
</tr>
</tbody>
</table>

iWaitTime in
The wait time for a result (in ms). This time is used for synchronising the TMC task.

lValidForRec out
Indicates validity of the registration

iRetCodeForMsg out
Return code of the measurement

lChkIncRange in
TRUE: check incline range immediate

See Also
GSI_CreateMDlg
GSI_UpdateMDlg
GSI_DeleteDialog

Example
See example GSI_CreateMDlg and example file "meas.gbs".

6.4.45 GSI_Measure

Description
Measure and registration dialog.

Declaration
GSI_Measure ( )

Remarks
This procedure opens the measure and registration dialog.

Parameters
none
### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Success</td>
</tr>
</tbody>
</table>

### Example

Do a measure and registration dialog.

```basic
GSI_Measure();
```

---

#### 6.4.46 GSI_ExecuteAutoDist

**Description**
Executes an automatic distance measurement.

**Declaration**
```basic
GSI_ExecuteAutoDist();
```

**Remarks**
This procedure starts a distance measurement on condition that “Auto Dist” is enabled and one of the distance measurement-program buttons (FNC-menu) was pressed.

**Parameters**

### Return Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Success</td>
</tr>
</tbody>
</table>

**Example**
See example file `.meas.gbs` or `.meas_od.gbs`.

---

#### 6.4.47 GSI_CheckTracking

**Description**
Returns if distance tracking is running.

**Declaration**
```basic
GSI_CheckTracking(lTracking As Logical);
```

**Remarks**
This returns if a distance tracking is running.

An automatic start of distance tracking can be started on several conditions, i.e. by Quick-Coding, `GSI_ExecuteAutoDist` or by pressing buttons in the FNC-menu.

Tracking can be terminated by the instrument itself due several reasons, i.e. for laser security reasons (US-configuration).
6.4.48 GSI_RecordRecMask

Description  Recording the given wi mask.

Declaration  GSI_RecordRecMask (  
  RecList AS GSI_REC_ID_LIST,  
  BYVAL eProgFunction AS Logical,  
  BYVAL bCheckStdMask AS Logical,  
  BYVAL bIncAndSetRunPt AS Logical)

Remarks  This procedure records the given wi list. The target can be the  
          memory card or the interface. The parameter for the interface  
          depends on the GSI communication settings. Errors will shown on  
          the display, when recording list will be stored in the memory card.  
          Otherwise the error messages will be given on the interface.

Parameters  
  RecList in recording list  
  eProgFunction in program flag in the wi's (TRUE = ON, FALSE = OFF)  
  bCheckStdMask in testing the standard recording mask  
  bIncAndSetRunPt in increment the point number

Return Codes  
  RC_OK Success  
  RC_IVRESULT registration failure

See Also
Example

Record RecList.

DIM RecList AS GSI_REC_ID_LIST

' initialize RecList with adequate values
GSI_RecordRecMask ( RecList, TRUE, TRUE, TRUE )
6.5 CENTRAL SERVICE FUNCTIONS CSV

6.5.1 Summarizing Lists of CSV Types and Procedures

6.5.1.1 Types

<table>
<thead>
<tr>
<th>type name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPS_Fam_Type</td>
<td>Information about the current hardware.</td>
</tr>
<tr>
<td>Date_Time_Type</td>
<td>Date and time information.</td>
</tr>
<tr>
<td>Date_Type</td>
<td>Date information.</td>
</tr>
<tr>
<td>Time_Type</td>
<td>Time information.</td>
</tr>
</tbody>
</table>

6.5.1.2 Procedures

<table>
<thead>
<tr>
<th>procedure name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_ChangeFace</td>
<td>Do an absolute positioning to the opposite.</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>Returns if an alternative user-task was running.</td>
</tr>
<tr>
<td>CSV_Delay</td>
<td>Delay routine</td>
</tr>
<tr>
<td>CSV_GetATRStatus</td>
<td>Gets the current ATR state.</td>
</tr>
<tr>
<td>CSV_GetDateTime</td>
<td>Get the date and the time of the system.</td>
</tr>
<tr>
<td>CSV_GetElapseSysTime</td>
<td>Returns the difference between a reference time and the system time.</td>
</tr>
<tr>
<td>CSV_GetGBIVersion</td>
<td>Returns the release number of the GeoBASIC interpreter.</td>
</tr>
<tr>
<td>CSV_GetInstrumentFamily</td>
<td>Get information about the system.</td>
</tr>
<tr>
<td>CSV_GetInstrumentName</td>
<td>Get the LEICA specific instrument name.</td>
</tr>
<tr>
<td>CSV_GetInstrumentNo</td>
<td>Get the instrument number.</td>
</tr>
<tr>
<td>CSV_GetLaserPlummet</td>
<td>Returns the laser plummet state.</td>
</tr>
<tr>
<td>CSV_GetLockStatus</td>
<td>Gets the current state of the locking facility.</td>
</tr>
<tr>
<td>CSV_GetLRStatus</td>
<td>Returns the status of the system.</td>
</tr>
<tr>
<td>procedure name</td>
<td>description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>CSV_GetPrismType</td>
<td>Returns the used prism</td>
</tr>
<tr>
<td>CSV_GetSWVersion</td>
<td>Get the version of the system software.</td>
</tr>
<tr>
<td>CSV_GetSysTime</td>
<td>Returns the system time.</td>
</tr>
<tr>
<td>CSV_GetTargetType</td>
<td>Get the target type for distance measurements.</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>Returns the internal temperature of the instrument.</td>
</tr>
<tr>
<td>CSV_Laserpointer</td>
<td>Switch on / off the laser pointer.</td>
</tr>
<tr>
<td>CSV_LibCall</td>
<td>Call a GeoBASIC routine from another program.</td>
</tr>
<tr>
<td>CSV_LibCallAvailable</td>
<td>Check if GeoBASIC routine from another program is available.</td>
</tr>
<tr>
<td>CSV_LockIn</td>
<td>Starts locking (ATR)</td>
</tr>
<tr>
<td>CSV_LockOut</td>
<td>Stops locking (ATR)</td>
</tr>
<tr>
<td>CSV_MakePositioning</td>
<td>Do an absolute positioning.</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>Resets the “alternative user-task was running” flag.</td>
</tr>
<tr>
<td>CSV_SetATRStatus</td>
<td>Sets the current state of Automatic Target Recognition.</td>
</tr>
<tr>
<td>CSV_SetLaserPlummet</td>
<td>Switches the laser plummet.</td>
</tr>
<tr>
<td>CSV_SetLightGuide</td>
<td>Switch on / off the light guide.</td>
</tr>
<tr>
<td>CSV_SetLockStatus</td>
<td>Sets the current state of the locking facility.</td>
</tr>
<tr>
<td>CSV_SetPrismType</td>
<td>Sets the used prism</td>
</tr>
<tr>
<td>CSV_SetTargetType</td>
<td>Set the target type for distance measurements.</td>
</tr>
<tr>
<td>CSV_SysCall</td>
<td>Call a system function.</td>
</tr>
<tr>
<td>CSV_SysCallAvailable</td>
<td>Check if system function is available.</td>
</tr>
</tbody>
</table>
6.5.2 Data Structures for the Central Service Functions

6.5.2.1 Date_Time_Type: Date and Time

Description: These data structures are used to store date and time information.

```
TYPE Date_Type
  iVarYear AS Integer  ' year as a 4 digit number
  iVarMonth AS Integer ' month as a 2 digit number
  iVarDay AS Integer   ' day as a 2 digit number
END Date_Type

TYPE Time_Type
  iVarHour AS Integer    ' hour as a 2 digit number (24 hours format)
  iVarMinute AS Integer ' minutes as a 2 digit number
  iVarSecond AS Integer ' seconds as a 2 digit number
END Time_Type

Date_Time_Type
  Date AS Date_Type    ' date (as defined above)
  Time AS Time_Type   ' time (as defined above)
END Time_Type
```
6.5.2.2 TPS_Fam_Type: Information about the system

Description
This data structure is used to store information about the hardware. Further information about the hardware can be obtained by your local Leica representative.

```plaintext
TYPE TPS_Fam_Type
  iClass AS Integer
  The class of the system. Values:
  Id Meaning
  TPS1101 TPS1100 accuracy 1"
  TPS1102 TPS1100 accuracy 2"
  TPS1103 TPS1100 accuracy 3"
  TPS1105 TPS1100 accuracy 5"

  lEDMBuiltIn AS Logical
  EDM built-in

  lEDMTypeII AS Logical
  EDM built-in, type II

  lEDMTypeIII AS Logical
  EDM built-in, type III

  lEDMReflectorless AS Logical
  Red Laser

  lMotorized AS Logical
  Motorised

  lATR AS Logical
  Automatic Target Recognition (ATR)

  lEGL AS Logical
  EGL Guide Light

  lLaserPlummet AS Logical
  Laser Plummet

  lAutoCollimation AS Logical
  Auto-collimation lamp

  lSimulator AS Logical
  Hardware is simulator on Windows-PC

END TPS_Fam_Type
```
6.5.3 CSV_GetDateTime

**Description**
Get the date and the time of the system.

**Declaration**
```
CSV_GetDateTime(
    DateAndTime AS Date_Time_Type )
```

**Remarks**
The CSV_GetDateTime routine reads the date and the time from the system's real-time clock (RTC) and returns the values in the structure Date_Time_Type. In the case of TPS_Sim the system clock will be read.

**Parameters**
- **DateAndTime out** The structure for the date and the time.

**Return Codes**
- **RC_UNDEFINED** The date and time is not set (not yet/not any longer).

**Example**
The example uses the CSV_GetDateTime routine to get the date and the time of the system and displays the values.

```
DIM DT AS Date_Time_Type
ON ERROR RESUME
CSV_GetDateTime( DT )

IF ERR = RC_OK THEN
    MMI_PrintInt( 0, 0, 5, DT.Date.iYear, TRUE )
    MMI_PrintInt( 6, 0, 3, DT.Date.iMonth, TRUE )
    MMI_PrintInt( 10, 0, 3, DT.Date.iDay, TRUE )
    MMI_PrintInt( 0, 1, 3, DT.Time.iHour, TRUE )
    MMI_PrintInt( 4, 1, 3, DT.Time.iMinute, TRUE )
    MMI_PrintInt( 8, 1, 3, DT.Time.iSecond, TRUE )
ELSEIF ERR = RC_UNDEFINED THEN
    MMI_PrintStr( 0, 0, "Date and time not set.", TRUE )
ELSE
    MMI_PrintStr( 0, 0, "Unexpected error code.", TRUE )
END IF
```
### 6.5.4 CSV_GetTemperature

**Description**
Returns the internal temperature of the instrument.

**Declaration**
CSV_GetTemperature( IntTemp AS Temperature )

**Remarks**
This routine returns the internal temperature.

**Parameters**
- IntTemp out Internal temperature

### 6.5.5 CSV_GetInstrumentName

**Description**
Get the LEICA specific instrument name.

**Declaration**
CSV_GetInstrumentName( sName AS String30 )

**Remarks**
The CSV_GetInstrumentName routine returns the name of the system in the string sName.

**Parameters**
- sName out The LEICA specific instrument name.

**Return Codes**
none

**See Also**
CSV_GetInstrumentNo,
CSV_GetInstrumentFamily

**Example**
The example uses the CSV_GetInstrumentName routine to get the instrument name and displays it.

```geo
DIM sName AS String30
CSV_GetInstrumentName ( sName )
MMI_PrintStr ( 0, 0, sName, TRUE )
```
6.5.6 CSV_GetInstrumentNo

**Description**  
Get the instrument number.

**Declaration**  
CSV_GetInstrumentNo( iSerialNo AS Integer )

**Remarks**  
The CSV_GetInstrumentNo routine returns the serial number of the system.

**Parameters**

- **iSerialNo** out  
The serial number of the system.

**Return Codes**

none

**See Also**
CSV_GetInstrumentName,  
CSV_GetInstrumentFamily

**Example**  
The example uses the CSV_GetInstrumentNo routine to get the instrument number and displays it.

```geo
DIM iSerialNo AS Integer
CSV_GetInstrumentNo( iSerialNo )
MMI_PrintInt( 0, 1, 20, iSerialNo, TRUE )
```

6.5.7 CSV_GetInstrumentFamily

**Description**  
Get information about the system.

**Declaration**  
CSV_GetInstrumentFamily(  
  Family AS TPS_Fam_Type )

**Remarks**  
The CSV_GetInstrumentFamily routine returns the class and the instrument type of the system (see description of the data structure TPS_Fam for return values).

**Parameters**

- **Family** out  
  Contains the class and instrument type data. See description of the data structure TPS_Fam for return values.

**See Also**

TPS_Sim  
Always sets Family.lSimulator to TRUE.
See Also
CSV_GetInstrumentName, CSV_GetInstrumentNo

Example
The example uses the CSV_GetInstrumentFamily routine to get information about the instrument and displays it.

```
DIM Family AS TPS_Fam_Type
CSV_GetInstrumentFamily( Family )
MMI_PrintInt( 0, 1, 10, Family.iClass, TRUE )
IF (Family.lSimulator) THEN
  MMI_PrintString( 0, 2, 10, "ON TPS_SIM", TRUE)
END IF
```

6.5.8 CSV_GetSWVersion

Description
Get the version of the system software.

Declaration
CSV_GetSWVersion( iRelease AS Integer,
iVersion AS Integer )

Remarks
The CSV_GetSWVersion routine returns the Release number and the number of the system software version. These numbers can be interpreted together as software identification (Release.Version, e.g. 1.05).

```
TPS_Sim  Delivers the version of the simulator.
```

Parameters
```
iRelease       out  value of the Release number can be in the range from 0 to 99
iVersion        out  value of the version number can be in the range from 0 to 99
```

See Also
Example  The example uses the `CSV_GetSWVersion` routine to get the system software version and displays it.
DIM iRelease AS Integer
DIM iVersion AS Integer

    CSV_GetSWVersion( iRelease, iVersion )
    MMI_PrintVal( 0, 0, 6, 2, 
                   iRelease + iVersion / 100, TRUE )

6.5.9 CSV_GetGBIVersion

Description  Returns the release number of the GeoBASIC interpreter.

Declaration  CSV_GetGBIVersion( 
              iRelease as Integer, 
              iVersion as Integer, 
              iSubVersion as Integer )

Remarks  This function returns the release version of the running GeoBASIC interpreter.

Parameters  

    iRelease       out  Release number
    iVersion       Out  Version Number
    iSubVersion    out  Subversion number

Return-Codes  

    RC_OK  Successful termination.
Example
This example shows the currently used GeoBASIC interpreter release number.

```
DIM iRel As Integer
DIM iVer As Integer
DIM iSubVer As Integer

MMI_CreateTextDialog(
  6, "-CSV-", "Test CSV", "no help available")
CSV_GetGBIVersion (iRel, iVer, iSubVer)
MMI_PrintStr(0, 0,
  "GBI: \"+Str$(iRel) + "." +
  Str$(iVer) + "."+Str$(iSubVer), TRUE)
MMI_DeleteDialog()
```

6.5.10  CSV_GetElapseSysTime

Description
Returns the difference between a reference time and the system time.

Declaration

```
CSV_GetElapseSysTime( iRefTime AS Integer,
                        iElapse AS Integer )
```

Remarks
The routine CSV_GetElapseSysTime returns the difference of between a given reference time iRefTime and the systems time. Whenever the system starts up, the system time is reset.

Parameters

```
iRefTime   in   The reference time.
iElapse    out  The difference between iRefTime and the system time. The difference is returned in [ms].
```

See Also
CSV_GetSysTime,
CSV_GetDateTime
Example
The example uses the routine `CSV_GetElapseSysTime` to get a time difference.

```basic
DIM iElapse AS Integer
DIM iRefTime AS Integer

CSV_GetSysTime(iRefTime)'returns reference time
'do something...
CSV_GetElapseSysTime( iRefTime, iElapse )
MML_PrintInt ( 0, 0, 20, iElapse, TRUE )
```

### 6.5.11 CSV_GetSysTime

| **Description** | Returns the system time. |
| **Declaration** | `CSV_GetSysTime( iTime AS Integer )` |
| **Remarks** | The routine returns the systems time. Whenever the system starts up, the system time is reset. |
| **Parameters** | iTime out The system time in ms. |

**See Also**
`CSV_GetElapseSysTime`, `CSV_GetDateTime`

**Example** See `CSV_GetElapsedTime`.

### 6.5.12 CSV_GetLRStatus

| **Description** | Returns the status of the system. |
| **Declaration** | `CSV_GetLRStatus( iLRStatus AS Integer )` |
| **Remarks** | The routine `CSV_GetLRStatus` returns the mode of the system. The system can either be in local or in Remote mode. For Release 1.0 this function always delivers local mode as an answer. |

TPS_Sim Delivers the system up time of the PC.
Note This function is reserved for future purposes and has no special usage in the current implementation.

TPS_Sim Always delivers LOCAL_MODE.

Parameters

Parameters:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCAL_MODE</td>
<td>0</td>
<td>local mode</td>
</tr>
<tr>
<td>REMOTE_MODE</td>
<td>1</td>
<td>Remote mode</td>
</tr>
</tbody>
</table>

Example

The example uses the routine CSV_GetLRStatus to get the mode of the system.

```basic
DIM iLRStatus AS Integer

CSV_GetLRStatus( iLRStatus )

MMI_PrintInt( 0, 0, 10, iLRStatus, TRUE )
```

6.5.13 CSV_SetGuideLight

Description Set the guide light intensity.

Declaration

`CSV_SetGuideLight( BYVAL iLight AS Integer )`

Remarks Sets the guide light intensity.

Parameters

<table>
<thead>
<tr>
<th><code>iLight</code></th>
<th>in</th>
<th>Guide light intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td></td>
<td>Meaning</td>
</tr>
<tr>
<td>CSV_EGL_OFF</td>
<td></td>
<td>Switching off</td>
</tr>
<tr>
<td>CSV_EGL_LOW</td>
<td></td>
<td>Low intensity</td>
</tr>
<tr>
<td>CSV_EGL_MID</td>
<td></td>
<td>Middle intensity</td>
</tr>
<tr>
<td>CSV_EGL_HIGH</td>
<td></td>
<td>High intensity</td>
</tr>
</tbody>
</table>

Return Codes

| RC_SYSBUSY       | EDM is busy. Guide light cannot be switched. |
| RC_NOT_IMPL      | Guide light Hardware is not available       |
### 6.5.14 CSV_Laserpointer

**Description**
Switch on / off the laser pointer.

**Declaration**

```plaintext
CSV_Laserpointer( BYVAL lLaser AS Logical )
```

**Remarks**
Switches on / off the laser pointer.

**Parameters**

- `lLaser` in
  Switch on / off the Laser pointer (TRUE = on, FALSE = off)

**Return Codes**

- **RC_SYSBUSY**
  EDM is busy. Laser pointer cannot be switched.

- **RC_NOT_IMPL**
  Laser pointer Hardware is not available.

**Example**
Switch off the laser pointer.

```plaintext
CSV_Laserpointer( FALSE )
```

### 6.5.15 CSV_MakePositioning

**Description**
Do an absolute positioning.

**Declaration**

```plaintext
CSV_MakePositioning(BYVAL dHz AS Double,
                     BYVAL dV AS Double)
```

**Remarks**
Absolute positioning of the Theodolite axes to the desired angles with the currently active tolerance for positioning. If any control function is active at the point of call, it will be cancelled and the positioning will be performed. After the positioning the controller will be automatically activated for manual input for the moving device. When starting the positioning the calling application has to take care that a valid inclination plane is available for an angle measure, as it can normally not be redone during positioning.

The positioning is done with the planes valid at the beginning of
it. During the process no inclination will be measured. The used positioning method can cause inexact results, especially for steep V > ~25 GON

**Parameters**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dHz</td>
<td>Corrected Hz-angle [Radiant]</td>
</tr>
<tr>
<td>dV</td>
<td>Corrected V-angle [Radiant]</td>
</tr>
</tbody>
</table>

**Return Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_IVPARAM</td>
<td>No valid positioning angle.</td>
</tr>
<tr>
<td>CSV_DETENT_ERROR</td>
<td>target angle is out of the limits or a collision is occurred.</td>
</tr>
<tr>
<td>CSV_TIMEOUT</td>
<td>time out at positioning of one or both axes</td>
</tr>
<tr>
<td>CSV_MOTOR_ERROR</td>
<td>error in subsystem</td>
</tr>
<tr>
<td>CSV_ANGLE_ERROR</td>
<td>error at measuring the angle</td>
</tr>
<tr>
<td>RC_FATAL</td>
<td>fatal error</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>system abort</td>
</tr>
</tbody>
</table>

**See Also**

BAP_PosTelescope

**Example**

Perform an absolute positioning.

```vbscript
CSV_MakePositioning( 0, 2*atn(1) ) ' (0, Pi/2)
```

### 6.5.16 CSV_ChangeFace

**Description**

Do an absolute positioning to the opposite.

**Declaration**

```vbscript
CSV_ChangeFace()
```

**Remarks**

Perform positioning into the position opposite to the current. If any control function is active at the point of call, it will be cancelled and the positioning will be performed. After the positioning the controller will be automatically activated for manual input for the moving device. When starting the positioning the calling application has to take care that a valid inclination plane is available for an angle measure, as it can normally not be redone during positioning.
The positioning is done with the planes valid at the beginning of it. During the process no inclination will be measured. The used positioning method can cause inexact results, especially for steep \( V > \sim 25 \text{ GON} \)

**Parameters**

none

**Return Codes**

- RC_IVPARAM: No valid positioning angle.
- CSV_DETENT_ERROR: Target angle is out of the limits or a collision is occurred.
- CSV_TIMEOUT: Time out at positioning of one or both axes.
- CSV_MOTOR_ERROR: Error in subsystem.
- CSV_ANGLE_ERROR: Error at measuring the angle.
- RC_FATAL: Fatal error.
- RC_ABORT: System abort.

**See Also**

BAP_PosTelescope

**Example**

Perform a change of face.

```basic
CSV_ChangeFace()
```

### 6.5.17 CSV_SetLockStatus

**Description**

Sets the current state of the locking facility.

**Declaration**

```basic
CSV_SetLockStatus(BYVAL lOn AS Logical )
```

**Remarks**

It switches the locking facility on or off.

**Parameters**

- **lOn**
  - In
  - Switches on / off the locking facility
  - (TRUE = on, FALSE = off)
Return Codes

- RC_FATAL: fatal error
- RC_NOT_IMPL: if ATR hardware is not available
- RC_ABORT: system abort

See Also
- CSV_SetLockStatus,
- CSV_LockIn,
- CSV_LockOut

Example

Perform an absolute positioning.

```plaintext
CSV_SetLockStatus( TRUE ) ' switches locking on
```

6.5.18 CSV_GetLockStatus

Description

Gets the current state of the locking facility.

Declaration

```plaintext
CSV_GetLockStatus( lOn AS Logical )
```

Remarks

It queries the TPS system if the locking facility is on or off.

Parameters

- lOn: meaning
  - FALSE: Locking is switched off.
  - TRUE: Locking is switched on.

Return Codes

- RC_FATAL: fatal error
- RC_NOT_IMPL: if ATR hardware is not available
- RC_ABORT: system abort

See Also
- CSV_GetLockStatus,
- CSV_LockIn,
- CSV_LockOut

Example

Perform an absolute positioning.

```plaintext
DIM l AS Logical
CSV_SetLockStatus( l ) ' queries locking
```
6.5.19 CSV_LockIn

**Description**
Starts the locking facility.

**Declaration**
```c
CSV_LockIn( )
```

**Remarks**
If ATR is switched on then locking to the target will be done. If no target available, then manual positioning will be started.

**Parameters**
```c
none
```

**Return Codes**
- `AUT_RC_NOT_ENABLED`: Theodolite without ATR or lock status not set
- `AUT_RC_MOTOR_ERROR`: Error at motor control.
- `AUT_RC_DETECTOR_ERROR`: Error at ATR
- `AUT_RC_NO_TARGET`: No target at the detection range
- `AUT_RC_BAD_ENVIRONMENT`: Bad environment at the detection range (bad light…)
- `RC_NOT_IMPL`: if ATR hardware is not available

**See Also**
- `CSV_GetLockStatus`,
- `CSV_SetLockStatus`,
- `CSV_LockOut`

**Example**
This example starts locking.
```c
CSV_LockIn( )
```
6.5.20 CSV_LockOut

**Description**
Stops a running locking function.

**Declaration**
CSV_LockOut( )

**Parameters**
none

**Return Codes**
- RC_OK: no error
- RC_NOT_IMPL: if ATR hardware is not available

**See Also**
CSV_GetLockStatus, CSV_SetLockStatus, CSV_LockIn

**Example**
This example stops locking.
CSV_LockOut( )

6.5.21 CSV_SetATRStatus

**Description**
Sets the current state of Automatic Target Recognition.

**Declaration**
CSV_SetATRStatus(BYVAL lOn AS Logical )

**Remarks**
It switches the ATR facility on or off.

**Parameters**
- lOn in
  Switches on / off the ATR facility
  (TRUE = on, FALSE = off)

**Return Codes**
- RC_FATAL: fatal error
- RC_ABORT: system abort
- RC_NOT_IMPL: if ATR hardware is not available

**Example**
Perform an absolute positioning.
CSV_SetATRStatus( TRUE ) ' switches ATR on
6.5.22 CSV_GetATRStatus

Description  
Gets the current ATR state.

Declaration  
CSV_GetATRStatus(lOnl AS Logical )

Remarks  
It queries the TPS system if the ATR facility is on or off.

Parameters  
Out  
meaning  
FALSE  
ATR is switched off.  
TRUE  
ATR is switched on.

Return Codes  
RC_FATAL  
fatal error  
RC_ABORT  
system abort  
RC_NOT_IMPL  
if ATR hardware is not available

Example  
Get current ATR status.

DIM l AS Logical  
CSV_SetATRStatus( l )

6.5.23 CSV_Delay

Description  
This routine delays the execution of a program.

Declaration  
CSV_Delay( BYVAL iDelay AS Integer )

Remarks  
This routine delay using the operating system, that means that other Theodolite tasks can run during the delay (It is not a busy waiting).

Note  
Avoid busy waiting using FOR - or WHILE loops.

TPS_Sim  
Delay resolution is one second. iDelay < 500 means no delay
6.5.24 CSV_SetTargetType

**Description**  
Set the target type for distance measurements.

**Declaration**  
```basic
CSV_SetTargetType(  
    BYVAL iTargetType as Integer  
)
```

**Remarks**  
This routine sets the target type for distance measurements. The target type defines if the next distance measurement happens with prism or without prism.

**Parameters**  
- `iTargetType` in  
  Target type

<table>
<thead>
<tr>
<th>Valid target types</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_WITH_REFLECTOR</td>
<td>With reflector</td>
</tr>
<tr>
<td>CSV_WITHOUT_REFLECTOR</td>
<td>Without reflector</td>
</tr>
</tbody>
</table>

**Return-Codes**  
- RC_OK  
  Successful termination.
- RC_IVPARAM  
  Instrument don’t support this target type
6.5.25 CSV_GetTargetType

Description
Get the target type for distance measurements.

Declaration
CSV_GetTargetType( iTargetType as Integer )

Remarks
This routine fetches the target type for distance measurements. The target type defines if the next distance measurement happens with prism or without prism.

Parameters
- **iTargetType**
  - `out` Target type

<table>
<thead>
<tr>
<th>Valid target types</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_WITH_REFLECTOR</td>
<td>With reflector</td>
</tr>
<tr>
<td>CSV_WITHOUT_REFLECTOR</td>
<td>Without reflector</td>
</tr>
</tbody>
</table>

Return-Codes
- **RC_OK** Successful termination.

See
CSV_SetTargetType, BAP_SetMeasPrg, BAP_GetMeasPrg

Example
The example fetches the target type.

```
DIM iTargetType AS Integer
CSV_GetTargetType(iTargetType)
```
6.5.26 CSV_SetPrismType

Description  Sets the used prism.

Declaration  CSV_SetPrismType( BYVAL iPrism as Integer)

Remarks  This routine sets the used prism iPrism (BAP_PRISM_ROUND, BAP_PRISM_TAPE, BAP_PRISM_MINI, BAP_PRISM_360, BAP_PRISM_USER1, BAP_PRISM_USER2 or BAP_PRISM_USER3). If iPrism is one of the user defined prisms and this prism is actually not defined then this routine will return RC_IVRESULT.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPrism</td>
<td>Integer</td>
<td>Used prism</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVRESULT</td>
<td>Prism not defined.</td>
</tr>
</tbody>
</table>

See  CSV_GetPrismType

Example  The example sets the 360 degrees prism.

CSV_SetPrismType(BAP_PRISM_360)

6.5.27 CSV_GetPrismType

Description  Returns the used prism.

Declaration  CSV_GetPrismType(iPrism as Integer)

Remarks  This routine returns the used prism iPrism.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPrism</td>
<td>out</td>
<td>Used prism</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
<tr>
<td>RC_IVRESULT</td>
<td>Prism not defined.</td>
</tr>
</tbody>
</table>
See CSV_SetPrismType

Example
The example returns the used prism.

```plaintext
DIM iPrism AS Integer
CSV_SetPrismType( iPrism )
```

### 6.5.28 CSV_SetLaserPlummet

**Description**
Switches the laser plummet.

**Declaration**
```plaintext
CSV_SetLaserPlummet( BYVAL lOn as Logical )
```

**Remarks**
This function switches the optional laser plummet. The plummet will be switched off automatically after 3 minutes.

**Parameters**
- **lOn**
  - **in**
  - **TRUE**: switch plummet on

**Return-Codes**
- **RC_OK**
  - Successful termination.

See CSV_GetLaserPlummet, CSV_GetInstrumentFamily

### 6.5.29 CSV_GetLaserPlummet

**Description**
Returns the laser plummet state.

**Declaration**
```plaintext
CSV_GetLaserPlummet( lOn as Logical )
```

**Remarks**
This function returns the state of the optional laser plummet.

**Parameters**
- **lOn**
  - **out**
  - **TRUE**: plummet is switched on

**Return-Codes**
- **RC_OK**
  - Successful termination.
6.5.30 CSV_CheckAltUserTask

**Description**
Returns if an alternative user-task was running.

**Declaration**
CSV_CheckAltUserTask(lWasRunning AS Logical)

**Remarks**
This routine returns if an alternative user-task was running. One of these tasks can be started by pressing one of the buttons FNC, Shift-FNC, PROG, Shift-PROG, Light and Level. Functions, executed by an alternative user task, can change several system settings. The CSV_CheckAltUserTask routine notifies the running GeoBASIC application that it was interrupted by another program. With this information, the GeoBASIC program is able to respond to these changes.

After processing this information, the subroutine CSV_ResetAltUserTask must be called.

**Parameters**
- lWasRunning out TRUE: a task was running

**Return-Codes**
- RC_OK Successful termination.

**See**
CSV_ResetAltUserTask

**Example**
The example checks if an alternative task was running.
```
CSV_CheckAltUserTask( l )
IF l THEN
   send("AltUserTask: was running")
ELSE
   send("AltUserTask: was NOT running")
END IF
CSV_ResetAltUserTask( )
```

6.5.31 CSV_ResetAltUserTask

**Description**
Resets the “alternative user-task was running” flag.

**See**
CSV_ResetAltUserTask
**Declaration**

CSV_ResetAltUserTask()

**Remarks**

This routine restarts the alternative user-task tracking.

**Parameters**

none

**Return-Codes**

RC_OK

Successful termination.

**See**

CSV_CheckAltUserTask

---

### 6.5.32 CSV_SysCall

**Description**

Call a system function.

**Declaration**

CSV_SysCall( BYVAL CId AS CIdType)

**Remarks**

This routine works in two different forms depending on the parameter CId. If CId is a system function CSV_SysCall calls the function directly. In the other form the CId is a system event. In this case CSV_SysCall calls the system function (or dialog, menu, macro, application) which is defined in the current configuration to handle this event. See description of the system functions and system events in the appendix H.

**Parameters**

CId in System function or system event

**Return-Codes**

RC_OK

Successful termination.

RC_IVPARAM

No function defined to handle the event

RC_NOT_IMPL

System function not available
See CSV_SysCallAvailable

Example The example calls the system function electronic level.

CSV_SysCall(CSV_SFNC_Libelle)

6.5.33 CSV_SysCallAvailable

Description Check if system function is available.

Declaration CSV_SysCallAvailable(
    BYVAL CId AS CIdType,
    lAvailable AS Logical )

Remarks This routine checks, if it is possible to call the function CId if CId is a system function or if there is a function defined and available to handle the event CId if CId is an system event. See the description of system functions and system events in appendix H.

Parameters

<table>
<thead>
<tr>
<th>CId</th>
<th>in</th>
<th>System function or system event.</th>
</tr>
</thead>
<tbody>
<tr>
<td>lAvailable</td>
<td>out</td>
<td>TRUE: System function is available or function (dialog, menu, macro, application) to handle the event is defined and available.</td>
</tr>
</tbody>
</table>

Return-Codes

| RC_OK | Successful termination. |
See CSV_SysCall

Example The example checks if the red laser is available.
DIM lAvailable AS Logical

CSV_SysCallAvailable(CSV_SFNC_ToggleRedLaser,
lAvailable)

6.5.34 CSV_LibCall

Description Call a GeoBASIC or C application routine of another program.

Declaration CSV_LibCall( BYVAL PrgName AS String255,
BYVAL FuncName AS String255,
BYVAL CptShort AS _Token )

Remarks This routine is used to call a GeoBASIC routine which is defined in
another program. Please refer also to Appendix

Parameters
PrgName in Program name
FuncName in Function name
CptShort In Short caption for dialogs

Return-Codes
RC_OK Successful termination.

See CSV_LibCallAvailable

Example See IAC.GBS and IAC2.GBS for an example.

6.5.35 CSV_LibCallAvailable

Description Check if the GeoBASIC routine from another program is available.

Declaration CSV_LibCallAvailable( BYVAL PrgName AS String255,
BYVAL FuncName AS String255,
lAvailable AS Logical )
Remarks
This routine checks if a GeoBASIC routine which is defined in another program is available. Usually this means that it checks if the other program is loaded and the specified entry point exists.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrgName</td>
<td>Program name</td>
</tr>
<tr>
<td>FuncName</td>
<td>Function name</td>
</tr>
<tr>
<td>lAvailable</td>
<td>Routine is available</td>
</tr>
</tbody>
</table>

Return-Codes

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>Successful termination.</td>
</tr>
</tbody>
</table>

See
CSV_LibCall

Example
See IAC.GBS and IAC2.GBS for an example.
StatementSequence ::= { [ ErrorLabel ] Statement }
ErrorLabel ::= HandlerLabel ":" 
Statement ::= ( SequentialStatement | SelectionStatement | LoopStatement | OnErrorStatement | ExitStatement | IOSStatement )
SequentialStatement ::= ( Assignment | SubroutineCall )
Assignment ::= Variable "=" Expression
SelectionStatement ::= ( IfStatement | SelectStatement )
IfStatement ::= "IF" Condition "THEN"
StatementSequence
{ "ELSEIF" Condition "THEN"
StatementSequence } [ "ELSE"
StatementSequence ]
"END IF"
Condition ::= LogicalExpression
SelectStatement ::= "SELECT CASE" Expression
{ "CASE" ConstantList
StatementSequence } [ "CASE ELSE"
StatementSequence ]
"END SELECT"
ConstantList ::= Constant { "," Constant }
LoopStatement ::= ( WhileLoop | UntilLoop | ForLoop )
WhileLoop ::= "DO" [ "WHILE" Condition ]
StatementSequence "LOOP"
UntilLoop ::= "DO"
StatementSequence "LOOP" [ "UNTIL" Condition ]
ForLoop ::= "FOR" CounterName "=" Start "TO"
Finish [ "STEP" Step ]
StatementSequence "NEXT" [ CounterName ]
Condition ::= LogicalExpression
Start ::= IntegerExpression
GeoBASIC Reference Manual
Appendix A — GeoBasic Syntax

Finish ::= IntegerExpression
Step ::= IntegerExpression
ExitStatement ::= (LoopExit | RoutineExit)
LoopExit ::= "EXIT"

RoutineDeclaration ::= (SubroutineDeclaration | FunctionDeclaration)
SubroutineDeclaration ::= ["GLOBAL"] "SUB"
SubroutineName ::= [ParameterList] Body
"END" [SubroutineName]
FunctionDeclaration ::= "FUNCTION" FunctionName ParameterList
"AS" DataTypeName Body
"END" [FunctionName]

ParameterList ::= 
| ParameterSpecification {","} ParameterSpecification |
|"
ParameterSpecification ::= ["BYVAL"] ParameterName
"AS" DataTypeName

Body ::= {CVTDeclaration | LabelDeclaration | CodePart}
CVTDeclaration ::= (ConstantDeclaration | VariableDeclaration | TypeDeclaration)

CodePart ::= StatementSequence
ExitStatement ::= (LoopExit | RoutineExit)
RoutineExit ::= "EXIT" ("SUB" | "FUNCTION")

SubroutineCall ::= ["CALL"] SubroutineName
| ActualParameterList |
FunctionCall ::= FunctionName ActualParameterList
ActualParameterList ::= "(" [Expression {","Expression} ] ")"

LabelDeclaration ::= "LABEL" HandlerLabel
OnErrorStatement ::= "ON ERROR" ("RESUME NEXT" | "GOTO"
| HandlerLabel | "0")

HandlerLabel ::= Name
ErrorLabel ::= HandlerLabel ":="

Program ::= "PROGRAM" ProgramName
| CVTDeclaration | RoutineDeclaration |
"END" [ProgramName]

IOStatement ::= "WRITE" Expression
AppInfo Syntax

```
AppInfo ::= "APPINFO "
  [ GeneralSection ]
  { GlobalSubSection }
"END" "APPINFO"

GeneralSection ::= "GENERAL"
  { GeneralSectionEntry }
"END" "GENERAL"

GlobalSubSection ::= "ENTRYPOINT" GlobalSubName
  { GlobalSubSectionEntry }
"END" [ GlobalSubName ]

GeneralSectionEntry ::= "SET"
  GeneralSectionKey
  StringConstant

GlobalSubSectionEntry ::= "SET"
  GlobalSubSectionKey
  StringConstant

GeneralSectionKey ::= "AUTHOR" |
  "DESC" |
  "THEOMODEL"

GlobalSubSectionKey ::= "CAPSH" |
  "DESC" |
  "HELP"
```
Appendix B — GLOSSARY

**ATR**
Automatic Recognition means that the TPS can search and recognise a target automatically.

**BAP**
This means Basic Application Programs. This subsystem contains several basic functionalities:
- Setup the configuration
- Distance measurement and entering the manual distance
- Positioning the telescope

**CSV**
This abbreviation stands for Central Services.
The subsystem contains several administration functions:
- Clock and time functions
- Functions for instrument identification (instrument name, instrument family, ...)
- Functions for system information (local, Remote, locking...)
- Functions for positioning the theodolite

**External Routine**
A routine that resides in a different part of the TPS-1100-System. Its interface must conform to certain rules, and it must be made known to the compiler, i.e. the definition must be compiled and linked to it. External routines can be called from a GeoBASIC routine like any other subroutine. They return an error code in the predefined variable **Err**.

**TPS**
Theodolite Positioning System
**TPS-1100-System**

The target hardware and its software, comprising, among others, the GeoBASIC loader objects.

**Loader Object**

Strictly speaking this is the result of the compilation of a program; a binary file that can be downloaded onto the target hardware. In a more general sense it also used as a synonym for "program".

**GM**

The section Geodesy Mathematics contains mathematical functions, which are often used in geodesy applications, for example calculation of intersection, clothoid, average values, triangle etc. Furthermore, the accuracy of deviated values can be calculated.

**GSI**

This abbreviation stands for Geodesy Serial Interface. The subsystem contains several functions:

- Functions for registration (point number, rec.-mask,..)
- Functions for create, show, update or delete dialogs
- Functions for fetching data from WIR data pool

**MMI**

The subsystem MMI (Man Machine Interface) manages the user interaction with the system.

**Module**

A GeoBASIC subroutine that has been declared with the prefix `global` and can be called from the TPS-1100-System. Modules are numbered sequentially, and it is this number that is made known to the loader and the TPS-1100-System.
Predefined Type

Structured types used by external routines can be made known to the compiler in a way similar to the definition of the interface of an external routine. Their definition must be compiled and linked to the GeoBASIC compiler.

Predefined Variable

There is one GeoBASIC variable, Err, that is defined for all programs. It is used to contain the return code of an external routine. Its value is passed to the TPS-1100-System upon completion of the execution of a module.

Program

A collection of GeoBASIC modules that have some commonality, such as common (global) variables. A GeoBASIC program contains one or more modules, plus any number of global types, variables, subroutines, and functions. A program is compiled in its entirety; this produces a loader object that is subsequently downloaded onto the target hardware.

Routine

Generic name for subroutines, functions, modules, and external routines. Subroutines and functions are entirely local to a GeoBASIC program and not accessible from outside. Modules can be called from outside, i.e. from the TPS-1100-System. External routines are routines that reside somewhere else in the TPS-1100-System, but are called from a GeoBASIC routine.

TMC

The Theo Measurement function contains some fundamental measurement procedures.

_Token

Special kind of string parameters to be passed to TPS-1100-system software routines. Actual values of such parameters must be of type string literal or string constant. The compiler generates automatically a token number out the string value, which will be used as an index from the interpreter. But, of course, this has to be calculated during compile time and cannot be a runtime calculated one.
Appendix C — List of Reserved Words

The following words are reserved by GeoBASIC and cannot be used as names (identifiers) in a GeoBASIC program. They must be written as given, except that upper and lower case letters are not distinguished.

<table>
<thead>
<tr>
<th>Reserved Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>FOR</td>
</tr>
<tr>
<td>AS</td>
<td>SELECT</td>
</tr>
<tr>
<td>BYVAL</td>
<td>STEP</td>
</tr>
<tr>
<td>CALL</td>
<td>STRING</td>
</tr>
<tr>
<td>CASE</td>
<td>SUB</td>
</tr>
<tr>
<td>CONST</td>
<td>THEN</td>
</tr>
<tr>
<td>DIM</td>
<td>TO</td>
</tr>
<tr>
<td>DO</td>
<td>TYPE</td>
</tr>
<tr>
<td>ELSE</td>
<td>UNTIL</td>
</tr>
<tr>
<td>ELSEIF</td>
<td>WHILE</td>
</tr>
<tr>
<td>END</td>
<td>WRITE</td>
</tr>
<tr>
<td>EXIT</td>
<td>PROGRAM</td>
</tr>
</tbody>
</table>
## Appendix D — Derived Mathematical Functions

The following is a list of non intrinsic mathematical functions that can be derived from the intrinsic math functions provided with GeoBASIC:

<table>
<thead>
<tr>
<th>Function</th>
<th>GeoBASIC equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secant</td>
<td>Sec(X) = 1 / Cos(X)</td>
</tr>
<tr>
<td>Cosecant</td>
<td>Cosec(X) = 1 / Sin(X)</td>
</tr>
<tr>
<td>Cotangent</td>
<td>Cotan((X) = 1 / Tan(X)</td>
</tr>
<tr>
<td>Inverse Sine</td>
<td>Arcsin(X) = Atn(X / Sqr(-X * X + 1))</td>
</tr>
<tr>
<td>Inverse Cosine</td>
<td>Arccos(X) = Atn(-X / Sqr(-X * X + 1)) + 1.5708</td>
</tr>
<tr>
<td>Inverse Secant</td>
<td>Arcsec(X) = Atn(X / Sqr(X * X - 1)) + Sgn(Sgn(X) -1) * 1.5708</td>
</tr>
<tr>
<td>Inverse Cosecant</td>
<td>Arccosec(X) = Atn(X/Sqr(X * X - 1)) + (Sgn(X) - 1) * 1.5708</td>
</tr>
<tr>
<td>Inverse Cotangent</td>
<td>Arccotan(X) = Atn(X) + 1.5708</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>GeoBASIC equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperbolic Sine</td>
<td>HSin(X) = (Exp(X) - Exp(-X)) / 2</td>
</tr>
<tr>
<td>Hyperbolic Cosine</td>
<td>HCos(X) = (Exp(X) + Exp(-X)) / 2</td>
</tr>
<tr>
<td>Hyperbolic Tangent</td>
<td>HTan(X) = (Exp(X) - Exp(-X)) / (Exp(X) + Exp(-X))</td>
</tr>
<tr>
<td>Hyperbolic Secant</td>
<td>HSec(X) = 2 / (Exp(X) + Exp(-X))</td>
</tr>
<tr>
<td>Hyperbolic Cosecant</td>
<td>HCosec(X) = 2 / (Exp(X) - Exp(-X))</td>
</tr>
<tr>
<td>Hyperbolic Cotangent</td>
<td>HCotan(X) = (Exp(X) + Exp(-X)) / (Exp(X) - Exp(-X))</td>
</tr>
<tr>
<td>Inverse Hyperbolic Sine</td>
<td>HArcsin(X) = Log(X + Sqr(X * X + 1))</td>
</tr>
<tr>
<td>Inverse Hyperbolic Cosine</td>
<td>HArccos(X) = Log(X + Sqr(X * X - 1))</td>
</tr>
<tr>
<td>Function</td>
<td>GeoBASIC equivalent</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Inverse Hyperbolic Tangent</td>
<td>HArctan(X) = Log((1 + X) / (1 - X)) / 2</td>
</tr>
<tr>
<td>Inverse Hyperbolic Secant</td>
<td>HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)</td>
</tr>
<tr>
<td>Inverse Hyperbolic Cosecant</td>
<td>HArccosec(X) = Log((Sgn(X) * Sqr(X * X + 1) +1) / X)</td>
</tr>
<tr>
<td>Inverse Hyperbolic Cotangent</td>
<td>HArccotan(X) = Log((X + 1) / (X - 1)) / 2</td>
</tr>
<tr>
<td>Logarithm</td>
<td>LogN(X) = Log(X) / Log(N)</td>
</tr>
</tbody>
</table>
Appendix E — GEOFONT
Errors which may occur during execution of a GeoBASIC program are associated with several subsystems which are supported by GeoBASIC. For each subsystem we know a different range of return values which will be listed in the following tables. Since some of the explanations of the return values are dependent on the context see the descriptions of the system functions in the reference manual too.

### Appendix F — SYSTEM RETURN CODES

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_OK</td>
<td>0</td>
<td>0x0</td>
<td>Function successfully completed.</td>
</tr>
<tr>
<td>RC_UNDEFINED</td>
<td>1</td>
<td>0x1</td>
<td>Unknown error, result unspecified.</td>
</tr>
<tr>
<td>RC_IVPARAM</td>
<td>2</td>
<td>0x2</td>
<td>Invalid parameter detected. Result unspecified.</td>
</tr>
<tr>
<td>RC_IVRESULT</td>
<td>3</td>
<td>0x3</td>
<td>Invalid result.</td>
</tr>
<tr>
<td>RC_FATAL</td>
<td>4</td>
<td>0x4</td>
<td>Fatal error.</td>
</tr>
<tr>
<td>RC_NOT_Impl</td>
<td>5</td>
<td>0x5</td>
<td>Not implemented yet.</td>
</tr>
<tr>
<td>RC_TIME_OUT</td>
<td>6</td>
<td>0x6</td>
<td>Function execution timed out. Result unspecified.</td>
</tr>
<tr>
<td>RC_SET_INCOMPL</td>
<td>7</td>
<td>0x7</td>
<td>Parameter setup for subsystem is incomplete.</td>
</tr>
<tr>
<td>RC_ABORT</td>
<td>8</td>
<td>0x8</td>
<td>Function execution has been aborted.</td>
</tr>
<tr>
<td>RC_NOMEMORY</td>
<td>9</td>
<td>0x9</td>
<td>Fatal error - not enough memory.</td>
</tr>
<tr>
<td>RC_NOTINIT</td>
<td>10</td>
<td>0xA</td>
<td>Fatal error - subsystem not initialized.</td>
</tr>
<tr>
<td>RC_SHUT_DOWN</td>
<td>12</td>
<td>0xC</td>
<td>Subsystem is down.</td>
</tr>
<tr>
<td>RC_SYSBUSY</td>
<td>13</td>
<td>0xD</td>
<td>System busy/already in use of another process. Cannot execute function.</td>
</tr>
<tr>
<td>RC_HWFAILURE</td>
<td>14</td>
<td>0xE</td>
<td>Fatal error - hardware failure.</td>
</tr>
<tr>
<td>RC_ABORT_APPL</td>
<td>15</td>
<td>0xF</td>
<td>Execution of application has been aborted (SHIFT-ESC).</td>
</tr>
<tr>
<td>RC_LOW_POWER</td>
<td>16</td>
<td>0x10</td>
<td>Operation aborted - insufficient power supply level.</td>
</tr>
<tr>
<td>RC_IVVERSION</td>
<td>17</td>
<td>0x11</td>
<td>Invalid version of file, ...</td>
</tr>
<tr>
<td>RC_BATT_EMPTY</td>
<td>18</td>
<td>0x12</td>
<td>Battery empty</td>
</tr>
<tr>
<td>RC_NO_EVENT</td>
<td>20</td>
<td>0x14</td>
<td>no event pending.</td>
</tr>
<tr>
<td>RC_OUT_OF_TEMP</td>
<td>21</td>
<td>0x15</td>
<td>out of temperature range</td>
</tr>
</tbody>
</table>
### GeoBASIC Reference Manual Appendix F — System Return Codes

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_INSTRUMENT_TILT</td>
<td>22</td>
<td>0x16</td>
<td>instrument tilting out of range</td>
</tr>
<tr>
<td>RC_COM_SETTING</td>
<td>23</td>
<td>0x17</td>
<td>communication error</td>
</tr>
<tr>
<td>RC_NO_ACTION</td>
<td>24</td>
<td>0x18</td>
<td>RC_TYPE Input 'do no action'</td>
</tr>
<tr>
<td>RC_SLEEP_MODE</td>
<td>25</td>
<td>0x19</td>
<td>Instr. run into the sleep mode</td>
</tr>
</tbody>
</table>

### ANG 256 0x100

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANG_ERROR</td>
<td>257</td>
<td>0x101</td>
<td>Angles and Inclinations not valid</td>
</tr>
<tr>
<td>ANG_INCL_ERROR</td>
<td>258</td>
<td>0x102</td>
<td>inclinations not valid</td>
</tr>
<tr>
<td>ANG_BAD_ACC</td>
<td>259</td>
<td>0x103</td>
<td>value accuracy not reached</td>
</tr>
<tr>
<td>ANG_BAD_ANGLE_ACC</td>
<td>260</td>
<td>0x104</td>
<td>angle-accuracy not reached</td>
</tr>
<tr>
<td>ANG_BAD_INCLIN_ACC</td>
<td>261</td>
<td>0x105</td>
<td>inclination accuracy not reached</td>
</tr>
<tr>
<td>ANG_WRITE_PROTECTED</td>
<td>266</td>
<td>0x10A</td>
<td>no write access allowed</td>
</tr>
<tr>
<td>ANG_OUT_OF_RANGE</td>
<td>267</td>
<td>0x10B</td>
<td>value out of range</td>
</tr>
<tr>
<td>ANG_IR_OCCURED</td>
<td>268</td>
<td>0x10C</td>
<td>function aborted due to interrupt</td>
</tr>
<tr>
<td>ANG_HZMOVED</td>
<td>269</td>
<td>0x10D</td>
<td>hz moved during incline measurement</td>
</tr>
<tr>
<td>ANG_OS_ERROR</td>
<td>270</td>
<td>0x10E</td>
<td>troubles with operation system</td>
</tr>
<tr>
<td>ANG_DATA_ERROR</td>
<td>271</td>
<td>0x10F</td>
<td>overflow at parameter values</td>
</tr>
<tr>
<td>ANG_PEAK_CNT_UFL</td>
<td>272</td>
<td>0x110</td>
<td>too less peaks</td>
</tr>
<tr>
<td>ANG_TIME_OUT</td>
<td>273</td>
<td>0x111</td>
<td>reading timeout</td>
</tr>
<tr>
<td>ANG_TOO_MANY_EXPOS</td>
<td>274</td>
<td>0x112</td>
<td>too many exposures wanted</td>
</tr>
<tr>
<td>ANG_PIX_CTRL_ERR</td>
<td>275</td>
<td>0x113</td>
<td>picture height out of range</td>
</tr>
<tr>
<td>ANG_MAX_POS_SKIP</td>
<td>276</td>
<td>0x114</td>
<td>positive exposure dynamic overflow</td>
</tr>
<tr>
<td>ANG_MAX_NEG_SKIP</td>
<td>277</td>
<td>0x115</td>
<td>negative exposure dynamic overflow</td>
</tr>
<tr>
<td>ANG_EXP_LIMIT</td>
<td>278</td>
<td>0x116</td>
<td>exposure time overflow</td>
</tr>
<tr>
<td>ANG_UNDER_EXPOSURE</td>
<td>279</td>
<td>0x117</td>
<td>picture under-exposed</td>
</tr>
<tr>
<td>ANG_OVER_EXPOSURE</td>
<td>280</td>
<td>0x118</td>
<td>picture over-exposed</td>
</tr>
<tr>
<td>ANG_TLESS_PEAKS</td>
<td>301</td>
<td>0x12D</td>
<td>too less peaks detected</td>
</tr>
<tr>
<td>ANG_PEAK_TOO_Slim</td>
<td>302</td>
<td>0x12E</td>
<td>peak too slim</td>
</tr>
<tr>
<td>ANG_PEAK_TOO_WIDE</td>
<td>303</td>
<td>0x12F</td>
<td>peak to wide</td>
</tr>
<tr>
<td>ANG_BAD_PEAKDIFF</td>
<td>304</td>
<td>0x130</td>
<td>bad peak difference</td>
</tr>
<tr>
<td>ANG_UNDER_EXP_PICT</td>
<td>305</td>
<td>0x131</td>
<td>too less peak amplitude</td>
</tr>
<tr>
<td>ANG_PEAKS_INHOMOGEN</td>
<td>306</td>
<td>0x132</td>
<td>in-homogenous peak amplitudes</td>
</tr>
<tr>
<td>ANG_NO_DECOD_POSS</td>
<td>307</td>
<td>0x133</td>
<td>no peak decoding possible</td>
</tr>
<tr>
<td>ANG_UNSTABLE_DECOD</td>
<td>308</td>
<td>0x134</td>
<td>peak decoding not stable</td>
</tr>
</tbody>
</table>
### GeoBASIC Reference Manual

**Appendix F — System Return Codes**

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANG_TLESS_FPEAKS</td>
<td>309</td>
<td>0x135</td>
<td>too less valid fine-peaks</td>
</tr>
<tr>
<td>ATA_RC_NOT_READY</td>
<td>512</td>
<td>0x200</td>
<td>ATR-System is not ready.</td>
</tr>
<tr>
<td>ATA_RC_NO_RESULT</td>
<td>513</td>
<td>0x201</td>
<td>Result isn't available yet.</td>
</tr>
<tr>
<td>ATA_RC_SEVERAL_TARGETS</td>
<td>514</td>
<td>0x202</td>
<td>Several Targets detected.</td>
</tr>
<tr>
<td>ATA_RC_BIG_SPOT</td>
<td>515</td>
<td>0x203</td>
<td>Spot is too big for analyze.</td>
</tr>
<tr>
<td>ATA_RC_BACKGROUND</td>
<td>516</td>
<td>0x204</td>
<td>Background is too bright.</td>
</tr>
<tr>
<td>ATA_RC_NO_TARGETS</td>
<td>517</td>
<td>0x205</td>
<td>No targets detected.</td>
</tr>
<tr>
<td>ATA_RC_SPOT_ON_EDGE</td>
<td>518</td>
<td>0x206</td>
<td>Spot is on the edge of the sensing area.</td>
</tr>
<tr>
<td>ATA_RC_BLOOMING</td>
<td>519</td>
<td>0x207</td>
<td>Blooming or spot on edge detected.</td>
</tr>
<tr>
<td>ATA_RC_NOT_BUSY</td>
<td>520</td>
<td>0x208</td>
<td>ATR isn't in a continuous mode.</td>
</tr>
<tr>
<td>ATA_RC_STRANGE_LIGHT</td>
<td>521</td>
<td>0x209</td>
<td>Not the spot of the own target illuminator.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>522</td>
<td>0x20A</td>
<td>Communication error to sensor (ATR).</td>
</tr>
<tr>
<td>ATA_RC_HZ_FAIL</td>
<td>523</td>
<td>0x20B</td>
<td>No Spot detected in Hz-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>524</td>
<td>0x20C</td>
<td>No Spot detected in V-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>525</td>
<td>0x20D</td>
<td>Strange light in Hz-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>526</td>
<td>0x20E</td>
<td>Strange light in V-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>527</td>
<td>0x20F</td>
<td>On multiple ATA_SLDR_OpenTransfer.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>528</td>
<td>0x210</td>
<td>No Spot detected in Hz-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>529</td>
<td>0x211</td>
<td>No Spot detected in V-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>530</td>
<td>0x212</td>
<td>Strange light in Hz-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>531</td>
<td>0x213</td>
<td>Strange light in V-direction.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>532</td>
<td>0x214</td>
<td>No ATA_SLDR_OpenTransfer happened.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>533</td>
<td>0x215</td>
<td>Unexpected data format received.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>534</td>
<td>0x216</td>
<td>Checksum error in transmitted data.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>535</td>
<td>0x217</td>
<td>Address out of valid range.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>536</td>
<td>0x218</td>
<td>Firmware file has invalid format.</td>
</tr>
<tr>
<td>ATA_RC_V24_FAIL</td>
<td>537</td>
<td>0x219</td>
<td>Current (loaded) Firmware doesn't support upload.</td>
</tr>
</tbody>
</table>

TPS1100-Version 1.30
## EDM System Return Codes

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDM_SYSTEM_ERR</td>
<td>769</td>
<td>0x301</td>
<td>Fatal EDM sensor error. See for the exact reason the original EDM sensor error number. In the most cases a service problem.</td>
</tr>
<tr>
<td>EDM_INVALID_COMMAND</td>
<td>770</td>
<td>0x302</td>
<td>Invalid command or unknown command, see command syntax.</td>
</tr>
<tr>
<td>EDM_BOOM_ERR</td>
<td>771</td>
<td>0x303</td>
<td>Boomerang error.</td>
</tr>
<tr>
<td>EDM_SIGN_LOW_ERR</td>
<td>772</td>
<td>0x304</td>
<td>Received signal to low, prism to far away, or natural barrier, bad environment, etc.</td>
</tr>
<tr>
<td>EDM_DIL_ERR</td>
<td>773</td>
<td>0x305</td>
<td>DIL distance measurement out of limit.</td>
</tr>
<tr>
<td>EDM_SIGN_HIGH_ERR</td>
<td>774</td>
<td>0x306</td>
<td>Received signal to strong, prism to near, stranger light effect.</td>
</tr>
<tr>
<td>EDM_DEV_NOT_INSTALLED</td>
<td>778</td>
<td>0x30A</td>
<td>Device like EGL, DL is not installed.</td>
</tr>
<tr>
<td>EDM_NOT_FOUND</td>
<td>779</td>
<td>0x30B</td>
<td>Search result invalid. For the exact explanation see in the description of the called function.</td>
</tr>
<tr>
<td>EDM_ERROR_RECEIVED</td>
<td>780</td>
<td>0x30C</td>
<td>Communication ok, but an error reported from the EDM sensor.</td>
</tr>
<tr>
<td>EDM_MISSING_SRVPWD</td>
<td>781</td>
<td>0x30D</td>
<td>No service password is set.</td>
</tr>
<tr>
<td>EDM_INVALID_ANSWER</td>
<td>782</td>
<td>0x30E</td>
<td>Communication ok, but an unexpected answer received.</td>
</tr>
<tr>
<td>EDM_SEND_ERR</td>
<td>783</td>
<td>0x30F</td>
<td>Data send error, sending buffer is full.</td>
</tr>
<tr>
<td>EDM.Receive_ERR</td>
<td>784</td>
<td>0x310</td>
<td>Data receive error, like parity buffer overflow.</td>
</tr>
<tr>
<td>EDM_INTERNAL_ERR</td>
<td>785</td>
<td>0x311</td>
<td>Internal EDM subsystem error.</td>
</tr>
<tr>
<td>EDM_BUSY</td>
<td>786</td>
<td>0x312</td>
<td>Sensor is working already, abort current measuring first.</td>
</tr>
<tr>
<td>EDM_NO_MEASACTIVITY</td>
<td>787</td>
<td>0x313</td>
<td>No measurement activity started.</td>
</tr>
<tr>
<td>EDM_CHKSUM_ERR</td>
<td>788</td>
<td>0x314</td>
<td>Calculated checksum, resp. received data wrong (only in binary communication mode possible).</td>
</tr>
</tbody>
</table>
### GeoBASIC Reference Manual
#### Appendix F — System Return Codes

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDM_INIT_OR_STOP_ERR</td>
<td>789</td>
<td>0x315</td>
<td>During start up or shut down phase an error occurred. It is saved in the DEL buffer.</td>
</tr>
<tr>
<td>EDM_SRL_NOTAVAILABLE</td>
<td>790</td>
<td>0x316</td>
<td>Red laser not available on this sensor HW.</td>
</tr>
<tr>
<td>EDM_MEAS_ABORTED</td>
<td>791</td>
<td>0x317</td>
<td>Measurement will be aborted (will be used for the lasersecurity)</td>
</tr>
<tr>
<td>EDM_SLDR_TRANSFER_PENDING</td>
<td>798</td>
<td>0x31E</td>
<td>Multiple OpenTransfer calls.</td>
</tr>
<tr>
<td>EDM_SLDR_TRANSFER_ILLEGAL</td>
<td>799</td>
<td>0x31F</td>
<td>No opentransfer happened.</td>
</tr>
<tr>
<td>EDM_SLDR_DATA_ERROR</td>
<td>800</td>
<td>0x320</td>
<td>Unexpected data format received.</td>
</tr>
<tr>
<td>EDM_SLDR_CHK_SUM_ERROR</td>
<td>801</td>
<td>0x321</td>
<td>Checksum error in transmitted data.</td>
</tr>
<tr>
<td>EDM_SLDR_ADDR_ERROR</td>
<td>802</td>
<td>0x322</td>
<td>Address out of valid range.</td>
</tr>
<tr>
<td>EDM_SLDR_INV_LOADFILE</td>
<td>803</td>
<td>0x323</td>
<td>Firmware file has invalid format.</td>
</tr>
<tr>
<td>EDM_SLDR_UNSUPPORTED</td>
<td>804</td>
<td>0x324</td>
<td>Current (loaded) firmware doesn't support upload.</td>
</tr>
<tr>
<td>EDM_UNKNOW_ERR</td>
<td>808</td>
<td>0x328</td>
<td>Undocumented error from the EDM sensor, should not occur.</td>
</tr>
</tbody>
</table>

### GMF
#### RetCodeName

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM_WRONG_AREA_DEF</td>
<td>1025</td>
<td>0x401</td>
<td>Wrong Area Definition.</td>
</tr>
<tr>
<td>GM_IDENTICAL_PTS</td>
<td>1026</td>
<td>0x402</td>
<td>Identical Points.</td>
</tr>
<tr>
<td>GM_PTS_IN_LINE</td>
<td>1027</td>
<td>0x403</td>
<td>Points on one line.</td>
</tr>
<tr>
<td>GM_OUT_OF_RANGE</td>
<td>1028</td>
<td>0x404</td>
<td>Out of range.</td>
</tr>
<tr>
<td>GM_PLAUSIBILITY_ERR</td>
<td>1029</td>
<td>0x405</td>
<td>Plausibility error.</td>
</tr>
<tr>
<td>GM_TOO_FEW_OBSERVATIONS</td>
<td>1030</td>
<td>0x406</td>
<td>To few Observations to calculate the average.</td>
</tr>
<tr>
<td>GM_NO_SOLUTION</td>
<td>1031</td>
<td>0x407</td>
<td>No Solution.</td>
</tr>
<tr>
<td>GM_ONE_SOLUTION</td>
<td>1032</td>
<td>0x408</td>
<td>Only one solution.</td>
</tr>
<tr>
<td>GM_TWO_SOLUTIONS</td>
<td>1033</td>
<td>0x409</td>
<td>Second solution.</td>
</tr>
<tr>
<td>GM_ANGLE_SMALLER_15GON</td>
<td>1034</td>
<td>0x40A</td>
<td>Warning: Intersection angle &lt; 15gon.</td>
</tr>
<tr>
<td>GM_INVALID_</td>
<td>1035</td>
<td>0x40B</td>
<td>Invalid triangle.</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>TRIANGLE_TYPE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM_INVALID_ANGLE_SYSTEM</td>
<td>1036</td>
<td>0x40C</td>
<td>Invalid angle unit.</td>
</tr>
<tr>
<td>GM_INVALID_DIST_SYSTEM</td>
<td>1037</td>
<td>0x40D</td>
<td>Invalid distance unit.</td>
</tr>
<tr>
<td>GM_INVALID_V_SYSTEM</td>
<td>1038</td>
<td>0x40E</td>
<td>Invalid vertical angle.</td>
</tr>
<tr>
<td>GM_INVALID_TEMP_SYSTEM</td>
<td>1039</td>
<td>0x40F</td>
<td>Invalid temperature system.</td>
</tr>
<tr>
<td>GM_INVALID_PRES_SYSTEM</td>
<td>1040</td>
<td>0x410</td>
<td>Invalid pressure unit.</td>
</tr>
<tr>
<td>GM_RADIUS_NOT_POSSIBLE</td>
<td>1041</td>
<td>0x411</td>
<td>Invalid radius.</td>
</tr>
<tr>
<td>GM_NO_PROVISIONAL_VALUES</td>
<td>1042</td>
<td>0x412</td>
<td>GM2: insufficient data.</td>
</tr>
<tr>
<td>GM_SINGULAR_MATRIX</td>
<td>1043</td>
<td>0x413</td>
<td>GM2: bad data.</td>
</tr>
<tr>
<td>GM_TOO_MANY_ITERATIONS</td>
<td>1044</td>
<td>0x414</td>
<td>GM2: bad data distr.</td>
</tr>
<tr>
<td>GM_IDENTICAL_TIE_POINTS</td>
<td>1045</td>
<td>0x415</td>
<td>GM2: same tie points.</td>
</tr>
<tr>
<td>GM_SETUP_EQUALS_TIE_POINT</td>
<td>1046</td>
<td>0x416</td>
<td>GM2: sta/tie point same.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TMC 1280 0x500</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>TMC_NO_FULLCORRECTION</td>
<td>1283</td>
<td>0x503</td>
<td>Warning: measurement without full correction</td>
</tr>
<tr>
<td>TMC_ACCURACY_GUARANTEE</td>
<td>1284</td>
<td>0x504</td>
<td>Info : accuracy can not be guarantee</td>
</tr>
<tr>
<td>TMC_ANGLE_OK</td>
<td>1285</td>
<td>0x505</td>
<td>Warning: only angle measurement valid</td>
</tr>
<tr>
<td>TMC_ANGLE_NOFULLCORRECTION</td>
<td>1288</td>
<td>0x508</td>
<td>Warning: only angle measurement valid but without full correction</td>
</tr>
<tr>
<td>TMC_ANGLE_ACCURACY_GUARANTEE</td>
<td>1289</td>
<td>0x509</td>
<td>Info : only angle measurement valid but accuracy can not be guarantee</td>
</tr>
<tr>
<td>TMC_ANGLE_ERROR</td>
<td>1290</td>
<td>0x50A</td>
<td>Error : no angle measurement</td>
</tr>
<tr>
<td>TMC_DIST_PPM</td>
<td>1291</td>
<td>0x50B</td>
<td>Error : wrong setting of PPM or MM on EDM</td>
</tr>
</tbody>
</table>
## GeoBASIC Reference Manual Appendix F — System Return Codes

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMC_DIST_ERROR</td>
<td>1292</td>
<td>0x50C</td>
<td>Error: distance measurement not done (no aim, etc.)</td>
</tr>
<tr>
<td>TMC_BUSY</td>
<td>1293</td>
<td>0x50D</td>
<td>Error: system is busy (no measurement done)</td>
</tr>
<tr>
<td>TMC_SIGNAL_ERROR</td>
<td>1294</td>
<td>0x50E</td>
<td>Error: no signal on EDM (only in signal mode)</td>
</tr>
</tbody>
</table>

### MEM 1536 0x600

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEM_OUT_OF_MEMORY</td>
<td>1536</td>
<td>0x600</td>
<td>out of memory</td>
</tr>
<tr>
<td>MEM_OUT_OF_HANDLES</td>
<td>1537</td>
<td>0x601</td>
<td>out of memory handles</td>
</tr>
<tr>
<td>MEM_TAB_OVERFLOW</td>
<td>1538</td>
<td>0x602</td>
<td>memory table overflow</td>
</tr>
<tr>
<td>MEM_HANDLE_INVALID</td>
<td>1539</td>
<td>0x603</td>
<td>used handle is invalid</td>
</tr>
<tr>
<td>MEM_DATA_NOT_FOUND</td>
<td>1540</td>
<td>0x604</td>
<td>memory data not found</td>
</tr>
<tr>
<td>MEM_DELETE_ERROR</td>
<td>1541</td>
<td>0x605</td>
<td>memory delete error</td>
</tr>
<tr>
<td>MEM_ZERO_ALLOC_ERR</td>
<td>1542</td>
<td>0x606</td>
<td>tried to allocate 0 bytes</td>
</tr>
<tr>
<td>MEM_REORG_ERR</td>
<td>1543</td>
<td>0x607</td>
<td>can't reorganize memory</td>
</tr>
</tbody>
</table>

### MOT 1792 0x700

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOT_RC_UNREADY</td>
<td>1792</td>
<td>0x700</td>
<td>Motorization not ready</td>
</tr>
<tr>
<td>MOT_RC_BUSY</td>
<td>1793</td>
<td>0x701</td>
<td>Motorization is handling another task</td>
</tr>
<tr>
<td>MOT_RC_NOT_OCONST</td>
<td>1794</td>
<td>0x702</td>
<td>Not in velocity mode</td>
</tr>
<tr>
<td>MOT_RC_NOT_CONFIG</td>
<td>1795</td>
<td>0x703</td>
<td>Motorization is in the wrong mode or busy</td>
</tr>
<tr>
<td>MOT_RC_NOT_POSIT</td>
<td>1796</td>
<td>0x704</td>
<td>Not in posit mode</td>
</tr>
<tr>
<td>MOT_RC_NOT_SERVICE</td>
<td>1797</td>
<td>0x705</td>
<td>Not in service mode</td>
</tr>
<tr>
<td>MOT_RC_NOT_BUSY</td>
<td>1798</td>
<td>0x706</td>
<td>Motorization is handling no task</td>
</tr>
<tr>
<td>MOT_RC_NOT_LOCK</td>
<td>1799</td>
<td>0x707</td>
<td>Not in tracking mode</td>
</tr>
<tr>
<td>MOT_RC_NOT_SPIRAL</td>
<td>1800</td>
<td>0x708</td>
<td>Not in spiral mode</td>
</tr>
</tbody>
</table>
### LDR 2048 0x800

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDR_PENDING</td>
<td>2048</td>
<td>0x800</td>
<td>Transfer is already open</td>
</tr>
<tr>
<td>LDR_PRGM_OCC</td>
<td>2049</td>
<td>0x801</td>
<td>Maximal number of applications reached</td>
</tr>
<tr>
<td>LDR_TRANSFER_ILLEGAL</td>
<td>2050</td>
<td>0x802</td>
<td>No Transfer is open</td>
</tr>
<tr>
<td>LDR_NOT_FOUND</td>
<td>2051</td>
<td>0x803</td>
<td>Function or program not found</td>
</tr>
<tr>
<td>LDR_ALREADY_EXIST</td>
<td>2052</td>
<td>0x804</td>
<td>Loadable object already exists</td>
</tr>
<tr>
<td>LDR_NOT_EXIST</td>
<td>2053</td>
<td>0x805</td>
<td>Can't delete. Object does not exist</td>
</tr>
<tr>
<td>LDR_SIZE_ERROR</td>
<td>2054</td>
<td>0x806</td>
<td>Error in loading object</td>
</tr>
<tr>
<td>LDR_MEM_ERROR</td>
<td>2055</td>
<td>0x807</td>
<td>Error at memory allocation/release</td>
</tr>
<tr>
<td>LDR_PRGM_NOT_EXIST</td>
<td>2056</td>
<td>0x808</td>
<td>Can't load text-object because application does not exist</td>
</tr>
<tr>
<td>LDR_FUNC_LEVEL_ERR</td>
<td>2057</td>
<td>0x809</td>
<td>Call-stack limit reached</td>
</tr>
<tr>
<td>LDR_RECURSIV_ERR</td>
<td>2058</td>
<td>0x80A</td>
<td>Recursive calling of an loaded function</td>
</tr>
<tr>
<td>LDR_INST_ERR</td>
<td>2059</td>
<td>0x80B</td>
<td>Error in installation function</td>
</tr>
<tr>
<td>LDR_FUNC_OCC</td>
<td>2060</td>
<td>0x80C</td>
<td>Maximal number of functions reached</td>
</tr>
<tr>
<td>LDR_RUN_ERROR</td>
<td>2061</td>
<td>0x80D</td>
<td>Error during a loaded application program</td>
</tr>
<tr>
<td>LDR_DEL_MENU_ERR</td>
<td>2062</td>
<td>0x80E</td>
<td>Error during deleting of menu entries of an application</td>
</tr>
<tr>
<td>LDR_OBJ_TYPE_ERROR</td>
<td>2063</td>
<td>0x80F</td>
<td>Loadable object is unknown</td>
</tr>
<tr>
<td>LDR_WRONG_SECKEY</td>
<td>2064</td>
<td>0x810</td>
<td>Wrong security key</td>
</tr>
<tr>
<td>LDR_ILLEGAL_LOADADR</td>
<td>2065</td>
<td>0x811</td>
<td>Illegal application memory address</td>
</tr>
<tr>
<td>LDR_IEEE_ERROR</td>
<td>2066</td>
<td>0x812</td>
<td>Loadable object file is not IEEE format</td>
</tr>
<tr>
<td>LDR_WRONG_APPL_VERSION</td>
<td>2067</td>
<td>0x813</td>
<td>Bad application version number</td>
</tr>
</tbody>
</table>

### BMM 2304 0x900

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMM_XFER_PENDING</td>
<td>2305</td>
<td>0x901</td>
<td>Loading process already opened</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------</td>
<td>-------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>BMM_NO_XFER_OPEN</td>
<td>2306</td>
<td>0x902</td>
<td>Transfer not opened</td>
</tr>
<tr>
<td>BMM_INVALID_LAYERS</td>
<td>2319</td>
<td>0x90F</td>
<td>Layer cannot be deleted</td>
</tr>
<tr>
<td>BMM_UNKNOWN_CHARSET</td>
<td>2307</td>
<td>0x903</td>
<td>Unknown character set</td>
</tr>
<tr>
<td>BMM_NOT_INSTALLED</td>
<td>2308</td>
<td>0x904</td>
<td>Display module not present</td>
</tr>
<tr>
<td>BMM_ALREADY_EXIST</td>
<td>2309</td>
<td>0x905</td>
<td>Character set already exists</td>
</tr>
<tr>
<td>BMM_CANT_DELETE</td>
<td>2310</td>
<td>0x906</td>
<td>Character set cannot be deleted</td>
</tr>
<tr>
<td>BMM_MEM_ERROR</td>
<td>2311</td>
<td>0x907</td>
<td>Memory cannot be allocated</td>
</tr>
<tr>
<td>BMM_CHARSET_USED</td>
<td>2312</td>
<td>0x908</td>
<td>Character set still used</td>
</tr>
<tr>
<td>BMM_CHARSET_SAVED</td>
<td>2313</td>
<td>0x909</td>
<td>Character set cannot be deleted or is protected</td>
</tr>
<tr>
<td>BMM_INVALID_ADR</td>
<td>2314</td>
<td>0x90A</td>
<td>Attempt to copy a character block outside the allocated memory</td>
</tr>
<tr>
<td>BMM_INVALID_SIZE</td>
<td>2316</td>
<td>0x90C</td>
<td>Number of bytes specified in header does not match the bytes read</td>
</tr>
<tr>
<td>BMM_INETVALID_SIZE</td>
<td>2317</td>
<td>0x90D</td>
<td>Allocated memory could not be released</td>
</tr>
<tr>
<td>BMM_ALL_GROUP_OCC</td>
<td>2318</td>
<td>0x90E</td>
<td>Max. number of character sets already loaded</td>
</tr>
<tr>
<td>TXT_OTHER_LANG</td>
<td>2560</td>
<td>0xA00</td>
<td>text found, but in an other language</td>
</tr>
<tr>
<td>TXT_UNDEF_TOKEN</td>
<td>2561</td>
<td>0xA01</td>
<td>text not found, token is undefined</td>
</tr>
<tr>
<td>TXT_UNDEF_LANG</td>
<td>2562</td>
<td>0xA02</td>
<td>language is not defined</td>
</tr>
<tr>
<td>TXT_TOOMANY_LANG</td>
<td>2563</td>
<td>0xA03</td>
<td>maximal number of languages reached</td>
</tr>
<tr>
<td>TXT_GROUP_OCC</td>
<td>2564</td>
<td>0xA04</td>
<td>desired text group is already in use</td>
</tr>
<tr>
<td>TXT_INVALID_GROUP</td>
<td>2565</td>
<td>0xA05</td>
<td>text group is invalid</td>
</tr>
<tr>
<td>TXT_OUT_OF_MEM</td>
<td>2566</td>
<td>0xA06</td>
<td>out of text memory</td>
</tr>
<tr>
<td>TXT_MEM_ERROR</td>
<td>2567</td>
<td>0xA07</td>
<td>memory write / allocate error</td>
</tr>
<tr>
<td>TXT_TRANSFER_PENDING</td>
<td>2568</td>
<td>0xA08</td>
<td>text transfer is already open</td>
</tr>
<tr>
<td>TXT_TRANSFER_ILLEGAL</td>
<td>2569</td>
<td>0xA09</td>
<td>text transfer is not opened</td>
</tr>
<tr>
<td>TXT_INVALID_SIZE</td>
<td>2570</td>
<td>0xA0A</td>
<td>illegal text data size</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>TXT_ALREADY_EXIST</td>
<td>2571</td>
<td>0xA0B</td>
<td>language already exists</td>
</tr>
<tr>
<td>MMI_BUTTON_ID.Exists</td>
<td>2817</td>
<td>0xB01</td>
<td>Button ID already exists</td>
</tr>
<tr>
<td>MMI_DLG_NOT_OPEN</td>
<td>2818</td>
<td>0xB02</td>
<td>Dialog not open</td>
</tr>
<tr>
<td>MMI_DLG_OPEN</td>
<td>2819</td>
<td>0xB03</td>
<td>Dialog already open</td>
</tr>
<tr>
<td>MMI_DLG_SPEC_MISMATCH</td>
<td>2820</td>
<td>0xB04</td>
<td>Number of fields specified with OpenDialogDef does not match</td>
</tr>
<tr>
<td>MMI_DLGDEF_EMPTY</td>
<td>2821</td>
<td>0xB05</td>
<td>Empty dialog definition</td>
</tr>
<tr>
<td>MMI_DLGDEF_NOT_OPEN</td>
<td>2822</td>
<td>0xB06</td>
<td>Dialog definition not open</td>
</tr>
<tr>
<td>MMI_DLGDEF_OPEN</td>
<td>2823</td>
<td>0xB07</td>
<td>Dialog definition still open</td>
</tr>
<tr>
<td>MMI_FIELD_ID_EXISTS</td>
<td>2824</td>
<td>0xB08</td>
<td>Field ID already exists</td>
</tr>
<tr>
<td>MMI_ILLEGAL_APP_ID</td>
<td>2825</td>
<td>0xB09</td>
<td>Illegal application ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_BUTTON_ID</td>
<td>2826</td>
<td>0xB0A</td>
<td>Illegal button ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_DLG_ID</td>
<td>2827</td>
<td>0xB0B</td>
<td>Illegal dialog ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_FIELD_COORDS</td>
<td>2828</td>
<td>0xB0C</td>
<td>Illegal field coordinates or length/height</td>
</tr>
<tr>
<td>MMI_ILLEGAL_FIELD_ID</td>
<td>2829</td>
<td>0xB0D</td>
<td>Illegal field ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_FIELD_TYPE</td>
<td>2830</td>
<td>0xB0E</td>
<td>Illegal field type</td>
</tr>
<tr>
<td>MMI_ILLEGAL_FIELD_FORMAT</td>
<td>2831</td>
<td>0xB0F</td>
<td>Illegal field format</td>
</tr>
<tr>
<td>MMI_ILLEGAL_FIXLINES</td>
<td>2832</td>
<td>0xB10</td>
<td>Illegal number of fix dialog lines</td>
</tr>
<tr>
<td>MMI_ILLEGAL_MB_TYPE</td>
<td>2833</td>
<td>0xB11</td>
<td>Illegal message box type</td>
</tr>
<tr>
<td>MMI_ILLEGAL_MENU_ID</td>
<td>2834</td>
<td>0xB12</td>
<td>Illegal menu ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_MENUITEM_ID</td>
<td>2835</td>
<td>0xB13</td>
<td>Illegal menu item ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_NEXT_ID</td>
<td>2836</td>
<td>0xB14</td>
<td>Illegal next field ID</td>
</tr>
<tr>
<td>MMI_ILLEGAL_TOPLINE</td>
<td>2837</td>
<td>0xB15</td>
<td>Illegal topline number</td>
</tr>
<tr>
<td>MMI_NOMORE_BUTTONS</td>
<td>2838</td>
<td>0xB16</td>
<td>No more buttons per dialog/menu available</td>
</tr>
<tr>
<td>MMI_NOMORE_DLGS</td>
<td>2839</td>
<td>0xB17</td>
<td>No more dialogs available</td>
</tr>
<tr>
<td>MMI_NOMORE_FIELDS</td>
<td>2840</td>
<td>0xB18</td>
<td>No more fields per dialog available</td>
</tr>
<tr>
<td>MMI_NOMORE_MENUS</td>
<td>2841</td>
<td>0xB19</td>
<td>No more menus available</td>
</tr>
<tr>
<td>MMI_NOMORE_MENUITEMS</td>
<td>2842</td>
<td>0xB1A</td>
<td>No more menu items available</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
<td>--------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>MMI_NOMORE_WINDOWS</td>
<td>2843</td>
<td>0xB1B</td>
<td>No more windows available</td>
</tr>
<tr>
<td>MMI_SYS_BUTTON</td>
<td>2844</td>
<td>0xB1C</td>
<td>The button belongs to the MMI</td>
</tr>
<tr>
<td>MMI_VREF_UNDEF</td>
<td>2845</td>
<td>0xB1D</td>
<td>The parameter list for OpenDialog is uninitialized</td>
</tr>
<tr>
<td>MMI_EXIT_DLG</td>
<td>2846</td>
<td>0xB1E</td>
<td>The MMI should exit the dialog</td>
</tr>
<tr>
<td>MMI_KEEP_FOCUS</td>
<td>2847</td>
<td>0xB1F</td>
<td>The MMI should keep focus within field being edited</td>
</tr>
<tr>
<td>MMI_NOMORE_ITEMS</td>
<td>2848</td>
<td>0xB20</td>
<td>Notification to the MMI that no more items available</td>
</tr>
<tr>
<td>COM 2850</td>
<td>0xC00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RC_COM_ERO</td>
<td>3072</td>
<td>0xC00</td>
<td>Initiate Extended Runtime Operation (ERO).</td>
</tr>
<tr>
<td>RC_COM_CANT_ENCODE</td>
<td>3073</td>
<td>0xC01</td>
<td>Cannot encode arguments in client.</td>
</tr>
<tr>
<td>RC_COM_CANT_DECODE</td>
<td>3074</td>
<td>0xC02</td>
<td>Cannot decode results in client.</td>
</tr>
<tr>
<td>RC_COM_CANT_SEND</td>
<td>3075</td>
<td>0xC03</td>
<td>Hardware error while sending.</td>
</tr>
<tr>
<td>RC_COM_CANT_RECV</td>
<td>3076</td>
<td>0xC04</td>
<td>Hardware error while receiving.</td>
</tr>
<tr>
<td>RC_COM_TIMEDOUT</td>
<td>3077</td>
<td>0xC05</td>
<td>Request timed out.</td>
</tr>
<tr>
<td>RC_COM_WRONG_FFORMAT</td>
<td>3078</td>
<td>0xC06</td>
<td>Packet format error.</td>
</tr>
<tr>
<td>RC_COM_VER_MISMATCH</td>
<td>3079</td>
<td>0xC07</td>
<td>Version mismatch between client and server.</td>
</tr>
<tr>
<td>RC_COM_CANT_DECODE_REQ</td>
<td>3080</td>
<td>0xC08</td>
<td>Cannot decode arguments in server.</td>
</tr>
<tr>
<td>RC_COM_PROC_UNAVAIL</td>
<td>3081</td>
<td>0xC09</td>
<td>Unknown RPC, procedure ID invalid.</td>
</tr>
<tr>
<td>RC_COM_CANT_ENCODE_REP</td>
<td>3082</td>
<td>0xC0A</td>
<td>Cannot encode results in server.</td>
</tr>
<tr>
<td>RC_COM_SYSTEM_ERR</td>
<td>3083</td>
<td>0xC0B</td>
<td>Unspecified generic system error.</td>
</tr>
<tr>
<td>RC_COM_FAILED</td>
<td>3085</td>
<td>0xC0D</td>
<td>Unspecified error.</td>
</tr>
<tr>
<td>RC_COM_NO_BINARY</td>
<td>3086</td>
<td>0xC0E</td>
<td>Binary protocol not available.</td>
</tr>
<tr>
<td>RC_COM_INTR</td>
<td>3087</td>
<td>0xC0F</td>
<td>Call interrupted.</td>
</tr>
<tr>
<td>RC_COMQUIRES_8DBITS</td>
<td>3090</td>
<td>0xC12</td>
<td>Protocol needs 8bit encoded characters.</td>
</tr>
<tr>
<td>RC_COM_TR_ID_MISMATCH</td>
<td>3093</td>
<td>0xC15</td>
<td>Transaction ID mismatch error.</td>
</tr>
<tr>
<td>RC_COM_NOT_GEOCOM</td>
<td>3094</td>
<td>0xC16</td>
<td>Protocol not recognizable.</td>
</tr>
</tbody>
</table>
### GeoBASIC Reference Manual

**Appendix F — System Return Codes**

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_COM_UNKNOWN_PORT</td>
<td>3095</td>
<td>0xC17</td>
<td>(WIN) Invalid port address.</td>
</tr>
<tr>
<td>RC_COM_ERO_END</td>
<td>3099</td>
<td>0xC1B</td>
<td>ERO is terminating.</td>
</tr>
<tr>
<td>RC_COM_OVERRUN</td>
<td>3100</td>
<td>0xC1C</td>
<td>Internal error: data buffer overflow.</td>
</tr>
<tr>
<td>RX_CHECKSUM_ERROR</td>
<td>3101</td>
<td>0xC1D</td>
<td>Invalid checksum on server side received.</td>
</tr>
<tr>
<td>RC_COM_CLNT_RX_CHECKSUM_ERROR</td>
<td>3102</td>
<td>0xC1E</td>
<td>Invalid checksum on client side received.</td>
</tr>
<tr>
<td>RC_COM_PORT_NOT_AVAILABLE</td>
<td>3103</td>
<td>0xC1F</td>
<td>(WIN) Port not available.</td>
</tr>
<tr>
<td>RC_COM_PORT_NOT_OPEN</td>
<td>3104</td>
<td>0xC20</td>
<td>(WIN) Port not opened.</td>
</tr>
<tr>
<td>RC_COM_NO_PARTNER</td>
<td>3105</td>
<td>0xC21</td>
<td>(WIN) Unable to find TPS.</td>
</tr>
<tr>
<td>RC_COM_ERO_NOT_STARTED</td>
<td>3106</td>
<td>0xC22</td>
<td>Extended Runtime Operation could not be started.</td>
</tr>
<tr>
<td>RC_COM_CONS_REQ</td>
<td>3107</td>
<td>0xC23</td>
<td>Att to send cons reqs</td>
</tr>
<tr>
<td>RC_COM_SRVR_IS_SLEEPING</td>
<td>3108</td>
<td>0xC24</td>
<td>TPS has gone to sleep. Wait and try again.</td>
</tr>
<tr>
<td>RC_COM_SRVR_IS_OFF</td>
<td>3109</td>
<td>0xC25</td>
<td>TPS has shut down. Wait and try again.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DPL 3328 0xD00</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPL_RC_NOCREATE</td>
<td>3328</td>
<td>0xD00</td>
<td>no file creation, fatal</td>
</tr>
<tr>
<td>DPL_RC_NOTOPEN</td>
<td>3329</td>
<td>0xD01</td>
<td>bank not open</td>
</tr>
<tr>
<td>DPL_RC_ALRDYOPEN</td>
<td>3330</td>
<td>0xD02</td>
<td>a databank is already open</td>
</tr>
<tr>
<td>DPL_RC_NOTFOUND</td>
<td>3331</td>
<td>0xD03</td>
<td>databank file does not exist</td>
</tr>
<tr>
<td>DPL_RC_EXISTS</td>
<td>3332</td>
<td>0xD04</td>
<td>databank already exists</td>
</tr>
<tr>
<td>DPL_RC_EMPTY</td>
<td>3333</td>
<td>0xD05</td>
<td>databank is empty</td>
</tr>
<tr>
<td>DPL_RC_BADATA</td>
<td>3334</td>
<td>0xD06</td>
<td>bad data detected</td>
</tr>
<tr>
<td>DPL_RC_BADFIELD</td>
<td>3335</td>
<td>0xD07</td>
<td>bad field type</td>
</tr>
<tr>
<td>DPL_RC_BADINDEX</td>
<td>3336</td>
<td>0xD08</td>
<td>bad index information</td>
</tr>
<tr>
<td>DPL_RC_BADKEY</td>
<td>3337</td>
<td>0xD09</td>
<td>bad key type</td>
</tr>
<tr>
<td>DPL_RC_BADMODE</td>
<td>3338</td>
<td>0xD0A</td>
<td>bad mode</td>
</tr>
<tr>
<td>DPL_RC_BADRANGE</td>
<td>3339</td>
<td>0xD0B</td>
<td>bad range</td>
</tr>
<tr>
<td>DPL_RC_DUPLICATE</td>
<td>3340</td>
<td>0xD0C</td>
<td>duplicate keys not allowed</td>
</tr>
<tr>
<td>DPL_RC_INCOMPLETE</td>
<td>3341</td>
<td>0xD0D</td>
<td>record is incomplete</td>
</tr>
<tr>
<td>DPL_RC_INVDBID</td>
<td>3342</td>
<td>0xD0E</td>
<td>invalid db project id</td>
</tr>
<tr>
<td>DPL_RC_IVNAME</td>
<td>3343</td>
<td>0xD0F</td>
<td>invalid name</td>
</tr>
<tr>
<td>DPL_RC_LOCKED</td>
<td>3344</td>
<td>0xD10</td>
<td>data locked</td>
</tr>
<tr>
<td>DPL_RC_NOTLOCKED</td>
<td>3345</td>
<td>0xD11</td>
<td>data not locked</td>
</tr>
</tbody>
</table>

F-12

TPS1100-Version 1.30
<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPL_RC_NODATA</td>
<td>3346</td>
<td>0xD12</td>
<td>no data found</td>
</tr>
<tr>
<td>DPL_RC_NOMATCH</td>
<td>3347</td>
<td>0xD13</td>
<td>no matching key found</td>
</tr>
<tr>
<td>DPL_RC_NOSPACE</td>
<td>3348</td>
<td>0xD14</td>
<td>no more (disk) space left</td>
</tr>
<tr>
<td>DPL_RC_NOCLOSE</td>
<td>3349</td>
<td>0xD15</td>
<td>could not close db (sys. error)</td>
</tr>
<tr>
<td>DPL_RC_RELATIONS</td>
<td>3350</td>
<td>0xD16</td>
<td>record still has relations</td>
</tr>
<tr>
<td>DPL_RC_NULLPTR</td>
<td>3351</td>
<td>0xD17</td>
<td>null pointer</td>
</tr>
<tr>
<td>DPL_RC_BADFORMAT</td>
<td>3352</td>
<td>0xD18</td>
<td>bad databank format, wrong version</td>
</tr>
<tr>
<td>DPL_RC_BADRECTYPE</td>
<td>3353</td>
<td>0xD19</td>
<td>bad record type</td>
</tr>
<tr>
<td>DPL_RC_OUTOFMEM</td>
<td>3354</td>
<td>0xD1A</td>
<td>no more (memory) space left</td>
</tr>
<tr>
<td>DPL_RC_CODE_</td>
<td>3355</td>
<td>0xD1B</td>
<td>code mismatch</td>
</tr>
<tr>
<td>DPL_RC_NOTINIT</td>
<td>3356</td>
<td>0xD1C</td>
<td>db has not been initialized</td>
</tr>
<tr>
<td>DPL_RC_NOTEXIST</td>
<td>3357</td>
<td>0xD1D</td>
<td>trf. for old db's does not exist</td>
</tr>
<tr>
<td>DPL_RC_NOTOK</td>
<td>4864</td>
<td>0x1300</td>
<td>not ok</td>
</tr>
<tr>
<td>DPL_RC_IVAPPL</td>
<td>4865</td>
<td>0x1301</td>
<td>invalid database system appl.</td>
</tr>
<tr>
<td>DPL_RC_NOTAVAILABLE</td>
<td>4866</td>
<td>0x1302</td>
<td>database not available</td>
</tr>
<tr>
<td>DPL_RC_NO_CODELIST</td>
<td>4867</td>
<td>0x1303</td>
<td>no codelist found</td>
</tr>
<tr>
<td>DPL_RC_TO_MANY_</td>
<td>4868</td>
<td>0x1304</td>
<td>more then DPL_MAX_CODELISTS found</td>
</tr>
</tbody>
</table>

**FIL**

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_FIL_NO_ERROR</td>
<td>3840</td>
<td>0xF00</td>
<td>Operation completed successfully.</td>
</tr>
<tr>
<td>RC_FIL_FILENAME_</td>
<td>3845</td>
<td>0xF05</td>
<td>File name not found.</td>
</tr>
<tr>
<td>RC_FIL_NO_MAKE_</td>
<td>3880</td>
<td>0xF28</td>
<td>Cannot create directory.</td>
</tr>
<tr>
<td>RC_FIL_RENAME_</td>
<td>3886</td>
<td>0xF2E</td>
<td>Rename of file failed.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_PATH</td>
<td>3888</td>
<td>0xF30</td>
<td>Invalid path specified.</td>
</tr>
<tr>
<td>RC_FIL_FILE_</td>
<td>3898</td>
<td>0xF3A</td>
<td>Cannot delete file.</td>
</tr>
<tr>
<td>RC_FIL_ILLEGAL_ORIGIN</td>
<td>3906</td>
<td>0xF42</td>
<td>Illegal origin.</td>
</tr>
<tr>
<td>RC_FIL_END_OF_FILE</td>
<td>3924</td>
<td>0xF54</td>
<td>End of file reached.</td>
</tr>
<tr>
<td>RC_FIL_NO_MORE_</td>
<td>3931</td>
<td>0xF5B</td>
<td>Medium full.</td>
</tr>
<tr>
<td>RC_FIL_PATTERN_</td>
<td>3932</td>
<td>0xF5C</td>
<td>Pattern does not match file names.</td>
</tr>
<tr>
<td>RC_FIL_FILE_ALREADY_</td>
<td>3948</td>
<td>0xF6C</td>
<td>File is already open with write</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OPENED_FOR_WR</td>
<td></td>
<td></td>
<td>permission.</td>
</tr>
<tr>
<td>RC_FIL_WRITE_TO_MEDIUM_FAILED</td>
<td>3957</td>
<td>0xF75</td>
<td>Write operation to medium failed.</td>
</tr>
<tr>
<td>RC_FIL_START_SEARCH_NOT_CALLED</td>
<td>3963</td>
<td>0xF7B</td>
<td>FIL_StartList not called.</td>
</tr>
<tr>
<td>RC_FIL_NO_STORAGE_MEDIUM_IN_DEVICE</td>
<td>3964</td>
<td>0xF7C</td>
<td>No medium existent in device.</td>
</tr>
<tr>
<td>RC_FIL_ILLEGAL_FILE_OPEN_TYPE</td>
<td>3965</td>
<td>0xF7D</td>
<td>Illegal file open type.</td>
</tr>
<tr>
<td>RC_FIL_MEDIUM_NEWLY_INSERTED</td>
<td>3966</td>
<td>0xF7E</td>
<td>Medium freshly inserted into device.</td>
</tr>
<tr>
<td>RC_FIL_MEMORY_FAILED</td>
<td>3967</td>
<td>0xF7F</td>
<td>Memory failure. No more memory available.</td>
</tr>
<tr>
<td>RC_FIL_FATAL_ERROR</td>
<td>3968</td>
<td>0xF80</td>
<td>Fatal error during file operation.</td>
</tr>
<tr>
<td>RC_FIL_FAT_ERROR</td>
<td>3969</td>
<td>0xF81</td>
<td>Fatal error in file allocation table.</td>
</tr>
<tr>
<td>RC_FIL_ILLEGAL_DRIVE</td>
<td>3970</td>
<td>0xF82</td>
<td>Illegal drive chosen.</td>
</tr>
<tr>
<td>RC_FIL_INVALID_FILE_DESCR</td>
<td>3971</td>
<td>0xF83</td>
<td>Illegal file descriptor.</td>
</tr>
<tr>
<td>RC_FIL_SEEK_FAILED</td>
<td>3972</td>
<td>0xF84</td>
<td>Seek failed.</td>
</tr>
<tr>
<td>RC_FIL_CANNOT_DELETE</td>
<td>3973</td>
<td>0xF85</td>
<td>Cannot delete file.</td>
</tr>
<tr>
<td>RC_FIL_MEDIUM_WRITE_PROTECTED</td>
<td>3974</td>
<td>0xF86</td>
<td>Medium is write protected.</td>
</tr>
<tr>
<td>RC_FIL_BATTERY_LOW</td>
<td>3975</td>
<td>0xF87</td>
<td>Medium backup battery is low.</td>
</tr>
<tr>
<td>RC_FIL_BAD_FORMAT</td>
<td>3976</td>
<td>0xF88</td>
<td>Bad medium format.</td>
</tr>
<tr>
<td>RC_FIL_UNSUPPORTED_MEDIUM</td>
<td>3977</td>
<td>0xF89</td>
<td>Unsupported PC-Card detected.</td>
</tr>
<tr>
<td>RC_FIL_RENAME_DIR_FAILED</td>
<td>3978</td>
<td>0xF8A</td>
<td>Directory exists already</td>
</tr>
</tbody>
</table>

**WIR**  

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WIR_PTNR_OVERFLOW</td>
<td>5121</td>
<td>0x1401</td>
<td>point number overflow</td>
</tr>
<tr>
<td>WIR_NUM_ASCII_CARRY</td>
<td>5122</td>
<td>0x1402</td>
<td>carry from number to ascii conversion</td>
</tr>
<tr>
<td>WIR_PTNR_NO_INC</td>
<td>5123</td>
<td>0x1403</td>
<td>can't increment point number</td>
</tr>
<tr>
<td>WIR_STEP_SIZE</td>
<td>5124</td>
<td>0x1404</td>
<td>wrong step size</td>
</tr>
<tr>
<td>WIR_BUSY</td>
<td>5125</td>
<td>0x1405</td>
<td>resource occupied</td>
</tr>
<tr>
<td>WIR_CONFIG_FNC</td>
<td>5127</td>
<td>0x1407</td>
<td>user function selected</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------</td>
<td>---------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>WIR_CANT_OPEN_FILE</td>
<td>5128</td>
<td>0x1408</td>
<td>can't open file</td>
</tr>
<tr>
<td>WIR_FILE_WRITE_ERROR</td>
<td>5129</td>
<td>0x1409</td>
<td>can't write into file</td>
</tr>
<tr>
<td>WIR_MEDIUM_NOMEM</td>
<td>5130</td>
<td>0x140A</td>
<td>no anymore memory on PC-Card</td>
</tr>
<tr>
<td>WIR_NO_MEDIUM</td>
<td>5131</td>
<td>0x140B</td>
<td>no PC-Card</td>
</tr>
<tr>
<td>WIR_EMPTY_FILE</td>
<td>5132</td>
<td>0x140C</td>
<td>empty GSI file</td>
</tr>
<tr>
<td>WIR_INVALID_DATA</td>
<td>5133</td>
<td>0x140D</td>
<td>invalid data in GSI file</td>
</tr>
<tr>
<td>WIR_F2_BUTTON</td>
<td>5134</td>
<td>0x140E</td>
<td>F2 button pressed</td>
</tr>
<tr>
<td>WIR_F3_BUTTON</td>
<td>5135</td>
<td>0x140F</td>
<td>F3 button pressed</td>
</tr>
<tr>
<td>WIR_F4_BUTTON</td>
<td>5136</td>
<td>0x1410</td>
<td>F4 button pressed</td>
</tr>
<tr>
<td>WIR_F5_BUTTON</td>
<td>5137</td>
<td>0x1411</td>
<td>F5 button pressed</td>
</tr>
<tr>
<td>WIR_F6_BUTTON</td>
<td>5138</td>
<td>0x1412</td>
<td>F6 button pressed</td>
</tr>
<tr>
<td>WIR_SHF2_BUTTON</td>
<td>5139</td>
<td>0x1413</td>
<td>SHIFT F2 button pressed</td>
</tr>
<tr>
<td>AUT_RC_TIMEOUT</td>
<td>8704</td>
<td>0x2200</td>
<td>Position not reached</td>
</tr>
<tr>
<td>AUT_RCDETENT_ERROR</td>
<td>8705</td>
<td>0x2201</td>
<td>Positioning not possible due to mounted EDM</td>
</tr>
<tr>
<td>AUT_RC_ANGLE_ERROR</td>
<td>8706</td>
<td>0x2202</td>
<td>Angle measurement error</td>
</tr>
<tr>
<td>AUT_RC_MOTOR_ERROR</td>
<td>8707</td>
<td>0x2203</td>
<td>Motorization error</td>
</tr>
<tr>
<td>AUT_RC_INCACC</td>
<td>8708</td>
<td>0x2204</td>
<td>Position not exactly reached</td>
</tr>
<tr>
<td>AUT_RC_DEV_ERROR</td>
<td>8709</td>
<td>0x2205</td>
<td>Deviation measurement error</td>
</tr>
<tr>
<td>AUT_RC_NO_TARGET</td>
<td>8710</td>
<td>0x2206</td>
<td>No target detected</td>
</tr>
<tr>
<td>AUT_RC_MULTIPLE_TARGETS</td>
<td>8711</td>
<td>0x2207</td>
<td>Multiple target detected</td>
</tr>
<tr>
<td>AUT_RC_BAD_ENVIRONMENT</td>
<td>8712</td>
<td>0x2208</td>
<td>Bad environment conditions</td>
</tr>
<tr>
<td>AUT_RC_DETECTOR_ERROR</td>
<td>8713</td>
<td>0x2209</td>
<td>Error in target acquisition</td>
</tr>
<tr>
<td>AUT_RC_NOT_ENABLED</td>
<td>8714</td>
<td>0x220A</td>
<td>Target acquisition not enabled</td>
</tr>
<tr>
<td>AUT_RC_CALACC</td>
<td>8715</td>
<td>0x220B</td>
<td>ATR-Calibration failed</td>
</tr>
<tr>
<td>AUT_RC_ACCURACY</td>
<td>8716</td>
<td>0x220C</td>
<td>Target position not exactly reached</td>
</tr>
<tr>
<td>AUT_RC_DIST_STARTED</td>
<td>8717</td>
<td>0x220D</td>
<td>Info: dist. Measurement has been started</td>
</tr>
</tbody>
</table>
### GeoBASIC Reference Manual

#### Appendix F — System Return Codes

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_CHANGE_ALL_TO_DIST</td>
<td>9217</td>
<td>0x2401</td>
<td>Command changed from ALL to DIST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAP_ILLEGAL_SYSMENU_NUM</td>
<td>9473</td>
<td>0x2501</td>
<td>Illegal system menu number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COD_RC_LIST_NOT_VALID</td>
<td>9728</td>
<td>0x2600</td>
<td>List not initialized.</td>
</tr>
<tr>
<td>COD_RC_SHORTCUT_UNKNOWN</td>
<td>9729</td>
<td>0x2601</td>
<td>Shortcut or code unknown.</td>
</tr>
<tr>
<td>COD_RC_NOT_SELECTED</td>
<td>9730</td>
<td>0x2602</td>
<td>Codelist selection wasn't possible.</td>
</tr>
<tr>
<td>COD_RC_MANDATORY_FAIL</td>
<td>9731</td>
<td>0x2603</td>
<td>Mandatory field has no valid value.</td>
</tr>
<tr>
<td>COD_RC_NO_MORE_ATTRIB</td>
<td>9732</td>
<td>0x2604</td>
<td>maximal number of attr. are defined.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAS_ILL_OPCODE</td>
<td>9984</td>
<td>0x2700</td>
<td>Illegal opcode.</td>
</tr>
<tr>
<td>BAS_DIV_BY_ZERO</td>
<td>9985</td>
<td>0x2701</td>
<td>Division by Zero occurred.</td>
</tr>
<tr>
<td>BAS_STACK_UNDERFLOW</td>
<td>9986</td>
<td>0x2702</td>
<td>Interpreter stack underflow.</td>
</tr>
<tr>
<td>BAS_STACK_OVERFLOW</td>
<td>9987</td>
<td>0x2703</td>
<td>Interpreter stack overflow.</td>
</tr>
<tr>
<td>BAS_NO_DLG_EXIST</td>
<td>9988</td>
<td>0x2704</td>
<td>No dialog is defined.</td>
</tr>
<tr>
<td>BAS_DLG_ALREADY</td>
<td>9989</td>
<td>0x2705</td>
<td>Only one dialog may be defined at</td>
</tr>
<tr>
<td>RetCodeName</td>
<td>Value</td>
<td>Hex</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------</td>
<td>------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>EXIST</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAS_INSTALL_ERR</td>
<td>9990</td>
<td>0x2706</td>
<td>General error during installation.</td>
</tr>
<tr>
<td>BAS_FIL_INV_MODE</td>
<td>9995</td>
<td>0x270B</td>
<td>Invalid file access mode.</td>
</tr>
<tr>
<td>BAS_FIL_TABLE_FULL</td>
<td>9996</td>
<td>0x270C</td>
<td>Maximum number of open files overflow.</td>
</tr>
<tr>
<td>BAS_FIL_ILL_NAME</td>
<td>9997</td>
<td>0x270D</td>
<td>Illegal file name.</td>
</tr>
<tr>
<td>BAS_FIL_ILL_POS</td>
<td>9998</td>
<td>0x270E</td>
<td>Illegal file position, hence &lt; 1.</td>
</tr>
<tr>
<td>BAS_FIL_ILL_OPER</td>
<td>9999</td>
<td>0x270F</td>
<td>Illegal operation on this kind of file.</td>
</tr>
<tr>
<td>BAS_MENU_ID_INVALID</td>
<td>10000</td>
<td>0x2710</td>
<td>Invalid menu id detected.</td>
</tr>
<tr>
<td>BAS_MENU_TABLE_FULL</td>
<td>10001</td>
<td>0x2711</td>
<td>Internal menu id table overflow.</td>
</tr>
<tr>
<td>IOS</td>
<td>10240</td>
<td>0x2800</td>
<td></td>
</tr>
<tr>
<td>IOS_CHNL_DISABLED</td>
<td>10240</td>
<td>0x2800</td>
<td>channel is disabled</td>
</tr>
<tr>
<td>IOS_NO_MORE_CHAR</td>
<td>10241</td>
<td>0x2801</td>
<td>no more data available</td>
</tr>
<tr>
<td>IOS_MAX_BLOCK_LEN</td>
<td>10242</td>
<td>0x2802</td>
<td>reached max. block length</td>
</tr>
<tr>
<td>IOS_HW_BUF_OVERRUN</td>
<td>10243</td>
<td>0x2803</td>
<td>hardware buffer overrun (highest priority)</td>
</tr>
<tr>
<td>IOS_PARITY_ERROR</td>
<td>10244</td>
<td>0x2804</td>
<td>parity error</td>
</tr>
<tr>
<td>IOS_FRAMING_ERROR</td>
<td>10245</td>
<td>0x2805</td>
<td>framing error</td>
</tr>
<tr>
<td>IOS_DECODE_ERROR</td>
<td>10246</td>
<td>0x2806</td>
<td>decode error</td>
</tr>
<tr>
<td>IOS_CHKSUM_ERROR</td>
<td>10247</td>
<td>0x2807</td>
<td>checksum error (lowest priority)</td>
</tr>
<tr>
<td>IOS_COM_ERROR</td>
<td>10248</td>
<td>0x2808</td>
<td>general communication error</td>
</tr>
<tr>
<td>IOS_FL_RD_ERROR</td>
<td>10280</td>
<td>0x2828</td>
<td>flash read error</td>
</tr>
<tr>
<td>IOS_FL_WR_ERROR</td>
<td>10281</td>
<td>0x2829</td>
<td>flash write error</td>
</tr>
<tr>
<td>IOS_FL_CL_ERROR</td>
<td>10282</td>
<td>0x282A</td>
<td>flash erase error</td>
</tr>
<tr>
<td>CNF</td>
<td>10496</td>
<td>0x2900</td>
<td></td>
</tr>
<tr>
<td>CNF_INI_NOTOPEN</td>
<td>10497</td>
<td>0x2901</td>
<td>INI-file not opened</td>
</tr>
<tr>
<td>CNF_INI_NOTFOUND</td>
<td>10498</td>
<td>0x2902</td>
<td>Warning: Could not find section or key</td>
</tr>
<tr>
<td>CNF_CONT</td>
<td>10499</td>
<td>0x2903</td>
<td>Return code of system function</td>
</tr>
<tr>
<td>CNF_ESC</td>
<td>10500</td>
<td>0x2904</td>
<td>Return code of system function</td>
</tr>
<tr>
<td>CNF.Quit</td>
<td>10501</td>
<td>0x2905</td>
<td>Return code of system function</td>
</tr>
<tr>
<td>CNF_DATA_INVALID</td>
<td>10502</td>
<td>0x2906</td>
<td>Config. file data not valid</td>
</tr>
</tbody>
</table>

GeoBASIC Reference Manual  Appendix F — System Return Codes
<table>
<thead>
<tr>
<th>RetCodeName</th>
<th>Value</th>
<th>Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNF_DATA_OVERFLOW</td>
<td>10503</td>
<td>0x2907</td>
<td>Config. file data exceed valid amount</td>
</tr>
<tr>
<td>CNF_NOT_COMPLETE</td>
<td>10504</td>
<td>0x2908</td>
<td>Config. file data not complete</td>
</tr>
<tr>
<td>CNF_DLGL_CNT_OVERFLOW</td>
<td>10505</td>
<td>0x2909</td>
<td>Too many executed dialogs</td>
</tr>
<tr>
<td>CNF_NOT_EXECUTABLE</td>
<td>10506</td>
<td>0x290A</td>
<td>Item not executable</td>
</tr>
<tr>
<td>CNF_AEXE_OVERFLOW</td>
<td>10507</td>
<td>0x290B</td>
<td>Autoexec table full</td>
</tr>
<tr>
<td>CNF_PAR_LOAD_ERR</td>
<td>10508</td>
<td>0x290C</td>
<td>Error in loading parameter</td>
</tr>
<tr>
<td>CNF_PAR_SAVE_ERR</td>
<td>10509</td>
<td>0x290D</td>
<td>Error in saving parameter</td>
</tr>
<tr>
<td>CNF_FILE_MISSING</td>
<td>10510</td>
<td>0x290E</td>
<td>Parameter filename/path not valid</td>
</tr>
<tr>
<td>CNF_SECTION_MISSING</td>
<td>10511</td>
<td>0x290F</td>
<td>Section in parameter file missing</td>
</tr>
<tr>
<td>CNF_HEADER_FAIL</td>
<td>10512</td>
<td>0x2910</td>
<td>Default file wrong or an entry is missing</td>
</tr>
<tr>
<td>CNF_PARMETER_FAIL</td>
<td>10513</td>
<td>0x2911</td>
<td>Parameter-line not complete or missing</td>
</tr>
<tr>
<td>CNF_PARMETER_SET</td>
<td>10514</td>
<td>0x2912</td>
<td>Parameter-set caused an error</td>
</tr>
<tr>
<td>CNF_RECMASK_FAIL</td>
<td>10515</td>
<td>0x2913</td>
<td>RecMask-line not complete or missing</td>
</tr>
<tr>
<td>CNF_RECMASK_SET</td>
<td>10516</td>
<td>0x2914</td>
<td>RecMask-set caused an error</td>
</tr>
<tr>
<td>CNF_MEASDLGLIST_FAIL</td>
<td>10517</td>
<td>0x2915</td>
<td>MeasDlgList-line not complete or missing</td>
</tr>
<tr>
<td>CNF_MEASDLGLIST_SET</td>
<td>10518</td>
<td>0x2916</td>
<td>MeasDlgList-set caused an error</td>
</tr>
<tr>
<td>CNF_APPL_OVERFLOW</td>
<td>10519</td>
<td>0x2917</td>
<td>Application table full</td>
</tr>
</tbody>
</table>
## Appendix G — Geodesy Mathematical Formulas

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.1</td>
<td>Generally</td>
</tr>
<tr>
<td>G.2</td>
<td>Conversion of angle</td>
</tr>
<tr>
<td>G.2.1</td>
<td>Generally</td>
</tr>
<tr>
<td>G.2.2</td>
<td>Conversion decimal-sexagesimal</td>
</tr>
<tr>
<td>G.2.3</td>
<td>Conversion sexagesimal-decimal</td>
</tr>
<tr>
<td>G.3</td>
<td>Conversion of Distance</td>
</tr>
<tr>
<td>G.4</td>
<td>Physical Conversion</td>
</tr>
<tr>
<td>G.5</td>
<td>Calculation of average</td>
</tr>
<tr>
<td>G.5.1</td>
<td>Generally</td>
</tr>
<tr>
<td>G.5.2</td>
<td>Calculation of average for directions</td>
</tr>
<tr>
<td>G.5.3</td>
<td>Calculation of median for directions</td>
</tr>
<tr>
<td>G.6</td>
<td>Calculation of coordinate</td>
</tr>
<tr>
<td>G.6.1</td>
<td>Calculation of azimuth and distance result from coordinate</td>
</tr>
<tr>
<td>G.6.2</td>
<td>Calculation of coordinate result from azimuth and distance</td>
</tr>
<tr>
<td>G.6.3</td>
<td>Conversion polar - rectangular</td>
</tr>
<tr>
<td>G.6.4</td>
<td>Conversion rectangular - polar</td>
</tr>
<tr>
<td>G.6.5</td>
<td>Calculation of zenith angle and slope distance as a result from coordinate</td>
</tr>
<tr>
<td>G.7</td>
<td>Transformation of coordinate</td>
</tr>
<tr>
<td>G.7.1</td>
<td>of mathematical coordinate systems</td>
</tr>
<tr>
<td>G.7.2</td>
<td>of geodetical coordinate systems</td>
</tr>
<tr>
<td>G.8</td>
<td>Calculation of triangle</td>
</tr>
<tr>
<td>G.8.1</td>
<td>Case SWS</td>
</tr>
<tr>
<td>G.8.2</td>
<td>Case SSS</td>
</tr>
<tr>
<td>G.8.3</td>
<td>Case SSW or WSS</td>
</tr>
<tr>
<td>G.8.4</td>
<td>Case WWS or SWW</td>
</tr>
</tbody>
</table>
G.9 Calculation of circle .................................................................G-23
  G.9.1 Radius and center result from 3 point.................................G-23

G.10 Calculation of intersection......................................................G-25
  G.10.1 Intersection line - line without parallel displacement.........G-25
  G.10.2 Intersection line - line with parallel displacement.............G-27
  G.10.3 Intersection line - circle....................................................G-28
  G.10.4 Intersection circle - circle................................................G-29

G.11 Calculation of Distance ..........................................................G-30
  G.11.1 Distance point - point ......................................................G-30
  G.11.2 Distance point - line .......................................................G-31
  G.11.3 Distance point - circle......................................................G-32
  G.11.4 Distance point - Clothoid ...............................................G-32

G.12 Calculation of the Base point of plumb line............................G-33
  G.12.1 Point on line ..................................................................G-33
  G.12.2 Point on circle ...............................................................G-34
  G.12.3 Point on Clothoid ............................................................G-35

G.13 Calculate point with distance on line......................................G-37
  G.13.1 Point with distance on line ..............................................G-37
  G.13.2 Calculate point on arc of circle with distance....................G-38

G.14 Calculation of Clothoid ..........................................................G-40
  G.14.1 Calculated Coordinate......................................................G-41

G.15 Transformation ........................................................................G-42

G.16 Planimetry ............................................................................G-44
  G.16.1 Planimetry result from coordinate (Gauss).......................G-44
  G.16.2 Planimetry result from measurement (triangle)..................G-45
  G.16.3 Segment Plane ...............................................................G-46

G.17 Excenter Observation re-centered to the Center.......................G-47
  G.17.1 Distance Measurement to the Mark.................................G-47
  G.17.2 Distance is not measured to the mark..............................G-48
G.1 GENERALLY

The formula is valid for the following sections:
- distances and height differences in meter
- angle, direction and azimuth in radiant

generally used nomenclature:
\[ \begin{align*}
&ds_{X,Y} : \text{Slope distance from point } X \text{ to point } Y \\
&dh_{X,Y} : \text{Horizontal distance from point } X \text{ to point } Y \text{ at sea level} \\
&Hz_{X,Y} : \text{Horizontal direction from point } X \text{ to point } Y \\
&V_{X,Y} : \text{Vertical direction from point } X \text{ to point } Y (\text{always means zenith distance}) \\
&Z_{X,Y} : \text{Azimuth from point } X \text{ to point } Y \\
&N_i,E_i,H_i : \text{N, E Coordinate and height at the point } P_i \\
&\Delta N_{X,Y} : \text{Coordinate difference in N-direction between point } X \text{ and point } Y \\
&\Delta E_{X,Y} : \text{Coordinate difference in E-direction between point } X \text{ and point } Y \\
&\Delta H_{X,Y} : \text{Height difference between point } X \text{ and point } Y
\end{align*} \]
mathematics functions:

- Int(x) : Function in order to calculate the integer part of the argument x
- Frac(x) : Function in order to calculate the fraction of the argument x
- Abs(x) : Function in order to calculate the absolute of the argument x
- Mod(x) : Function in order to calculate the rest of a division
- sin(x) : Function in order to calculate sine of the argument x
- cos(x) : Function in order to calculate cosine of the argument x
- tan(x) : Function in order to calculate tangent of the argument x
- asin(x) : Function in order to calculate arcs sinus of the argument x
- acos(x) : Function in order to calculate arcs cosine of the argument x
- atan(x) : Function in order to calculate arcs tangent of the argument x
G.2 CONVERSION OF ANGLE

G.2.1 Generally

Nomenclature:

GIVEN:

\( \alpha \)  
angle to convert

Formula:

Radiant in Neugrad:  
\[ f(\alpha) = \frac{200}{\pi} \times \alpha \]

Radiant in Altgrad:  
\[ f(\alpha) = \frac{180}{\pi} \times \alpha \]

Radiant in Artilleriepromille  
\[ f(\alpha) = \frac{3200}{\pi} \times \alpha \]

Neugrad in Radiant:  
\[ f(\alpha) = \frac{\pi}{200} \times \alpha \]

Altgrad in Radiant:  
\[ f(\alpha) = \frac{\pi}{180} \times \alpha \]

Artilleriepromille in Radiant:  
\[ f(\alpha) = \frac{\pi}{3200} \times \alpha \]
G.2.2 Conversion decimal-sexagesimal

Nomenclature:

GIVEN:
\( \alpha \) : angle to convert

WANTED:
min : minute
sek : second

Formula:

\[
\begin{align*}
\text{min} &= \text{Int}(\text{Frac}(\alpha) \times 60) \\
\text{sek} &= \text{Frac}(\text{Frac}(\alpha) \times 60) \times 60 \\
f(\alpha) &= \text{Int}(\alpha) + \frac{\text{min}}{10^2} + \frac{\text{sek}}{10^3}
\end{align*}
\]

Example:

\( \alpha = 3.562100 \)

\( \text{min} = 33.000000 \)

\( \text{sek} = 43.560000 \)

\( f(\alpha) = 3.334356 \)
G.2.3 Conversion sexagesimal-decimal

**Nomenclature:**

**GIVEN:**

\( \alpha \): angle to convert

**Formula:**

\[
f(\alpha) = \text{Int}(\alpha) + \frac{\text{Int}((\text{Frac}(\alpha) \times 10^2) \times 60 + \text{Frac}(\text{Frac}(\alpha) \times 10^2)) \times 10^2}{3600}
\]

**Example:**

\( \alpha = 3.334356 \)

\( f(\alpha) = 3.562100 \)

---

G.3 CONVERSION OF DISTANCE

**Nomenclature:**

**WANTED:**

\( \text{US}_{\text{foot}} \): American foot

\( \text{Inter}_{\text{foot}} \): International foot

**Formula:**

\( \text{US}_{\text{foot}} = \frac{3.937}{12} = 0.32808 \text{ m} \)

\( \text{Inter}_{\text{foot}} = \frac{9.144}{30} = 0.3048 \text{ m} \)
G.4 PHYSICAL CONVERSION

Nomenclature:

\begin{align*}
\text{mmHg} & : \text{mm mercury column} \\
\text{mbar} & : \text{Millibar} \\
\theta_K & : \text{Temperature in Kelvin} \\
\theta_F & : \text{Temperature in degree Fahrenheit} \\
\theta_C & : \text{Temperature in degree centigrade}
\end{align*}

Formula:

Pressure:

\[ 1 \text{ mm Hg} = 1.33322 \text{ mbar} = 1 / 760 \text{ atm} \]

Temperature:

\[ f(t_k) = t_k - 273.15 \]

\[ ^\circ \text{Fahrenheit in } ^\circ \text{C} \quad f(t_k) = 5/9*(t_F - 32) \]

G.5 CALCULATION OF AVERAGE:

G.5.1 Generally

Nomenclature:

\begin{align*}
\text{GIVEN} : \\
L_i & : \text{Measurement} \\
p_i & : \text{Significance of the measurement } L_i
\end{align*}
WANTED:

$L_{\text{mean}}$ : Average of all measurements
$v_i$ : Rectification of measurement $L_i$
$m_L$ : middle error of any measurement
$m_{\text{mean}}$ : middle error of average

Formula:

\[
L_{\text{mean}} = \frac{\sum p_i \cdot L_i}{\sum p_i}
\]

\[
v_i = L_{\text{mean}} - L_i
\]

\[
m_L = \sqrt{\frac{\sum (p_i \cdot v_i^2)}{n - 1}}
\]

\[
m_{\text{mean}} = \frac{m_i}{\sqrt{\sum p_i}}
\]

Authority: Lecture of surveying at the IBB Muttenz

G.5.2 Calculation of average for directions

Nomenclature:

GIVEN:

$R_i$ : i. direction element in array
$R_1$ : 1. direction element in array

WANTED:

$R_{\text{mean}}$ : arithmetical average direction
$m_R$ : middle error of any direction
$m_{\text{mean}}$ : middle error of average
Formula:
if \( \text{Abs} \left( R_i - R_j \right) > \pi \) then

begin
  if \( (R_i - R_j) > 0 \)
  then \( R_i := R_i + 2\pi \)
  else \( R_i := R_i - 2\pi \)
end

Calculation of \( R_{\text{mean}}, m_R, m_{\text{mean}} \) see formula calculation of average: generally

if \( R_{\text{mean}} < 0 \)
then \( R_{\text{mean}} := R_{\text{mean}} + 2\pi \)
else \( R_{\text{mean}} := R_{\text{mean}} \mod 2\pi \)

Authority: Specification circle-orientation for UD2 Report No GA 08/91

G.5.3 Calculation of median for directions

Nomenclature:
GIVEN:
\( n \) : Number of directions
\( R_i \) : \( i \). direction element in array
\( R_1 \) : 1. direction element in array
\( R_{n/2} \) : middle direction element in array
WANTED:
\( R_{\text{MED}} \) : as median averaged direction
Formula:
if \((n \mod 2) = 0\) then {even number of points}
begin
  if \(\text{Abs}(R_{n/2} - R_{n/2+1}) > \pi\)
  then \(R_{\text{med}} := \frac{R_{n/2} + R_{n/2+1} + 2\pi}{2}\) mod \(2\pi\)
  else \(R_{\text{med}} := \frac{R_{n/2} + R_{n/2+1}}{2}\)
end
else \(R_{\text{med}} := R_{n/2}\)

Authority: Specification circle orientation of UD2 Report No GA 08/91

G.6 CALCULATION OF COORDINATE

Nomenclature in general:

\(P_0\) (\(E_0, N_0, H_0\)) : Position and the coordinates
\(P_i\) (\(E_i, N_i, H_i\)) : Target point and the coordinates
\(\Delta E\) : Coordinate-difference in west-east direction
\(\Delta N\) : Coordinate-difference in north-south direction
G.6.1 Calculation of azimuth and distance result from coordinate

Formula:

\[ \Delta E = E_i - E_0 \]
\[ \Delta N = N_i - N_0 \]
\[ dh_{P_0P} = \sqrt{\Delta E^2 + \Delta N^2} \]
Case distinction:

if ((ΔN = 0) AND (ΔE = 0)) then error information
if (ΔN = 0)
    then if (ΔE > 0)
        then $Z_{p_i-p_i} := \pi/2$
        else $Z_{p_i-p_i} := 3/2 \pi$
    else begin
        $Z = \text{atan} \left( \frac{\Delta E}{\Delta N} \right)$
        if (ΔN < 0)
            then $Z_{p_i-p_i} := Z_{p_i-p_i} + \pi$
        else if (ΔE < 0)
            then $Z_{p_i-p_i} := Z_{p_i-p_i} + 2\pi$
    end

G.6.2 Calculation of coordinate result from azimuth and distance:

Formula:

$E_i = E_0 + \Delta E$
$N_i = N_0 + \Delta N$

$\Delta E = dh_{g_i-p_i} \cdot \sin (Z_{p_i-p_i})$

$\Delta N = dh_{g_i-p_i} \cdot \cos (Z_{p_i-p_i})$
G.6.3 Conversion polar - rectangular

see calculation of coordinate result from azimuth and distance

G.6.4 Conversion rectangular - polar

see calculation of azimuth and distance result from coordinate

G.6.5 Calculation of zenith angle and slope distance as a result from coordinate

Nomenclature:

GIVEN:
P_0 (E_0,N_0,H_0) : position and the coordinate
P_1 (E_i,N_i,H_i) : target point and the coordinate
i : Instrument height
s : Reflector height


Formula:
\[ \Delta E = E_i - E_0 \]
\[ \Delta N = N_i - N_0 \]
\[ \Delta h_{P_i - P_i} = \sqrt{\Delta E^2 + \Delta N^2} \]
\[ \Delta H_{P_i - P_i} = H_i - H_o \]

if \( ((\Delta H_{P_i - P_i} - i + s) = 0) \) then \[ V_{n-\eta} = \frac{\pi}{2} \]

else begin

\[ V_{n-\eta} = \operatorname{atan}\left(-\frac{\Delta h_{n-\eta}}{\Delta H_{P_i - P_i} - i + s}\right) \]

if \( (V_{n-\eta} < 0) \) then \[ V_{n-\eta} = V_{n-\eta} + \pi \]

end

\[ d_{s_{n-\eta}} = \Delta h_{n-\eta} \times \sin (V_{n-\eta}) \]
G.7 TRANSFORMATION OF COORDINATE

G.7.1 of mathematical coordinate systems

PICTURE :

Nomenclature :

GIVEN :
- \( P_0 \) : centre point known in both system.
- \( \varphi \) : Angle of rotation between the two coordinate systems. This is the angle (clockwise is positive) between the old and the new system.
- \( \Delta y, \Delta x \) : Coordinate of centre point \( P_0 \) of both coordinate systems.
- \( y_i, x_i \) : Coordinate in the old system (e.g. local system)

WANTED :
- \( y'_i, x'_i \) : Coordinate in the new system (e.g. country coordinate system)
Formula:
\[ \Delta y = y'_0 - y_0 \]
\[ \Delta x = x'_0 - x_0 \]
\[ y'_i = \Delta y + y_i \cdot \cos(\varphi) - x_i \cdot \sin(\varphi) \]
\[ x'_i = \Delta x + y_i \cdot \sin(\varphi) + x_i \cdot \cos(\varphi) \]

G.7.2 of geodetical coordinate systems

Picture:

Nomenclature:

**GIVEN:**
- \( P_0 \) : in both system known common points
- \( \varphi \) : Rotation angle between the two coordinate systems. This is the angle (clockwise is negative) between the old and the new system.
- \( \Delta y, \Delta x \) : Coordinate difference of the common point \( P_0 \) of both coordinate systems.
- \( E_i, N_i \) : Coordinates in the old system (i.e. country coordinate system)
WANTED:
y_i, x_i : Coordinate in the new system (i.e. mathematics system)

Formula:
\Delta y = y_0 - E_0
\Delta x = x_0 - N_0
y_i = \Delta y + N_i \cdot \sin(\varphi) - E_i \cdot \cos(\varphi)
x_i = \Delta x + N_i \cdot \cos(\varphi) + E_i \cdot \sin(\varphi)

G.8 CALCULATION OF TRIANGLE

G.8.1 Case SWS

Nomenclature:
GIVEN:
b, c : given triangle sides
\alpha : given angle
WANTED:

\( a \) : wanted triangle sides
\( \beta, \gamma \) : wanted angles

**Formula:**

\[ a = \sqrt{b^2 + c^2 - 2bc \cos(\alpha)} \]

\[ \beta = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right) \]

\[ \gamma = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right) \]

**G.8.2 Case SSS**

**Nomenclature:**

**GIVEN:**

\( a, b, c \) : given triangle sides

**WANTED:**

\( \alpha, \beta, \gamma \) : wanted angles

**Formula:**

Remark: if the sum of the two shorter sides are smaller than the longer side, there is no solution.

\[ \alpha = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right) \]

\[ \beta = \arcsin\left(\frac{b \times \sin(\alpha)}{a}\right) \]

\[ \gamma = \pi - (\alpha + \beta) \]
Nomenclature:
GIVEN:
\(a, c\) : given triangle sides
\(\gamma\) : given angle

WANTED:
\(b_1, b_2\) : wanted triangle sides
\(\alpha, \alpha_2, \beta, \beta_2\) : wanted angles

Formula:
Formula in general:
\(\beta = \pi - (\alpha + \gamma)\)
if \(((\gamma = 0) \text{ OR } (\gamma = \pi))\)
then if \((\gamma = 0)\)
then \(b = a + c\)
else \(b = \text{Abs} (a - c)\)
else \(b = \frac{c \times \sin \beta}{\sin \gamma}\)

First solution:
\(\alpha_1 = \text{asin} \left(\frac{a \times \sin \gamma}{c}\right)\)

Calculation of \(\beta_1\) and \(b_1\) with \(\alpha_1\) and \(\gamma\), see above formula in general
Case -Distinction :
if (c > a) then 2. solution
begin
   \[ \gamma_2 = \pi - \gamma \]
   \[ \alpha_2 = \alpha_1 \]
   Calculation of \( \beta_2 \) and \( b_2 \) with \( \alpha_2 \) and \( \gamma_2 \) see above formula in general
end
if (c = a) then only one solution, see above
if (c < a) then
   if (a * \sin \gamma > c) then no solution
   if (a * \sin \gamma = c) then only one solution, see above
   if (a * \sin \gamma < c) then 2. solution
begin
   \[ \alpha_2 = \pi - \alpha_1 \]
   Calculation of \( \beta_2 \) and \( b_2 \) with \( \alpha_2 \) and \( \gamma_2 \) see above formula in general
end

G.8.4 Case WWS or SWW

Nomenclature:
GIVEN:
\[ a \] : given triangle side
\[ \alpha, \beta \] : given angle
WANTED:
\( b,c \) : wanted triangle sides
\( \gamma \) : wanted angles

Formula:
if \( ((\alpha + \beta) \geq \pi) \text{ OR } (\sin \alpha = 0) \)
then no solution
else begin
\[
\begin{align*}
\gamma &= \pi - (\alpha + \beta) \\
b &= \frac{a \cdot \sin \beta}{\sin \alpha} \\
c &= \frac{a \cdot \sin (\alpha + \beta)}{\sin \alpha}
\end{align*}
\]
end

Nomenclature:
GIVEN:
\( a \) : given triangle side
\( \beta, \gamma \) : given angle

WANTED:
\( b,c \) : wanted triangle sides
\( \alpha \) : wanted angle
Formula:
if \((\sin(\beta + \gamma) = 0)\) then no solution
else begin
\[
\alpha = \pi - (\beta + \gamma)
\]
\[
b = \frac{a \cdot \sin \beta}{\sin(\beta + \gamma)}
\]
\[
c = \frac{a \cdot \sin \gamma}{\sin(\beta + \gamma)}
\]
end

G.9 CALCULATION OF CIRCLE

G.9.1 Radius and center result from 3 point

Picture:
Nomenclature:

GIVEN:
P1, P2, P3 : Coordinate from point P1 - P3

WANTED:
a, b, c : Chords
r : Radius

Formula and proceeding of calculation:

1. Calculation of chord a, b and c (see calculation of coordinate, azimuth and calculation of distance result from coordinate).

2. Calculation of angle $\alpha$, $\beta$, and $\gamma$ (see calculation of triangle case SSS)

   \[ \delta = \frac{\alpha - \beta - \gamma}{2} \]

3. 

   \[ r = \frac{a}{2 \cos (\delta)} \]

4. Calculation of azimuth from point 1 to point 3 (see calculation of coordinate, azimuth and distance result from coordinate)

5. Important: The points P1 to P3 are marked clockwise.

   \[ Z_{P1-centre} = Z_{P1-P3} + \delta \]

6. Calculation of centre coordinates with $Z_{P1-centre}$ and r (see calculation of coordinate result from azimuth and distance)

7. Control of centre coordinates: Calculation of distance centre - P2
if \( D_{centre-P2} \neq r \pm 0.001 \) then
{ The calculated centre co-ordinates are wrong
  Calculation of new centre co-ordinates }
begin
\[ Z_{P1-centre} = Z_{P1-P3} - \delta \]
Repetition of point 6.
end

G.10  CALCULATION OF INTERSECTION

G.10.1  Intersection line - line without parallel displacement

Picture :

Nomenclature :
GIVEN :
P1 - P4  : Coordinate from P1 - P4
WANTED:
$Z_{P1,P2}, Z_{P1,P3}, Z_{P3,P4}, Z_{P3,P1}$ : Azimuth
$\beta, \gamma$ : Assistance angle

Formula and proceeding of calculation:
1. Calculation of azimuth $Z_{P1,P2}, Z_{P1,P3}$ and $Z_{P3,P4}$ (see calculation of coordinate, azimuth and distance result from coordinate)

2. $Z_{P3,P1} = Z_{P1,P3} + \pi$

3. Calculation of distance P1 to P3 (see calculation of coordinate, azimuth and distance result from coordinate)

4. $\beta = Z_{P1,P3} - Z_{P1,P2}$ $\gamma = Z_{P3,P4} - Z_{P3,P1}$
   - if $\beta < 0$ then $\beta = \beta + \pi$
   - if $\gamma < 0$ then $\gamma = \gamma + \pi$

5. Calculation of distance P1 to P3 (see calculation of coordinate, azimuth and distance result from coordinate)

6. Calculation of distance P1 to S (see calculation of triangle, case WSW)

7. Calculation of intersection coordinate with the distance from P1 to S and azimuth $Z_{P1,P2}$ (see calculation of coordinate result from azimuth and distance)
G.10.2 Intersection line - line with parallel displacement

Remark: The parallel displacement on the left side of the line is negative, on the right side positive.

Formula and proceeding of calculation:

1. Calculation of azimuth (see calculation of intersection without parallel displacement, point 1. and point 2.)

2. Calculation of azimuth to the assistance point P1´ and P3´

\[ Z_{P1,P1'} = Z_{P1,P2} + \pi \]
\[ Z_{P3,P3'} = Z_{P3,P4} + \pi \]

3. Calculation of coordinate of assistance point P1´ and P3´ with azimuth \( Z_{P1,P1'} \) and \( Z_{P3,P3'} \) and parallel displacement pv1 and pv2. (see calculation of coordinate result from azimuth and distance) Important: Consider the sign of the parallel displacement.
4. After substitute $P_1 = P_1'$ and $P_3 = P_3'$, calculation of intersection $S$ (see calculation of intersection without parallel displacement: Points 3 - 7).

G.10.3 Intersection line - circle

**Picture:**

![Picture showing the intersection of a line and a circle with points labeled $P_1$, $P_2$, $SP_1$, $SP_2$, and the circle's center labeled as Zentrum.]

**Formula and proceeding of calculation:**

1. Calculation of azimuth $Z_{P_1P_2}$ (see calculation of coordinate, azimuth and distance result from coordinate).

2. Calculation of azimuth $Z_{P_1-Centre}$ and the distance $P_1$-centre (see calculation of coordinate, azimuth and distance result from coordinate).

3. $\alpha = Z_{P_1-Centre} - Z_{P_1P_2}$

4. Calculation of distance $P_1$-$SP_1$ and $P_1$-$SP_2$ with $\alpha$, distance $P_1$-centre and radius $r$. (see calculation of triangle, case SSW or WSS)
5. Calculation of intersection coordinate with the distances \( P1-SP1 \) resp. \( P1-SP2 \) and the azimuth \( Z_{P1-P2} \). (see calculation of coordinate result from azimuth and distance).

G.10.4 Intersection circle - circle

Picture:

\[ \text{Picture:} \]

Nomenclature:
- \( Z_1 \): centre of 1.circle
- \( Z_2 \): centre of 2.circle
- \( r_1 \): radius of 1.circle
- \( r_2 \): radius of 2.circle
- \( SP_1, SP_2 \): Intersection of both circles

Formula and proceeding of calculation:

1. Calculation of azimuth \( Z_{Z_1-Z_2} \) and the distance \( Z_1-Z_2 \) (see calculation of coordinate, azimuth and distance resulting from coordinate)
2. Calculation of angle $\alpha$ with $r_1, r_2$ and the distance $Z_1-Z_2$ (see calculation of triangle, case SSS)
   
   if ($\alpha = 0$)
   
   then only one intersection
   
   $Z_{SP_1} = Z_{Z_1-Z_2}$
   
   else begin
   
   $Z_{SP_1} = Z_{Z_1-Z_2} - \alpha$
   
   $Z_{SP_2} = Z_{Z_1-Z_2} + \alpha$
   
   end

4. Calculation of intersection coordinate with $Z_{P_1-SP_1}$ resp. $Z_{P_1-SP_2}$ and $r_1$ (see calculation of coordinate result from azimuth and distance).

G.11 CALCULATION OF DISTANCE

G.11.1 Distance point - point

see calculation of coordinate, azimuth and distance result from coordinate.
G.11.2 Distance point - line

Picture:

Nomenclature:
GIVEN:
P1 - P3 : Coordinate from point P1 - P3

WANTED:
x, y : Distances

Formula and proceeding calculation:
1. Calculation of azimuth $Z_{P1,P3}$ and the distance $P1-P3$ (see calculation of coordinate, azimuth and distance result from coordinate).

2. Calculation of azimuth $Z_{P1,P2}$

   $\alpha = Z_{P1,P2} - Z_{P1,P3}$

   $x = a \times \sin (\alpha)$

   $y = a \times \cos (\alpha)$
G.11.3 Distance point - circle

Nomenclature:
GIVEN:
Z and P1 : Coordinate of centre of the circle and of the point P1
r : radius

WANTED:
x : distance

Formula and proceeding calculation:
1. Calculation of distance $dh_{Z,P1}$ (see calculation of coordinate, azimuth and distance result from coordinate)
2. $x = dh_{Z,P1} - r$

G.11.4 Distance point - Clothoid

see calculation of the base point of foot of a perpendicular observation, point on Clothoid
G.12 CALCULATION OF THE BASE POINT OF PLUMB LINE

G.12.1 Point on line

Picture:

Nomenclature:
GIVEN:
P1 - P3 : Coordinate from point P1 - P3

WANTED:
x, y : Distances
FP : Base point of plumb line

Formula and proceeding calculation:
1. Calculation of distance y. (see calculation of distance, distance point - line)

2. Calculation of the Base point of plumb line FP. (see Point with distance on line)
G.12.2 Point on circle

Picture:

Nomenclature:

GIVEN:
Z and P1 : Coordinate of the centre of the circle and the point P1
r : radius

WANTED:
x : distance

Formula and proceeding calculation:

1. Calculation of azimuth $Z_{Z,P1}$ (see calculation of coordinate, azimuth and distance result from coordinate)

2. Calculation of the Base point of plumb line with $Z_{Z,P1}$ and the Radius r. (see calculation of coordinate result from azimuth and distance)
G.12.3  Point on Clothoid

Nomenclature:
GIVEN:
P1 : point to be plumbed out Point  
A : Clothoid parameter  
BA, BE : coordinates of the beginning (BA) and the end (BE) of the arc  
P_i : Base point of plumb line calculated at the i. iteration-step  

WANTED:
r : radius in the unitary clothoids  
l : length of the arc on the unity clothoid  
a : distance from P1 to the unity clothoid  
P_{i+1} : wanted base point of plumb line at the next iteration-step  

Formula and proceeding calculation:
This iteration algorithm is only applicable for solutions in the range  
0 < τ < $\frac{\pi}{2}$  

in the area of the clothoid.

First: Point P1 is transformed from the country-coordinate system to the mathematics system of the unity clothoid (\( A=1 \)).

Second: The first approximation for the start-value of \( l_n \) is the X-coordinate of the point P1.

\[
\text{if } (x_{P1} < \sqrt{\pi}) \text{ then } l_n = x_{P1} \quad \text{else} \quad l_n = \sqrt{\pi}
\]

Iteration-algorithm for the calculation of the Base point of plumb line:

1. Calculation of coordinates of point \( P_i \) with \( \tau = \frac{l_n^2}{2} \)

(see clothoid - Calculation, Calculated Coordinate)

2. Calculation of azimuth \( Z_{P_iP_1} \) and the distance \( a \). (see calculation of coordinate, azimuth and distance see result from coordinate).

Attention: The coordinates are located in the mathematics system, that means the substitution \( E=X \) and \( N=Y \) have to be used first, yet before the function is used.

\[
gw_l = 0.0001 \quad \{ \text{limit for arc-length} \}
\]

\[
\text{if } (l_n < gw_l) \text{ then } l_n = gw_l
\]

\[
\Delta l_n = \frac{\tan \left( \frac{a \cdot l_n \cdot \sin(\varphi)}{1 + a \cdot l_n \cdot \cos(\varphi)} \right)}{l_n}
\]

\[
l_{n+1} = l_n + \Delta l_n
\]

\[
\text{if } (l_{n+1} > \sqrt{\pi}) \text{ then } l_{n+1} = \sqrt{\pi}
\]

4. Termination-condition:
if \( \Delta l_n < 10^{-8} \) OR \( n > 5 \)  
then terminate iteration  
else next iteration - step with \( l_n = l_{n+1} \) (see point 1-3)  

5. Error treatment :  
\[
gw_{\text{terminate}} = 10^{-8} \quad \{ \text{limit for termination of iteration - algorithm} \}  
\]
if \( \Delta l_n > gw_{\text{terminate}} \) OR \( n > 5 \) then no solution found  

6. The Base point of plumb line in the clothoid-calculation, which is found in this proceeding has to be retransformed into the country coordinate system. (see calculation of clothoid - transformation)  

G.13 CALCULATE POINT WITH DISTANCE ON LINE  

G.13.1 Point with distance on line  

Nomenclature :  
GIVEN :  
P1,P2 : point on line  
x : distance of point to be calculated (P3) to point P1  

WANTED :  
P3 : point to be calculated  

Formula and proceeding calculation :  
1. Calculation of azimuth \( Z_{P1P2} \) (see calculation of coordinate, azimuth and distance, see result from coordinate).  
2. Calculation of point P3 with \( Z_{P1P2} \) and x (see calculation of coordinate with azimuth and distance).
G.13.2 Calculate point on arc of circle with distance

G.13.2.1 Beginning and end point of arc are given:

Picture:

Nomenclature:

GIVEN:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
<td>Start of arc</td>
</tr>
<tr>
<td>BE</td>
<td>End of arc</td>
</tr>
<tr>
<td>r</td>
<td>Radius</td>
</tr>
</tbody>
</table>

WANTED:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>new point</td>
</tr>
<tr>
<td>γ</td>
<td>displacement angle</td>
</tr>
<tr>
<td>b</td>
<td>arc-length (clockwise positive)</td>
</tr>
</tbody>
</table>

Formula and proceeding calculation:

1. Calculation of azimuth $Z_{BA, BE}$ and distance $dh_{BA, BE}$ (see calculation of coordinate, azimuth and distance result from coordinate).
2. if \( b < +r\pi \) then \( b = b - 2r\pi \)
   if \( b < -r\pi \) then \( b = b + 2r\pi \)
   \[ \gamma = \frac{b}{r} \]
   \[ s_1 = \text{Abs}(2r \times \sin\left(\frac{\gamma}{2}\right)) \]
   \[ \alpha = \cos\left(\frac{s_1}{2r}\right) \]
   \[ \beta = \cos\left(\frac{s_2}{2r}\right) \]
   if \( b < 0 \) then \( \alpha = 0 - \alpha \)
   \[ Z_{BA\text{-}NP} = Z_{BA\text{-}BE} - (\alpha - \beta) \]

3. Calculation of coordinate from the new point with \( Z_{BA\text{-}NP} \) and \( s_1 \) (see calculation of coordinate result from azimuth and distance).

**G.13.2.2 Center of Circle is given**

**Formula and proceeding calculation**:

1. Calculation of azimuth \( Z_{Z\text{-}P1} \) (see calculation of coordinate, azimuth and distance result from coordinate).
   \[ \gamma = \frac{b}{r} \]

2. \[ Z_{Z\text{-}p_{NP}} = Z_{Z\text{-}P1} + \gamma \]
3. Calculation of the new point NP with $Z_{2, \pi}$ and radius $r$. (see calculation of coordinate result from azimuth and distance).

**G.14 CALCULATION OF CLOTHOID**

Picture:

![Clothoid Diagram]

**Nomenclature:**
- $R$: Radius
- $L$: arc length
- $\tau$: Tangent-angle
- $A$: Clothoid parameter
  - If Clothoid rotates to the left, then $A$ is negative;
  - if to the right then $A$ is positive.
Formula in general:

\[
\begin{align*}
R &= \frac{A^2}{L} = \frac{L}{2\tau} = \frac{A}{\sqrt{2}\tau} \\
L &= \frac{A^2}{R} = 2\tau R = A\sqrt{2}\tau \\
\tau &= \frac{L}{2R} = \frac{L^2}{2A^2} = \frac{A^2}{2R^2} \\
A &= \sqrt{L*R} = \frac{L}{\sqrt{2}\tau} = R\sqrt{2}\tau
\end{align*}
\]

G.14.1 Calculated Coordinate

Nomenclature:

GIVEN:

\(\tau\) : Tangent -angle

WANTED:

\(x_i, y_i\) : Coordinate in the unity-clothoid system

Formula:

The formulas are valid only for the calculation of coordinates in the unity-clothoid system (\(A=1\)).

\[
\begin{align*}
x_i &= \sqrt{2}\tau * \sum_{n=1}^{\infty} ((-1)^{n+1} * \frac{\tau^{(2n-2)}}{(4n-3)*(2n-2)!}) \\
y_i &= \sqrt{2}\tau * \sum_{n=1}^{\infty} ((-1)^{n+1} * \frac{\tau^{(2n-1)}}{(4n-1)*(2n-1)!})
\end{align*}
\]
G.15 TRANSFORMATION

Nomenclature:

GIVEN:

\begin{align*}
A & : \text{clothoid parameter} \\
L & : \text{arc length} \\
\mathbf{P}_0 & : \text{Zero-point coordinate of the clothoid system} \\
\mathbf{P}_1 & : \text{given point on the clothoid} \\
\mathbf{P}_i & : \text{Coordinate of the point, which has to be transformed, in the old system.}
\end{align*}

WANTED:

\begin{align*}
\mathbf{P}_i' & : \text{Coordinate of the point, which has been transformed, in the new system.}
\end{align*}

Picture:
Formula and proceeding calculation:
1. Calculation of angle $\tau$ (see formula in general)

2. Calculation of coordinate of point $P_t$ in the unity clothoid system (see calculation of coordinate)

3. Calculation of angle $\beta$:

$$\beta = \text{atan} \left( \frac{y}{x} \right)$$

4. Calculation of rotation-angle
   if $(A > 0)$
   then $\alpha = (Z_{P_t-P_0} - \beta)$
   else $\alpha = (Z_{P_t-P_0} + \beta)$
   if (Transformation direction: Klothoidensystem into Country system)
   then $\alpha = 2\pi - \alpha$

5. Calculated transformation with $P_0$ as common point, $\alpha$ as rotation angle and point $P_t$.
   (see coordinate-transformation [geodetic Systems])
G.16 PLANIMETRY

G.16.1 Planimetry result from coordinate (Gauss)

Picture:

Nomenclature:
GIVEN:
- \( n \): Number of corner-points
- \( Y \): Y-coordinate
- \( X \): X-coordinate

WANTED:
- \( F \): Plane

Formula:
\[
2F = \sum_{i=1}^{n} Y_i * (X_{i-1} - X_{i+1})
\]
G.16.2 Planimetry result from measurement (triangle)

Picture:

\[ \alpha = \frac{dh_{P_0-P_1} + dh_{P_0-P_2}}{2} \]

Remark: The points \( P_1 \) and \( P_2 \) are defined clockwise. The result of exchanging the horizontal directions is a negative plane.

Nomenclature:

GIVEN:
- \( H_{z_{P_0-x}} \): horizontal direction from point \( P_0 \) to point \( x \)
- \( dh_{P_0-x} \): horizontal distance from point \( P_0 \) to point \( x \)

WANTED:
- \( F \): Triangle plane

Formula:
\[ \alpha = \frac{H_{z_{P_0-P_1}} - H_{z_{P_0-P_2}}}{dh_{P_0-P_1} \ast dh_{P_0-P_2} \ast \sin (\alpha)} \]
G.16.3 Segment Plane

Picture:

Nomenclature:
GIVEN:
s : Tendon length
r  : Radius

WANTED:
F  : Segment plane

Formula:
\[ \gamma = \frac{s}{r} \]
\[ F = \frac{r^2 (\gamma - \sin(\gamma))}{2} \]
G.17 EXCENTER OBSERVATION RE-CENTERED TO THE CENTER

G.17.1 Distance Measurement to the Mark

Picture:

Nomenclature:
GIVEN:
\( H_{z, \text{Ex-ZP}}, V_{x, \text{Ex-ZP}}, d_{s, \text{Ex-ZP}} \): Measure - element on the excenter
\( e \): Horizontal-distance centre-excenter

WANTED:
\( H_{z, \text{Z-ZP}}, V_{z, \text{Z-ZP}}, d_{s, \text{Z-ZP}} \): on the centre re-centre measure - element

Formula and proceeding calculation:
1. Calculation of horizontal distance \( d_{h, \text{Ex-ZP}} \) (see geometry reduction of the measured distance).

2. \( \alpha = H_{z, \text{Ex-ZP}} - H_{z, \text{Ex-Z}} \)
3. Calculation of $\beta$ and the horizontal distance $dh_{Z,ZP}$ with $e$, $dh_{Ex,ZP}$ and $\alpha$
   (see calculation of triangle, case SWS)

4. Calculation of the re-centred horizontal direction

   if $(Hz_{Ex,ZP} \geq 0) \text{ AND } (Hz_{Ex,ZP} \leq \pi)$ then $Hz_{Ex,ZP} = Hz_{Ex,ZP} + 2\pi$
   if $(Hz_{Ex,Z} \geq 0) \text{ AND } (Hz_{Ex,Z} \leq \pi)$ then $Hz_{Ex,Z} = Hz_{Ex,Z} + 2\pi$
   if $(Hz_{Ex,ZP} > Hz_{Ex,Z})$
      then $Hz_{Z,ZP} = 2\pi - \beta$
   else $Hz_{Z,ZP} = \beta$

5. Calculation of the re-centred vertical direction

   $\Delta V = \tan \left( \frac{\Delta H_{Z,Ex}}{dh_{Z,ZP}} \right)$
   if $(V_{Ex,ZP} < \pi)$ { test if the telescope is in I. position }
      then $V_{Z,ZP} = V_{Ex,ZP} + \Delta V$
   else $V_{Z,ZP} = V_{Ex,ZP} - \Delta V$

6. Calculation of the re-centred slope distance

   $ds_{Z,ZP} = dh_{Z,ZP} \times \sin (V_{Z,ZP})$

G.17.2 Distance is not measured to the mark

Remark: This assumes, that the coordinate of centre and mark are available.

**Formula and proceeding calculation:**

1. Calculation of $dh_{Z,ZP}$ (see calculation of coordinate, azimuth and Distance result from Coordinate).
2. Calculation of angle \( \alpha \)

\[
\alpha = H_{Ex-ZP} - H_{Ex-Z}
\]

if \((\alpha < 0)\) then \(\alpha = \alpha + 2\pi\)

if \((\alpha > \pi)\)

then begin

\[
\alpha = \alpha - \pi
\]

\(\beta\) of the 2. solution is OK (see calculation of triangle)

else \(\beta\) of the 1. solution is OK (see calculation of triangle)

3. Calculation of \(\beta\) with \(dh_{Z-ZP}\), \(e\) and \(\alpha\) (see calculation of triangle, case SSW)

4. Calculation of the re-centred horizontal direction

see above (Distance measured to the mark) point 4.

5. Calculation of the re-centred vertical direction

see above (Distance measured to the mark) point 5

---

G.18 TRANSVERSE - AND LONGITUDINAL DISPLACEMENT IN THE MARK

Picture:
Nomenclature:
GIVEN:
L : Longitudinal displacement
Q : Transverse displacement
Hz_gem : measured horizontal direction
dh : reduced horizontal distance

WANTED:
dhkorr : corrected horizontal distance
Hz_korr : corrected horizontal direction

Formula:
Correction in consequence of longitudinal displacement:
\[ dh_{korr} = dh + L \]

Correction in consequence of transverse displacement:
\[ dh_{korr} = \sqrt{dh^2 + Q^2} \]
\[ Hz_{korr} = Hz_{gem} + \arctan \left( \frac{Q}{dh} \right) \]
G.19 CALCULATION OF LIMB ORIENTATION

Nomenclature:

GIVEN:
- \( P_0 (E_0, N_0, H_0) \) : Position with the coordinate
- \( P_i (E_i, N_i, H_i) \) : Mark with the coordinate
- \( H_{zi} \) : Horizontal direction
- \( n \) : Number of marks
- \( T \) : Test size of \( L1 \)
- \( h \) : Auxiliary for analysis of observation

WANTED:
- \( Z_i \) : Azimuth from position \( P_0 \) to the mark \( P_i \)
- \( O_i \) : Orientation of limb
- \( O_{\text{mean}} \) : Orientation unknown quantity as arithmetic average
- \( O_{\text{med}} \) : Orientation unknown quantity as median
- \( V_{L1} \) : Improvement at the direction \( H_{zi} \) from \( L1 \)
- \( M_r \) : Exactness of one single direction
- \( M_{or} \) : Exactness of the orientation unknown quantity \( O_{\text{mean}} \)
- \( Q \) : Limit for \( M_{or} \) (a priori exactness)

Formula and proceeding calculation:

The formulas are only valid for the units meter and gon

1. Calculation of azimuth \( Z_i \) from position \( P_0 (E_0, N_0, H_0) \) to the mark \( P_i (E_i, N_i, H_i) \)

(see calculation of azimuth and distance result from coordinate)
2. \( O_1 = (Z_i - Hz_i + 2\pi) \mod 2\pi \)

3. Calculation of average \( O_{\text{mean}} \) result from \( O_i \)
   (see calculation of average for directions)

4. Calculation of average \( O_{\text{med}} \) result from \( O_i \)
   (see calculation of median for directions)

5. \( V_{\text{Li}} = Z_i - (O_{\text{med}} + Hz_i) \mod 2\pi \)

6. Calculation of the exactness of one single direction \( M_{\text{r}} \) and the exactness of the orientation unknown quantity \( M_{\text{or}} \) (see Calculation of average in generally)

7. if \( M_{\text{or}} \leq Q \) then result is accepted, no analysis of the observation has to be made

8. if \( n < 3 \) then no analysis of the observation has to be made

9.
   \[
   h = O_{\text{mean}} \\
   \text{if } \text{abs}(O_{\text{med}} - O_{\text{mean}}) > 2\pi \text{ then} \\
   \text{begin} \\
   \text{if } (O_{\text{med}} - O_{\text{mean}}) > 0 \text{ then } h = O_{\text{mean}} + 2\pi \\
   \text{if } (O_{\text{med}} - O_{\text{mean}}) < 0 \text{ then } h = O_{\text{mean}} - 2\pi \\
   \text{end} \\
   \]

10. \( T = 3 \times (O_{\text{med}} - h) \)
    if \( T < 0.0003 \text{ gon} \) then no analysis of the observation has to be made
10. Analysis of the observation: if \( T < \left| V_{L_i} \right| \) then \( H_z \) is wrong

G.20 HIDDEN POINT

Picture:
Geometry of the Staff:

Nomenclature:
GIVEN:

- $H^p_{p_{i-1}}$, $d_{p_{i-1}}$, $V_{p_{i-1}}$ : Measurement at the station Po
- $H^p_{p_{i-2}}$, $d_{p_{i-2}}$, $V_{p_{i-2}}$

- $a$ : Distance of both reflectors
- $l$ : Distance of the hidden point from the reflector first measured (also possible that it is negative)

WANTED:

- $H^p_{p_{i-3}}$, $d_{p_{i-3}}$, $V_{p_{i-3}}$ : calculated measured values to the hidden point

Formula and proceeding calculation:
1. Calculation of the horizontal distance \(dh_{P0,P1}\), \(dh_{P0,P2}\) and the height differences \(\Delta H_{P0-1}, \Delta H_{P0-2}\)
   (see geometry reduction of the measured distance)

2. \(\alpha_1 = Hz_{P2} - Hz_{P0}\)

3. Calculation of the angle \(\beta\) with \(dh_{P0-1}\), \(\alpha_1\) and \(dh_{P0-2}\). (see calculation of triangle, case SWS)
   if \((l < 0)\) then \(\beta = \pi - \beta\)

4. \(\gamma = \arcsin \left( \frac{\Delta H_{P0-2} - \Delta H_{P0-1}}{a} \right)\)
   \(\Delta H_{1,3} = l \cdot \sin(\gamma)\)
   \(dh_{1,3} = \text{Abs}(l) \cdot \cos(\gamma)\)

5. Calculation of the distance \(dh_{P0,3}\) and the angle \(\alpha_2\) with \(dh_{P0-1}\), \(\beta\) and \(dh_{1,3}\) (see calculation of triangle, case SWS).
   if \((l < 0)\) then \(\alpha_2 = 0 - \alpha_2\)

6. Calculation of the vertical direction \(V_{P0-3}\)

   \(\Delta V = \arctan \left( \frac{\Delta H_{1,3}}{dh_{P0-3}} \right)\)
   if \((V_{P0-1} < \pi)\) \{ test if telescope in I.position \}
   then \(V_{P0-3} = V_{P0-1} - \Delta V\)
   else \(V_{P0-3} = V_{P0-1} + \Delta V\)
7. Calculation of the slope distance $ds_{P_0-3}$

$$ds_{P_0-3} = \text{Abs} \left( \frac{dh_{P_0-3}}{\sin (V_{P_0-3})} \right)$$

8. Calculation of the horizontal direction $Hz_{P0-3}$

if $(Hz_{P_0-1} \geq 0)$ AND $(Hz_{P_0-1} \leq \pi)$ then $Hz_{P_0-1} = Hz_{P_0-1} + 2\pi$

if $(Hz_{P_0-2} \geq 0)$ AND $(Hz_{P_0-2} \leq \pi)$ then $Hz_{P_0-2} = Hz_{P_0-2} + 2\pi$

if $(Hz_{P_0-2} > Hz_{P_0-1})$

then $Hz_{P_0-3} = (Hz_{P_0-1} + \alpha_2) \mod 2\pi$

else $Hz_{P_0-3} = (Hz_{P_0-1} - \alpha_2) \mod 2\pi$
Appendix H — CSV_SysCall Constants

The following is a list of all system functions and system events which are defined for creating user configurations and can be used in GeoBASIC applications, too.

A system function can be executed directly with the GeoBASIC routine 
`CSV_SysCall(SystemFunction)`.

The same routine is also used to generate a system event. But the functionality is not the same. In the case of a system event, the system function (or dialog, menu, macro, application) will be executed which is connected to the event by the current configuration. If no system function (or dialog, menu, macro, application) is connected to the event, then the routine 
`CSV_SysCall(SystemEvent)` returns RC_IVPARAM. CSV_SFNC_* is the prefix of a system function, while CSV_EFNC_* denote an system event.

<table>
<thead>
<tr>
<th>System Functions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_SFNC_BeepAlarm</td>
<td>Beep alarm: Alarm beep</td>
</tr>
<tr>
<td>CSV_SFNC_BeepLong</td>
<td>Beep long: Long beep</td>
</tr>
<tr>
<td>CSV_SFNC_BeepNormal</td>
<td>Beep normal: Normal beep</td>
</tr>
<tr>
<td>CSV_SFNC_BeepShort</td>
<td>Beep short: Short Beep</td>
</tr>
<tr>
<td>CSV_SFNC_CallFreeCodingDlg</td>
<td>Coding with codelists: Standard coding application. Priorities:</td>
</tr>
<tr>
<td></td>
<td>1. GeoBASIC code-function</td>
</tr>
<tr>
<td></td>
<td>2. OSW free coding</td>
</tr>
<tr>
<td>CSV_SFNC_CallFreeCodingDlg</td>
<td>3. Standard coding</td>
</tr>
<tr>
<td>CSV_SFNC_ChangeFace</td>
<td>(lc=II) Change face: The dialog shows either the difference on non motorized theodolites or the turning info.</td>
</tr>
<tr>
<td>CSV_SFNC_CheckMemCard</td>
<td>Check PC-Card: Check PC-Card dialog</td>
</tr>
<tr>
<td>CSV_SFNC_CheckOrientation</td>
<td>Check orientation application (dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_ClearDist</td>
<td>Release V-angle: Release of a frozen V-Angle and clears the measured distance. This function is only useful in a measurement dialog.</td>
</tr>
<tr>
<td>CSV_SFNC_ClearOffset</td>
<td>Set all Offs. to 0.0: Set the target point offset to 0.0</td>
</tr>
<tr>
<td>CSV_SFNC_ConvertFile</td>
<td>Data conversion: Converts coordinate files (Dialogs)</td>
</tr>
<tr>
<td>CSV_SFNC_CreateNewJobDlg</td>
<td>Create new job: Creates a new job (Dialog)</td>
</tr>
<tr>
<td>System Functions</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CSV_SFNC_CurrentSetPpmDlg</td>
<td>Shows the ppm-setting dialog, depending on the predefined (Full or reduced)</td>
</tr>
<tr>
<td>CSV_SFNC_DataView</td>
<td>Data view and edit: Standard data view and edit application (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_DefineMeasDlg</td>
<td>Displ. mask definition: Defines the content of the standard measurement dialog (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_DefineRecMaskDlg</td>
<td>REC-Mask Definition: Defines the content of the recorded measurement data blocks (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_DefSearchAreaDlg</td>
<td>Defines the searching area (dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_DeleteLastGsiBlock</td>
<td>Delete last block: Deletes the last recorded GSI block (measurement data or code data)</td>
</tr>
<tr>
<td>CSV_SFNC_EdmTest</td>
<td>EDM Test signal/freq: Test EDM signal and frequency (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_ExecAutoexecItem</td>
<td>Autoexec-Handler: This handler is the standard function for the Autoexec event. It executes the pre-selected function/macro.</td>
</tr>
<tr>
<td>CSV_SFNC_FetchLastPoint</td>
<td>Get last rec. PointId: Sets the actual point number to the last recorded one.</td>
</tr>
<tr>
<td>CSV_SFNC_FormatMemCard</td>
<td>Format PC-Card: Formats after a query the PC-Card and shows the Card-Info dialog</td>
</tr>
<tr>
<td>CSV_SFNC_ImportStationCoordDlg</td>
<td>Import station coord: Imports the station-coordinates of a given pointId. If the coordinates are available, then there is no dialog visible.</td>
</tr>
<tr>
<td>CSV_SFNC_ImportStationCoordViewDlg</td>
<td>View/import stat.coor: Imports station-coordinates of the actual pointId. It shows the coordinates before importing (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_InstrCorrections</td>
<td>Instr. Calibration: Starts the instrument calibration application (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_Libelle</td>
<td>Electronic Level: Shows the standard electronic level dialog</td>
</tr>
<tr>
<td>CSV_SFNC_Light</td>
<td>Instrument lights: Shows the light configuration dialog</td>
</tr>
<tr>
<td>CSV_SFNC_LoadApplDlg</td>
<td>Shows the “Loading Application” dialog</td>
</tr>
<tr>
<td>CSV_SFNC_LoadConfigFile</td>
<td>Load configuration: Loads a new instrument configuration. After Loading, the instrument will restart. (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_LoadParamFile</td>
<td>Load system parameter: Load the instrument parameter file (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_LoadSysLangDlg</td>
<td>Shows the “Loading system language” dialog</td>
</tr>
<tr>
<td>System Functions</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSV_SFNC_ManageCodelistDlg</td>
<td>Codelist management: Codelist management dialog</td>
</tr>
<tr>
<td>CSV_SFNC_ManageDataJobDlg</td>
<td>Data job management: Data job management dialog</td>
</tr>
<tr>
<td>CSV_SFNC_ManageMeasJobDlg</td>
<td>Meas job management: Measurement job management dialog</td>
</tr>
<tr>
<td>CSV_SFNC_ManCoordDlg</td>
<td>Enter coordinate set: Manual entering of coordinates</td>
</tr>
<tr>
<td>CSV_SFNC_ManStationCoordDlg</td>
<td>Enter station coord: Manual entering of station coordinates.</td>
</tr>
<tr>
<td>CSV_SFNC_MeasAttributeCodeDlg</td>
<td>Attributes: Edit attributes of the current code</td>
</tr>
<tr>
<td>CSV_SFNC_MeasPrgm</td>
<td>EDM program selection: EDM measurement program and reflector selection (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_MeasureDist</td>
<td>(DIST) Distance measure: Measure a distance and show the distance measurement dialog during the distance measurement. This function is only useful in a measurement dialog.</td>
</tr>
<tr>
<td>CSV_SFNC_MeasureDistAndRec</td>
<td>(ALL) Measure and Rec: Forces a distance measurement and records the target point data. This function is only useful in a measurement dialog.</td>
</tr>
<tr>
<td>CSV_SFNC_PositCompassDlg</td>
<td>RCS Ori.with compass: RCS orientation with a compass (Dialog). This function is only at RCS mode available.</td>
</tr>
<tr>
<td>CSV_SFNC_PositHzVDlg</td>
<td>RCS Positioning Hz/V: Positioning with manual entering of the Hz- and V-angle (Dialog). This function is only at RCS mode available.</td>
</tr>
<tr>
<td>CSV_SFNC_PositJoystickDlg</td>
<td>RCS Move by joystick: RCS moving using the joystick functionality (Dialog). This function is only at RCS mode available.</td>
</tr>
<tr>
<td>CSV_SFNC_PositLastPoint</td>
<td>Turn to last rec.Pt: Turns the instrument to the last recorded position. This function needs a motorised instrument.</td>
</tr>
<tr>
<td>CSV_SFNC_QuickSet</td>
<td>Quick station setup: Quick orientation to a backside point and setting of the station coordinates. (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_RecordStationData</td>
<td>(REC) Station data: Recording of the station data.</td>
</tr>
</tbody>
</table>
### System Functions

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_SFNC_RecordTargetPoint</td>
<td>Target Point: Standard Recording of Target point according the selected REC-Mask and increments the running point number.</td>
</tr>
<tr>
<td>CSV_SFNC_SaveParamFile</td>
<td>Save param. to PC-Card: Saves the actual parameter setting to a PC-Card file (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_SetAccessoriesDlg</td>
<td>Accessories: Accessories definition dialog</td>
</tr>
<tr>
<td>CSV_SFNC_SetDefaultSearchRange</td>
<td>Sets the searching area back to default.</td>
</tr>
<tr>
<td>CSV_SFNC_SetFullPpmDlg</td>
<td>PPM (atm. + geom.): Shows the full ppm correction dialog. It allows a more detailed configuration than PPM atmospheric dialog</td>
</tr>
<tr>
<td>CSV_SFNC_SetGSIDefaultParam</td>
<td>Set GSI default param: Resets the GSI communication parameters to the default values</td>
</tr>
<tr>
<td>CSV_SFNC_SetHz</td>
<td>Set Hz to any angle: Sets the Hz-circle orientation to any angle (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_SetHz0</td>
<td>Set Hz to 0.0: Sets the Hz circle orientation to 0.0</td>
</tr>
<tr>
<td>CSV_SFNC_SetManDist</td>
<td>Horiz.distance entry: Manual entry of a horizontal distance (Dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_SetMeasDlgMask1</td>
<td>Set Display Mask 1: Set display #1 for the measurement dialog</td>
</tr>
<tr>
<td>CSV_SFNC_SetMeasDlgMask2</td>
<td>Set Display Mask 2: Set display #2 for the measurement dialog</td>
</tr>
<tr>
<td>CSV_SFNC_SetMeasDlgMask3</td>
<td>Set Display Mask 3: Set display #3 for the measurement dialog</td>
</tr>
<tr>
<td>CSV_SFNC_SetNextMeasDlgMask</td>
<td>Show next displ. mask: Show the next defined display mask (of the standard measurement dialog)</td>
</tr>
<tr>
<td>CSV_SFNC_SetOnlineDlg</td>
<td>GeoCOM On-Line mode: Leaves the manual control and switches to the GeoCOM controlled mode. Before switching, it shows a confirmation message</td>
</tr>
<tr>
<td>CSV_SFNC_SetRcsDlg</td>
<td>RCS (Remote) On/Off: Enables or disables the remote control</td>
</tr>
<tr>
<td>CSV_SFNC_SetRecMask1</td>
<td>Set Rec-Mask 1: Set recmask #1 for target point recording</td>
</tr>
<tr>
<td>CSV_SFNC_SetRecMask2</td>
<td>Set Rec-Mask 2: Set recmask #2 for target point recording</td>
</tr>
<tr>
<td>CSV_SFNC_SetRecMask3</td>
<td>Set Rec-Mask 3: Set recmask #3 for target point recording</td>
</tr>
<tr>
<td>System Functions</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSV_SFNC_SetRecMask4</td>
<td>Set Rec-Mask 4: Set recmask #4 for target point recording</td>
</tr>
<tr>
<td>CSV_SFNC_SetRecMask5</td>
<td>Set Rec-Mask 5: Set recmask #5 for target point recording</td>
</tr>
<tr>
<td>CSV_SFNC_SetSimplePpmDlg</td>
<td>PPM Atmospheric: Shows the simple ppm dialog</td>
</tr>
<tr>
<td>CSV_SFNC_StdCodeDlg</td>
<td>Coding (standard): Standard coding application (without a codelist)</td>
</tr>
<tr>
<td>CSV_SFNC_SWInfoDlg</td>
<td>Instr. Information: Shows the instrument information dialog</td>
</tr>
<tr>
<td>CSV_SFNC_SwitchDirectOff</td>
<td>(OFF) Instr.power Off: Switches the instrument off without a message-box</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleATR</td>
<td>Enable / Disable ATR: Enables or disables the ATR. This function is only at ATR instruments available</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleDisplayLight</td>
<td>Displ.illumin. On/Off: Toggles the display illumination</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleEglIllum</td>
<td>EGL Illumin. On/Off: Toggles the EGL illumination. This function is only at instruments with a build-in EGL available</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleLaserPlummet</td>
<td>Laser plummet On/Off: Toggles the laser plummet. This function is only available on instruments with a build-in laser plummet</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleLock</td>
<td>Enable / Disable LOCK: Enables or disables the Locking. This function is only at ATR instruments available</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleLockInt</td>
<td>Interrupt/Resume LOCK: Suspend or resume the locking. This function is only at ATR instruments available</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleMeasPrgFast-RapidTrk</td>
<td>Toggles the measurement program (Fast / Rapid-Tracking)</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleMeasPrgRefRL</td>
<td>Toggles the measurement mode (Prism / Reflectorless)</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleMeasPrgStdTracking</td>
<td>Toggles the measurement program (Standard / Tracking)</td>
</tr>
<tr>
<td>CSV_SFNC_TogglePointNumbering</td>
<td>Indiv/running Pointld: Toggles the target point numbering Individual &lt;-&gt; Running</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleQuickCode</td>
<td>Quick coding On/Off: Enables / disables the Quick-Coding</td>
</tr>
</tbody>
</table>
**System Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_SFNC_ToggleRedLaser</td>
<td>EDM red laser On/Off: Toggles the red laser on or off. This function is only available on instruments with a build-in EDM with a visible laser light available</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleReticuleIllum</td>
<td>Retic.illumin. On/Off: Toggles the reticule illumination</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleSearchArea</td>
<td>Enables / disables the searching working area</td>
</tr>
<tr>
<td>CSV_SFNC_ToggleVAngleMode</td>
<td>Toggles the V-Angle mode (Free / Fixed)</td>
</tr>
</tbody>
</table>

**System Events**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV_EFNC_ButtonCode</td>
<td>Button CODE: Generates the same event like pressing the CODE button. Usually connected to the &quot;Coding&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonFnc</td>
<td>Button FNC: Generates the same event like pressing the Fnc button. Usually connected to the FNC menu.</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonLevel</td>
<td>Button LEVEL: Generates the same event like pressing the level button. Usually connected to the &quot;Electronic Level&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonLight</td>
<td>Button LIGHT: Generates the same event like pressing the light button. Usually connected to &quot;Instrument Lights&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonPowerOff</td>
<td>Button Power OFF: Generates the same event like pressing the power off button. Usually not connected</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonPowerOn</td>
<td>Button Power ON: Generates the same event like after switching on the instrument. Usually connected to the &quot;Standard autoexec handler&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonProg</td>
<td>Button PROG: Generates the same event like pressing the PROG button. Usually connected to user defined application program menu.</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonShiftCode</td>
<td>Button Shift CODE: Generates the same event like pressing the shift CODE button. Usually not connected</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonShiftFnc</td>
<td>Button Shift FNC: Generates the same event like pressing the shift Fnc button. Usually not connected</td>
</tr>
<tr>
<td>System Events</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSV_EFNC_ButtonShiftProg</td>
<td>Button Shift PROG: Generates the same event like pressing the Shift PROG button. Usually not connected</td>
</tr>
<tr>
<td>CSV_EFNC_CompensatorSetting</td>
<td>Compensator/Level: Global compensator setting event. Usually connected to an user-defined compensator setting dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_DataView</td>
<td>Data View and Edit: Global Data View and Edit event. Usually connected to the &quot;Data view and edit&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_GeoComSetup</td>
<td>GeoCOM Setup: GeoCOM parameter setup event. Usually connected to a user-defined GeoCOM parameter dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_GsiSetup</td>
<td>GSI Setup: GSI parameter setup event. Usually connected to a user-defined GSI parameter dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_MeasDistance</td>
<td>(DIST) Distance meas: Global distance measurement event. Usually connected to the Distance measurement function</td>
</tr>
<tr>
<td>CSV_EFNC_MeasDistanceRecord</td>
<td>(ALL) Measure and Rec: Global measure and record a distance event. Usually connected to the &quot;Measure and Record&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_MeasRecord</td>
<td>Measure and Record: Global measurement and recording event. Usually connected to a user-defined measurement dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_RcsSetup</td>
<td>RCS Setup: RCS parameter setup event. Usually connected to a user-defined RCS parameter dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_RecordStation</td>
<td>(REC) Station Data: Global station data recording event. Usually connected to the &quot;Recording of station data&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_RecordTarget</td>
<td>(REC) Target point: Global Recording event. Usually connected to the &quot;Recording of target point&quot; function</td>
</tr>
<tr>
<td>CSV_EFNC_RootFunction</td>
<td>System MAIN: Generates the same event like after the power-on. Usually connected to the Main-Men</td>
</tr>
<tr>
<td>CSV_EFNC_SelectReflector</td>
<td>Reflector selection: Global reflector selection event. Usually connected to an user-defined reflector selection dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_SetStation</td>
<td>Station setup: Global Station data setting event. Usually connected to an user-defined station setting dialog.</td>
</tr>
<tr>
<td>System Events</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSV_EFNC_Setup</td>
<td>Job and Work settings: Global Setup event (setup station, jobs...). Usually connected to an user-defined setup dialog.</td>
</tr>
<tr>
<td>CSV_EFNC_TargetData</td>
<td>Target Data Settings: Global target data setting event. Usually connected to an user-defined target data dialog</td>
</tr>
<tr>
<td>CSV_EFNC_USER1</td>
<td>User Event 1: Usually not connected</td>
</tr>
<tr>
<td>CSV_EFNC_USER2</td>
<td>User Event 2: Usually not connected</td>
</tr>
<tr>
<td>CSV_EFNC_USER3</td>
<td>User Event 3: Usually not connected</td>
</tr>
</tbody>
</table>
Appendix I — Callable C-Application Functions

The entry points - listed below - are those, which can be called directly from a GeoBASIC application, if and only if the application is loaded already.
These entry points relate heavily on the application release, as the C-applications themselves relate to the firmware release. Please cross check this in future for new releases of either the C-applications and/or the TPS1100 firmware.

I.1 ENTRY POINTS

The table below lists all valid application names and entry points. An example of a valid call could be the following. Capitalisation is relevant!

    CSV_LibCall ( "FreeSt_Ori_Res", "ORIMain", sCaptionShort )

<table>
<thead>
<tr>
<th>Application - Version</th>
<th>Application Name – Entry Point Name(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area - 2.0</td>
<td>&quot;Area” – “AreaApplMain”</td>
</tr>
<tr>
<td>Auto Record - 2.0</td>
<td>“AutoRecord” – “AutoRecApplMain”</td>
</tr>
<tr>
<td>COGO - 2.0</td>
<td>“COGO” – “CogoApplMain”</td>
</tr>
<tr>
<td>DTM Stakeout - 2.0</td>
<td>“DTM-Stakeout” – “TinStakeMain”</td>
</tr>
<tr>
<td>Face Scan - 2.0</td>
<td>“FaceScanning” – “FscanApplMain”</td>
</tr>
<tr>
<td>Free Station Orientation Resection - 2.0</td>
<td>“FreeSt_Ori_Res” – “FRSMain” “ORIMain” “RESMain”</td>
</tr>
<tr>
<td>File Editor - 2.0</td>
<td>“FileEditor” – “FileEdMain”</td>
</tr>
<tr>
<td>File View - 2.0</td>
<td>“Text-View” – “FileViewApplMain”</td>
</tr>
<tr>
<td>Hidden Point - 2.0</td>
<td>“HiddenPoint” – “HiddenPtrApplMain”</td>
</tr>
<tr>
<td>Local Resection - 2.0</td>
<td>“LocalRes” – “LocalResMain”</td>
</tr>
<tr>
<td>Reference Line - 2.0</td>
<td>“REPL” – “RefLineApplMain”</td>
</tr>
<tr>
<td>Remote Height - 2.0</td>
<td>“RemoteHeight” – “RemHtMain”</td>
</tr>
<tr>
<td>Road Plus - 2.02</td>
<td>“RoadPlus” – “RoadPlusApplMain”</td>
</tr>
<tr>
<td>Sets of Angles - 2.00</td>
<td>“Sets” – “SetsApplMain”</td>
</tr>
<tr>
<td>Stakeout - 2.02</td>
<td>“Stakeout” – “StakeOutApplMain”</td>
</tr>
<tr>
<td>Application - Version</td>
<td>Entry Point Application Name – Entry Point Name(s)</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Traverse – 2.0</td>
<td>“Traverse” – “TravMain”</td>
</tr>
<tr>
<td>Tie Distance – 2.0</td>
<td>“TieDistance” – “TieDistApplMain”</td>
</tr>
</tbody>
</table>
### J.1 TYPES

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date_Time_Type</td>
<td>Date and time information.</td>
</tr>
<tr>
<td>Date_Type</td>
<td>Date information.</td>
</tr>
<tr>
<td>FileId</td>
<td>File identifier.</td>
</tr>
<tr>
<td>FileName</td>
<td>String * 100 for path and file name.</td>
</tr>
<tr>
<td>GM_4Transform_Param_Type</td>
<td>Transformation parameters.</td>
</tr>
<tr>
<td>GM_Circle_Type</td>
<td>Definition of a circle.</td>
</tr>
<tr>
<td>GM_Excenter_Elems_Type</td>
<td>Elements of the eccentric observation.</td>
</tr>
<tr>
<td>GM_Line_Type</td>
<td>Definition of a line.</td>
</tr>
<tr>
<td>GM_Mean_StdDev_Type</td>
<td>Average, middle error of average, and middle error of any observation.</td>
</tr>
<tr>
<td>GM_Measurements_Type</td>
<td>Structure used for measurement (polar coordinates).</td>
</tr>
<tr>
<td>GM_Point_Type</td>
<td>Definition of a point.</td>
</tr>
<tr>
<td>GM_QXX_Matrix_Type</td>
<td>Coefficients of the cofactor matrix of the unknown.</td>
</tr>
<tr>
<td>GM_Triangle_Accuracy_Type</td>
<td>Accuracy of angle and side of the triangle.</td>
</tr>
<tr>
<td>GM_Triangle_Values_Type</td>
<td>Sides and angles of a triangle.</td>
</tr>
<tr>
<td>GSI_Point_Coord_Type</td>
<td>Point coordinate data.</td>
</tr>
<tr>
<td>GSI_Rec_Id_List</td>
<td>Array of integers (indicating WI–identifications).</td>
</tr>
<tr>
<td>GSI_Widlg_Entry_Type</td>
<td>Dialog entry information.</td>
</tr>
<tr>
<td>ListArray</td>
<td>Array of String * 30 type</td>
</tr>
<tr>
<td>SLine</td>
<td>Display line</td>
</tr>
<tr>
<td>String10</td>
<td>String * 10 type</td>
</tr>
</tbody>
</table>
### J.1 Types

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date_Time_Type</td>
<td>Date and time information.</td>
</tr>
<tr>
<td>Date_Type</td>
<td>Date information.</td>
</tr>
<tr>
<td>FileId</td>
<td>File identifier</td>
</tr>
<tr>
<td>FileName</td>
<td>String * 100 for path and file name</td>
</tr>
<tr>
<td>GM_4Transform_Param_Type</td>
<td>Transformation parameters.</td>
</tr>
<tr>
<td>GM_Circle_Type</td>
<td>Definition of a circle.</td>
</tr>
<tr>
<td>GM_Excenter_Elems_Type</td>
<td>Elements of the eccentric observation.</td>
</tr>
<tr>
<td>GM_Line_Type</td>
<td>Definition of a line.</td>
</tr>
<tr>
<td>GM_Mean_StdDev_Type</td>
<td>Average, middle error of average, and middle error of any observation.</td>
</tr>
<tr>
<td>GM_Measurements_Type</td>
<td>Structure used for measurement (polar coordinates).</td>
</tr>
<tr>
<td>GM_Point_Type</td>
<td>Definition of a point.</td>
</tr>
<tr>
<td>GM_QXX_Matrix_Type</td>
<td>Coefficients of the cofactor matrix of the unknown.</td>
</tr>
<tr>
<td>GM_Triangle_Accuracy_Type</td>
<td>Accuracy of angle and side of the triangle.</td>
</tr>
<tr>
<td>GM_Triangle_Values_Type</td>
<td>Sides and angles of a triangle.</td>
</tr>
<tr>
<td>GSI_Point_Coord_Type</td>
<td>Point coordinate data.</td>
</tr>
<tr>
<td>GSI_Rec_Id_List</td>
<td>Array of integers (indicating WI–identifications).</td>
</tr>
<tr>
<td>GSI_WiDlg_Entry_Type</td>
<td>Dialog entry information.</td>
</tr>
<tr>
<td>ListArray</td>
<td>Array of String * 30 type</td>
</tr>
<tr>
<td>SLine</td>
<td>Display line</td>
</tr>
<tr>
<td>String10</td>
<td>String * 10 type</td>
</tr>
<tr>
<td>String18</td>
<td>String * 18 type</td>
</tr>
<tr>
<td>Type Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>String20</td>
<td>String * 20 type</td>
</tr>
<tr>
<td>String255</td>
<td>String * 255 type</td>
</tr>
<tr>
<td>String30</td>
<td>String * 30 type</td>
</tr>
<tr>
<td>Time_Type</td>
<td>Time information</td>
</tr>
<tr>
<td>TMC_ANG_SWITCH_Type</td>
<td>Angle measurement switches</td>
</tr>
<tr>
<td>TMC_Angle_Type</td>
<td>Data structure for measuring angles.</td>
</tr>
<tr>
<td>TMC_ATMOS_TEMPERATURE_Type</td>
<td>Corrections for distance measurement: to define PPM values of atmosphere</td>
</tr>
<tr>
<td>TMC_Coordinate_Type</td>
<td>Data structure for the coordinates (tracking and fixed coordinates).</td>
</tr>
<tr>
<td>TMC_DIST_SWITCHES_Type</td>
<td>Distance measurement switches</td>
</tr>
<tr>
<td>TMC_Distance_Type</td>
<td>Data structure for the distance measurement.</td>
</tr>
<tr>
<td>TMC_GEOM_PROJECTION_Type</td>
<td>Corrections for distance measurement: to define PPM values of projection</td>
</tr>
<tr>
<td>TMC_GEOM_REDUCTION_Type</td>
<td>Corrections for distance measurement: to define PPM values of reduction to the reference</td>
</tr>
<tr>
<td>TMC_HZ_V_Ang_Type</td>
<td>Horizontal and vertical angle.</td>
</tr>
<tr>
<td>TMC_Incline_Type</td>
<td>Data structure for the inclination measurement.</td>
</tr>
<tr>
<td>TMC_OFFSET_DIST_Type</td>
<td>Target offset</td>
</tr>
<tr>
<td>TMC_PPM_CORR_Type</td>
<td>Correction for distance measurement.</td>
</tr>
<tr>
<td>TMC_REFRACTION_Type</td>
<td>Refraction correction for distance measurement.</td>
</tr>
<tr>
<td>TMC_STATION_Type</td>
<td>Station coordinates.</td>
</tr>
<tr>
<td>TPS_Fam_Type</td>
<td>Information about the current hardware.</td>
</tr>
<tr>
<td>Wi_List</td>
<td>Array of GSI_WiDlg_Entry_Type.</td>
</tr>
</tbody>
</table>

### J.2 FUNCTIONS AND PROCEDURES

<table>
<thead>
<tr>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_FineAdjust</td>
<td>6-84</td>
</tr>
<tr>
<td>BAP_GetMeasPrg</td>
<td>6-90</td>
</tr>
<tr>
<td>BAP_MeasDistAngle</td>
<td>6-77</td>
</tr>
<tr>
<td>BAP_MeasRec</td>
<td>6-81</td>
</tr>
<tr>
<td>BAP_PosTelescope</td>
<td>6-91</td>
</tr>
</tbody>
</table>
BAP_SearchPrism................................................................. 6-85
BAP_SetAccessoriesDlg...................................................... 6-77
BAP_SetHz.................................................................... 6-93
BAP_SetManDist.............................................................. 6-86
BAP_SetMeasPrg............................................................... 6-88
BAP_SetPpm................................................................. 6-87
BAP_SetPrism................................................................. 6-88
ChDir......................................................................... 2-65
Close.......................................................................... 2-51
COM_ExecCmd................................................................. 2-78
COM_SetTimeOut............................................................. 2-77
CSV_ChangeFace............................................................... 6-196
CSV_CheckAltUserTask.................................................... 6-206
CSV_Delay.................................................................... 6-201
CSV_GetATRStatus.......................................................... 6-201
CSV_GetDateTime............................................................. 6-187
CSV_GetElapseSysTime.................................................... 6-192
CSV_GetGBIVersion......................................................... 6-191
CSV_GetInstrumentFamily............................................... 6-189
CSV_GetInstrumentName................................................. 6-188
CSV_GetInstrumentNo..................................................... 6-189
CSV_GetLaserPlummet..................................................... 6-205
CSV_GetLockStatus.......................................................... 6-198
CSV_GetLRStatus............................................................ 6-193
CSV_GetPrismType........................................................... 6-204
CSV_GetSWVersion.......................................................... 6-190
CSV_GetSysTime............................................................... 6-193
CSV_GetTargetType.......................................................... 6-203
CSV_GetTemperature......................................................... 6-188
CSV_Laserpointer.............................................................. 6-195
CSV_LibCall................................................................. 6-209
CSV_LibCallAvailable...................................................... 6-209
CSV_LockIn................................................................. 6-199
CSV_LockOut.................................................................................... 6-200
CSV_MakePositioning....................................................................... 6-195
CSV_ResetAltUserTask..................................................................... 6-206
CSV_SetATRStatus.......................................................................... 6-200
CSV_SetGuideLight......................................................................... 6-194
CSV_SetLaserPlummet..................................................................... 6-205
CSV_SetLockStatus.......................................................................... 6-197
CSV_SetPrismType.......................................................................... 6-204
CSV_SetTargetType.......................................................................... 6-202
CSV_SysCall.................................................................................... 6-207
CSV_SysCallAvailable..................................................................... 6-208
CurDir$ ............................................................................................ 2-64
Eof() (standard function)............................................................... 2-63
FileCopy........................................................................................... 2-72
Get – values.................................................................................... 2-55
GetDirectoryList............................................................................. 2-71
GetFileStat....................................................................................... 2-70
GetMemoryCardInfo......................................................................... 2-69
GSI_GetIndivNr............................................................................. 6-145
GSI_CheckTracking.......................................................................... 6-180
GSI_Coding....................................................................................... 6-148
GSI_CreateMDlg ............................................................................. 6-171
GSI_DefineMDlg............................................................................. 6-178
GSI_DefineRecMaskDlg.................................................................... 6-162
GSI_ExecuteAutoDist....................................................................... 6-150
GSI_GetDataPath............................................................................. 6-156
GSI_GetLineSysMDlg....................................................................... 6-169
GSI_GetMDlgNr............................................................................... 6-171
GSI_GetQCodeAvailable.................................................................. 6-149
GSI_GetRecMask............................................................................. 6-158
GSI_GetRecMaskNr.......................................................................... 6-162
GSI_GetRecOrder............................................................................. 6-152
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSI_GetRecPath</td>
<td>6-154</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>6-144</td>
</tr>
<tr>
<td>GSI_GetWiEntry</td>
<td>6-156</td>
</tr>
<tr>
<td>GSI_ImportCoordDlg</td>
<td>6-165</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>6-144</td>
</tr>
<tr>
<td>GSI_IsRunningNr</td>
<td>6-146</td>
</tr>
<tr>
<td>GSI_ManCoordDlg</td>
<td>6-162</td>
</tr>
<tr>
<td>GSI_IncPNumber</td>
<td>6-148</td>
</tr>
<tr>
<td>GSI_Measure</td>
<td>6-179</td>
</tr>
<tr>
<td>GSI_QuickSet</td>
<td>6-153</td>
</tr>
<tr>
<td>GSI_RecordRecMask</td>
<td>6-181</td>
</tr>
<tr>
<td>GSI_SelectCode</td>
<td>6-149</td>
</tr>
<tr>
<td>GSI_SetDataPath</td>
<td>6-155</td>
</tr>
<tr>
<td>GSI_SetIndivNr</td>
<td>6-146</td>
</tr>
<tr>
<td>GSI_SetIvPtNrStatus</td>
<td>6-147</td>
</tr>
<tr>
<td>GSI_SetLineMDlg</td>
<td>6-173</td>
</tr>
<tr>
<td>GSI_SetLineMDlgPar</td>
<td>6-175</td>
</tr>
<tr>
<td>GSI_SetLineMDlgText</td>
<td>6-174</td>
</tr>
<tr>
<td>GSI_SetLineSysMDlg</td>
<td>6-168</td>
</tr>
<tr>
<td>GSI_SetMDlgNr</td>
<td>6-170</td>
</tr>
<tr>
<td>GSI_SetQCodeMode</td>
<td>6-150</td>
</tr>
<tr>
<td>GSI_SetRecMask</td>
<td>6-159</td>
</tr>
<tr>
<td>GSI_SetRecMaskNr</td>
<td>6-161</td>
</tr>
<tr>
<td>GSI_SetRecOrder</td>
<td>6-152</td>
</tr>
<tr>
<td>GSI_SetRecPath</td>
<td>6-153</td>
</tr>
<tr>
<td>GSI_SetRunningNr</td>
<td>6-145</td>
</tr>
<tr>
<td>GSI_SetWiEntry</td>
<td>6-157</td>
</tr>
<tr>
<td>GSI_UpdateMDlg</td>
<td>6-177</td>
</tr>
<tr>
<td>GSI_UpdateMeasurement</td>
<td>6-178</td>
</tr>
<tr>
<td>Input</td>
<td>2-52</td>
</tr>
<tr>
<td>Kill</td>
<td>2-68</td>
</tr>
<tr>
<td>MkDir</td>
<td>2-66</td>
</tr>
<tr>
<td>MMI Data Types</td>
<td>6-9</td>
</tr>
<tr>
<td>MMI_AddButton</td>
<td>6-26</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>MMI_AddGBMenuButton</td>
<td>6-18</td>
</tr>
<tr>
<td>MMI_BeepAlarm</td>
<td>6-54</td>
</tr>
<tr>
<td>MMI_BeepLong</td>
<td>6-54</td>
</tr>
<tr>
<td>MMI_BeepNormal</td>
<td>6-54</td>
</tr>
<tr>
<td>MMI_CheckButton</td>
<td>6-23</td>
</tr>
<tr>
<td>MMI_CreateGBMenu</td>
<td>6-11</td>
</tr>
<tr>
<td>MMI_CreateGBMenuItem</td>
<td>6-13</td>
</tr>
<tr>
<td>MMI_CreateGBMenuItemStr</td>
<td>6-16</td>
</tr>
<tr>
<td>MMI_CreateGBMenuStr</td>
<td>6-14</td>
</tr>
<tr>
<td>MMI_CreateGraphDialog</td>
<td>6-21</td>
</tr>
<tr>
<td>MMI_CreateMenuItem</td>
<td>6-10</td>
</tr>
<tr>
<td>MMI_CreateTextDialog</td>
<td>6-19</td>
</tr>
<tr>
<td>MMI_DeleteButton</td>
<td>6-27</td>
</tr>
<tr>
<td>MMI_DeleteDialog</td>
<td>6-22</td>
</tr>
<tr>
<td>MMI_DeleteGBMenu</td>
<td>6-17</td>
</tr>
<tr>
<td>MMI_DrawBusyField</td>
<td>6-53</td>
</tr>
<tr>
<td>MMI_DrawCircle</td>
<td>6-51</td>
</tr>
<tr>
<td>MMI_DrawLine</td>
<td>6-48</td>
</tr>
<tr>
<td>MMI_DrawRect</td>
<td>6-49</td>
</tr>
<tr>
<td>MMI_DrawText</td>
<td>6-52</td>
</tr>
<tr>
<td>MMI_FormatVal</td>
<td>6-42</td>
</tr>
<tr>
<td>MMI_GetAngleRelation</td>
<td>6-60</td>
</tr>
<tr>
<td>MMI_GetAngleUnit</td>
<td>6-63</td>
</tr>
<tr>
<td>MMI_GetButton</td>
<td>6-24</td>
</tr>
<tr>
<td>MMI_GetCoordOrder</td>
<td>6-73</td>
</tr>
<tr>
<td>MMI_GetDateFormat</td>
<td>6-70</td>
</tr>
<tr>
<td>MMI_GetDistUnit</td>
<td>6-65</td>
</tr>
<tr>
<td>MMI_GetLangName</td>
<td>6-75</td>
</tr>
<tr>
<td>MMI_GetLanguage</td>
<td>6-74</td>
</tr>
<tr>
<td>MMI_GetPressUnit</td>
<td>6-67</td>
</tr>
<tr>
<td>MMI_GetTempUnit</td>
<td>6-68</td>
</tr>
<tr>
<td>MMI_GetTimeFormat</td>
<td>6-71</td>
</tr>
<tr>
<td>MMI_GetVAngleMode</td>
<td>6-61</td>
</tr>
</tbody>
</table>
GeoBASIC Reference Manual  Appendix J — List of Predefined Identifiers

MMI_GetVarBeepStatus...................................................................... 6-56
MMI_InputInt....................................................................................... 6-38
MMI_InputList..................................................................................... 6-40
MMI_InputStr...................................................................................... 6-33
MMI_InputVal..................................................................................... 6-35
MMI_PrintInt ....................................................................................... 6-32
MMI_PrintStr....................................................................................... 6-28
MMI_PrintTok ..................................................................................... 6-29
MMI_PrintVal...................................................................................... 6-30
MMI_SelectGBMenuItem.................................................................... 6-17
MMI_SetAngleRelation ....................................................................... 6-59
MMI_SetAngleUnit ............................................................................. 6-61
MMI_SetCoordOrder ........................................................................... 6-72
MMI_SetDateFormat ........................................................................... 6-69
MMI_SetDistUnit................................................................................. 6-63
MMI_SetLanguage............................................................................... 6-73
MMI_SetPressUnit............................................................................... 6-65
MMI_SetTempUnit.............................................................................. 6-67
MMI_SetTimeFormat .......................................................................... 6-70
MMI_SetVAngleMode ........................................................................ 6-60
MMI_StartVarBeep.............................................................................. 6-54
MMI_SwitchAFKey............................................................................. 6-57
MMI_SwitchIconsBeep........................................................................ 6-58
MMI_SwitchVarBeep........................................................................... 6-55
MMI_WriteMsg ................................................................................... 6-44
MMI_WriteMsgStr............................................................................... 6-46
Open..................................................................................................... 2-48
Print...................................................................................................... 2-54
Put – values......................................................................................... 2-58
Receive................................................................................................. 2-76
RenameDir ........................................................................................... 2-74
RenameFile .......................................................................................... 2-73
RmDir................................................................................................... 2-67
Seek ................................................................. 2-61
Send ................................................................. 2-75
Tell ................................................................. 2-60
TMC Data Structures .......................................... 6-96
TMC_DoMeasure ................................................. 6-99
TMC_Get/SetAngleFaceDef ................................. 6-117
TMC_Get/SetDistPpm ........................................... 6-119
TMC_Get/SetHeight .............................................. 6-119
TMC_Get/SetHzOffset .......................................... 6-118
TMC_Get/SetRefractiveCorr ..................... 6-120
TMC_Get/SetRefractiveMethod .......................... 6-120
TMC_Get/SetStation ........................................... 6-121
TMC_GetAngle .................................................. 6-107
TMC_GetAngle_WInc ........................................... 6-109
TMC_GetAngSwitch ............................................ 6-127
TMC_GetCoordinate ............................................ 6-104
TMC_GetDistSwitch ............................................ 6-123
TMC_GetFace1 .................................................... 6-126
TMC_GetInclineStatus ....................................... 6-128
TMC_GetInclineSwitch ...................................... 6-128
TMC_GetOffsetDist ............................................. 6-125
TMC_GetPolar ..................................................... 6-101
TMC_GetSimpleMea ............................................. 6-114
TMC_IfDistTapeMeasured ................................. 6-121
TMC_IfOffsetDistMeasured ............................... 6-125
TMC_QuickDist .................................................. 6-110
TMC_SetAngSwitch ............................................ 6-126
TMC_SetDistSwitch ............................................ 6-123
TMC_SetHandDist .............................................. 6-122
TMC_SetInclineSwitch ....................................... 6-127
TMC_SetOffsetDist ............................................. 6-124
<table>
<thead>
<tr>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String18</td>
<td>String * 18 type</td>
</tr>
<tr>
<td>String20</td>
<td>String * 20 type</td>
</tr>
<tr>
<td>String255</td>
<td>String * 255 type</td>
</tr>
<tr>
<td>String30</td>
<td>String * 30 type</td>
</tr>
<tr>
<td>Time_Type</td>
<td>Time information.</td>
</tr>
<tr>
<td>TMC_ANG_SWITCH_Type</td>
<td>Angle measurement switches</td>
</tr>
<tr>
<td>TMC_Angle_Type</td>
<td>Data structure for measuring angles.</td>
</tr>
<tr>
<td>TMC_ATMOS_TEMPERATURE_Type</td>
<td>Corrections for distance measurement: to define PPM values of atmosphere</td>
</tr>
<tr>
<td>TMC_Coordinate_Type</td>
<td>Data structure for the coordinates (tracking and fixed coordinates).</td>
</tr>
<tr>
<td>TMC_DIST_SWITCHES_Type</td>
<td>Distance measurement switches</td>
</tr>
<tr>
<td>TMC_Distance_Type</td>
<td>Data structure for the distance measurement.</td>
</tr>
<tr>
<td>TMC_GEOM_PROJECTION_Type</td>
<td>Corrections for distance measurement: to define PPM values of projection</td>
</tr>
<tr>
<td>TMC_GEOM_REDUCTION_Type</td>
<td>Corrections for distance measurement: to define PPM values of reduction to the reference</td>
</tr>
<tr>
<td>TMC_HZ_V_Ang_Type</td>
<td>Horizontal and vertical angle.</td>
</tr>
<tr>
<td>TMC_Incline_Type</td>
<td>Data structure for the inclination measurement.</td>
</tr>
<tr>
<td>TMC_OFFSET_DIST_Type</td>
<td>Target offset</td>
</tr>
<tr>
<td>TMC_PPM_CORR_Type</td>
<td>Correction for distance measurement.</td>
</tr>
<tr>
<td>TMC_REFRACTION_Type</td>
<td>Refraction correction for distance measurement.</td>
</tr>
<tr>
<td>TMC_STATION_Type</td>
<td>Station coordinates.</td>
</tr>
<tr>
<td>TPS_Fam_Type</td>
<td>Information about the current hardware.</td>
</tr>
<tr>
<td>Wi_List</td>
<td>Array of GSI_WiDlg_Entry_Type.</td>
</tr>
</tbody>
</table>

### J.2 FUNCTIONS AND PROCEDURES

<table>
<thead>
<tr>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAP_FineAdjust</td>
<td>6-85</td>
</tr>
<tr>
<td>BAP_GetMeasPrg</td>
<td>6-91</td>
</tr>
<tr>
<td>BAP_MeasDistAngle</td>
<td>6-78</td>
</tr>
<tr>
<td>BAP_MeasRec</td>
<td>6-82</td>
</tr>
<tr>
<td>Identifier</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>BAP_PosTelescope</td>
<td>6-92</td>
</tr>
<tr>
<td>BAP_SearchPrism</td>
<td>6-86</td>
</tr>
<tr>
<td>BAP_SetAccessoriesDlg</td>
<td>6-78</td>
</tr>
<tr>
<td>BAP_SetHz</td>
<td>6-94</td>
</tr>
<tr>
<td>BAP_SetManDist</td>
<td>6-87</td>
</tr>
<tr>
<td>BAP_SetMeasPrg</td>
<td>6-90</td>
</tr>
<tr>
<td>BAP_SetPpm</td>
<td>6-88</td>
</tr>
<tr>
<td>BAP_SetPrism</td>
<td>6-89</td>
</tr>
<tr>
<td>ChDir</td>
<td>2-65</td>
</tr>
<tr>
<td>Close</td>
<td>2-51</td>
</tr>
<tr>
<td>COM_ExecCmd</td>
<td>2-78</td>
</tr>
<tr>
<td>COM_SetTimeOut</td>
<td>2-77</td>
</tr>
<tr>
<td>CSV_ChangeFace</td>
<td>6-200</td>
</tr>
<tr>
<td>CSV_CheckAltUserTask</td>
<td>6-210</td>
</tr>
<tr>
<td>CSV_Delay</td>
<td>6-205</td>
</tr>
<tr>
<td>CSV_GetATRStatus</td>
<td>6-205</td>
</tr>
<tr>
<td>CSV_GetDateTime</td>
<td>6-191</td>
</tr>
<tr>
<td>CSV_GetElapseSysTime</td>
<td>6-196</td>
</tr>
<tr>
<td>CSV_GetGBIVersion</td>
<td>6-195</td>
</tr>
<tr>
<td>CSV_GetInstrumentFamily</td>
<td>6-193</td>
</tr>
<tr>
<td>CSV_GetInstrumentName</td>
<td>6-192</td>
</tr>
<tr>
<td>CSV_GetInstrumentNo</td>
<td>6-193</td>
</tr>
<tr>
<td>CSV_GetLaserPlummet</td>
<td>6-209</td>
</tr>
<tr>
<td>CSV_GetLockStatus</td>
<td>6-202</td>
</tr>
<tr>
<td>CSV_GetLRStatus</td>
<td>6-197</td>
</tr>
<tr>
<td>CSV_GetPrismType</td>
<td>6-208</td>
</tr>
<tr>
<td>CSV_GetSWVersion</td>
<td>6-194</td>
</tr>
<tr>
<td>CSV_GetSysTime</td>
<td>6-197</td>
</tr>
<tr>
<td>CSV_GetTargetType</td>
<td>6-207</td>
</tr>
<tr>
<td>CSV_GetTemperature</td>
<td>6-192</td>
</tr>
<tr>
<td>CSV_Laserpointer</td>
<td>6-199</td>
</tr>
<tr>
<td>CSV_LibCall</td>
<td>6-213</td>
</tr>
<tr>
<td>CSV_LibCallAvailable</td>
<td>6-213</td>
</tr>
<tr>
<td>Identifier</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------</td>
</tr>
<tr>
<td>CSV_LockIn</td>
<td>6-203</td>
</tr>
<tr>
<td>CSV_LockOut</td>
<td>6-204</td>
</tr>
<tr>
<td>CSV_MakePositioning</td>
<td>6-199</td>
</tr>
<tr>
<td>CSV_ResetAltUserTask</td>
<td>6-210</td>
</tr>
<tr>
<td>CSV_SetATRStatus</td>
<td>6-204</td>
</tr>
<tr>
<td>CSV_SetGuideLight</td>
<td>6-198</td>
</tr>
<tr>
<td>CSV_SetLaserPlummet</td>
<td>6-209</td>
</tr>
<tr>
<td>CSV_SetLockStatus</td>
<td>6-201</td>
</tr>
<tr>
<td>CSV_SetPrismType</td>
<td>6-208</td>
</tr>
<tr>
<td>CSV_SetTargetType</td>
<td>6-206</td>
</tr>
<tr>
<td>CSV_SysCall</td>
<td>6-211</td>
</tr>
<tr>
<td>CSV_SysCallAvailable</td>
<td>6-212</td>
</tr>
<tr>
<td>CurDir$</td>
<td>2-64</td>
</tr>
<tr>
<td>Eof() (standard function)</td>
<td>2-63</td>
</tr>
<tr>
<td>FileCopy</td>
<td>2-72</td>
</tr>
<tr>
<td>Get – values</td>
<td>2-55</td>
</tr>
<tr>
<td>GetDirectoryList</td>
<td>2-71</td>
</tr>
<tr>
<td>GetFileStat</td>
<td>2-70</td>
</tr>
<tr>
<td>GetMemoryCardInfo</td>
<td>2-69</td>
</tr>
<tr>
<td>GSI_GetIndivNr</td>
<td>6-149</td>
</tr>
<tr>
<td>GSI_CheckTracking</td>
<td>6-184</td>
</tr>
<tr>
<td>GSI_Coding</td>
<td>6-152</td>
</tr>
<tr>
<td>GSI_CreateMDlg</td>
<td>6-175</td>
</tr>
<tr>
<td>GSI_DefineMDlg</td>
<td>6-182</td>
</tr>
<tr>
<td>GSI_DefineRecMaskDlg</td>
<td>6-166</td>
</tr>
<tr>
<td>GSI_ExecuteAutoDist</td>
<td>6-154</td>
</tr>
<tr>
<td>GSI_GetDataPath</td>
<td>6-160</td>
</tr>
<tr>
<td>GSI_GetLineSysMDlg</td>
<td>6-173</td>
</tr>
<tr>
<td>GSI_GetMDlgNr</td>
<td>6-175</td>
</tr>
<tr>
<td>GSI_GetQCodeAvailable</td>
<td>6-153</td>
</tr>
<tr>
<td>GSI_GetRecMask</td>
<td>6-163</td>
</tr>
<tr>
<td>GSI_GetRecMaskNr</td>
<td>6-166</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>GSI_GetRecOrder</td>
<td>6-156</td>
</tr>
<tr>
<td>GSI_GetRecPath</td>
<td>6-158</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>6-148</td>
</tr>
<tr>
<td>GSI_GetWiEntry</td>
<td>6-161</td>
</tr>
<tr>
<td>GSI_GetWiEntryText</td>
<td>6-160</td>
</tr>
<tr>
<td>GSI_GetWiEntryText</td>
<td>6-160</td>
</tr>
<tr>
<td>GSI_GetRunningNr</td>
<td>6-150</td>
</tr>
<tr>
<td>GSI_ManCoordDlg</td>
<td>6-166</td>
</tr>
<tr>
<td>GSI_Measure</td>
<td>6-183</td>
</tr>
<tr>
<td>GSI_QuickSet</td>
<td>6-157</td>
</tr>
<tr>
<td>GSI_RecordRecMask</td>
<td>6-185</td>
</tr>
<tr>
<td>GSI_SelectCode</td>
<td>6-153</td>
</tr>
<tr>
<td>GSI_SetDataPath</td>
<td>6-159</td>
</tr>
<tr>
<td>GSI_SetIndivNr</td>
<td>6-150</td>
</tr>
<tr>
<td>GSI_SetIndivNrStatus</td>
<td>6-151</td>
</tr>
<tr>
<td>GSI_SetLineMDlg</td>
<td>6-177</td>
</tr>
<tr>
<td>GSI_SetLineMDlgPar</td>
<td>6-179</td>
</tr>
<tr>
<td>GSI_SetLineMDlgText</td>
<td>6-178</td>
</tr>
<tr>
<td>GSI_SetLineSysMDlg</td>
<td>6-172</td>
</tr>
<tr>
<td>GSI_SetMDlgNr</td>
<td>6-174</td>
</tr>
<tr>
<td>GSI_SetQCodeMode</td>
<td>6-154</td>
</tr>
<tr>
<td>GSI_SetRecMask</td>
<td>6-164</td>
</tr>
<tr>
<td>GSI_SetRecMaskNr</td>
<td>6-156</td>
</tr>
<tr>
<td>GSI_SetRecOrder</td>
<td>6-156</td>
</tr>
<tr>
<td>GSI_SetRecPath</td>
<td>6-157</td>
</tr>
<tr>
<td>GSI_SetRunningNr</td>
<td>6-149</td>
</tr>
<tr>
<td>GSI_SetWiEntry</td>
<td>6-162</td>
</tr>
<tr>
<td>GSI_UpdateMDlg</td>
<td>6-181</td>
</tr>
<tr>
<td>GSI_UpdateMeasurment</td>
<td>6-182</td>
</tr>
<tr>
<td>Input</td>
<td>2-52</td>
</tr>
<tr>
<td>Kill</td>
<td>2-68</td>
</tr>
<tr>
<td>MkDir</td>
<td>2-66</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MMI_DataTypes</td>
<td></td>
</tr>
<tr>
<td>MMI_AddButton</td>
<td></td>
</tr>
<tr>
<td>MMI_AddGBMenuButton</td>
<td></td>
</tr>
<tr>
<td>MMI_BeepAlarm, MMI_BeepNormal, MMI_BeepLong</td>
<td></td>
</tr>
<tr>
<td>MMI_CheckButton</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateGBMenu</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateGBMenuItem</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateGBMenuItemStr</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateGBMenuStr</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateGraphDialog</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateMenuItem</td>
<td></td>
</tr>
<tr>
<td>MMI_CreateTextDialog</td>
<td></td>
</tr>
<tr>
<td>MMI_DeleteButton</td>
<td></td>
</tr>
<tr>
<td>MMI_DeleteDialog</td>
<td></td>
</tr>
<tr>
<td>MMI_DeleteGBMenu</td>
<td></td>
</tr>
<tr>
<td>MMI_DrawBusyField</td>
<td></td>
</tr>
<tr>
<td>MMI_DrawCircle</td>
<td></td>
</tr>
<tr>
<td>MMI_DrawLine</td>
<td></td>
</tr>
<tr>
<td>MMI_DrawRect</td>
<td></td>
</tr>
<tr>
<td>MMI_DrawText</td>
<td></td>
</tr>
<tr>
<td>MMI_FormatVal</td>
<td></td>
</tr>
<tr>
<td>MMI_GetAngleRelation</td>
<td></td>
</tr>
<tr>
<td>MMI_GetAngleUnit</td>
<td></td>
</tr>
<tr>
<td>MMI_GetButton</td>
<td></td>
</tr>
<tr>
<td>MMI_GetCoordOrder</td>
<td></td>
</tr>
<tr>
<td>MMI_GetDateFormat</td>
<td></td>
</tr>
<tr>
<td>MMI_GetDistUnit</td>
<td></td>
</tr>
<tr>
<td>MMI_GetLangName</td>
<td></td>
</tr>
<tr>
<td>MMI_GetLanguage</td>
<td></td>
</tr>
<tr>
<td>MMI_GetPressUnit</td>
<td></td>
</tr>
<tr>
<td>MMI_GetTempUnit</td>
<td></td>
</tr>
<tr>
<td>MMI_GetTimeFormat</td>
<td></td>
</tr>
<tr>
<td>MMI_GetVAngleMode</td>
<td></td>
</tr>
</tbody>
</table>
GeoBASIC Reference Manual

Appendix J — List of Predefined Identifiers

MMI_GetVarBeepStatus .......................................................... 6-57
MMI_InputInt ........................................................................ 6-39
MMI_InputList ................................................................. 6-41
MMI_InputStr ................................................................. 6-34
MMI_InputVal ................................................................. 6-36
MMI_PrintInt .................................................................... 6-33
MMI_PrintStr .................................................................... 6-29
MMI_PrintTok .................................................................... 6-30
MMI_PrintVal ................................................................. 6-31
MMI_SelectGBMenuItem ...................................................... 6-18
MMI_SetAngleRelation ....................................................... 6-60
MMI_SetAngleUnit .......................................................... 6-62
MMI_SetCoordOrder ........................................................ 6-73
MMI_SetDateFormat ........................................................ 6-70
MMI_SetDistUnit .......................................................... 6-64
MMI_SetLanguage .......................................................... 6-74
MMI_SetPressUnit .......................................................... 6-66
MMI_SetTempUnit .......................................................... 6-68
MMI_SetTimeFormat ......................................................... 6-71
MMI_SetVAngleMode ......................................................... 6-61
MMI_StartVarBeep ............................................................ 6-55
MMI_SwitchAFKey ............................................................ 6-58
MMI_SwitchIconsBeep ........................................................ 6-59
MMI_SwitchVarBeep .......................................................... 6-56
MMI_WriteMsg .............................................................. 6-45
MMI_WriteMsgStr ............................................................ 6-47
Open ................................................................................. 2-48
Print .................................................................................. 2-54
Put – values ........................................................................ 2-58
Receive ............................................................................. 2-76
RenameDir ................................................................. 2-74
RenameFile ................................................................. 2-73
RmDir ................................................................. 2-67
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seek</td>
<td>2-61</td>
</tr>
<tr>
<td>Send</td>
<td>2-75</td>
</tr>
<tr>
<td>Tell</td>
<td>2-60</td>
</tr>
<tr>
<td>TMC Data Structures</td>
<td>6-97</td>
</tr>
<tr>
<td>TMC_DoMeasure</td>
<td>6-101</td>
</tr>
<tr>
<td>TMC_Get/SetAngleFaceDef</td>
<td>6-119</td>
</tr>
<tr>
<td>TMC_Get/SetAtmCorr</td>
<td>6-123</td>
</tr>
<tr>
<td>TMC_Get/SetDistPpm</td>
<td>6-121</td>
</tr>
<tr>
<td>TMC_Get/SetGeomProjection</td>
<td>6-122</td>
</tr>
<tr>
<td>TMC_Get/SetGeomReduction</td>
<td>6-122</td>
</tr>
<tr>
<td>TMC_Get/SetHeight</td>
<td>6-123</td>
</tr>
<tr>
<td>TMC_Get/SetHzOffset</td>
<td>6-120</td>
</tr>
<tr>
<td>TMC_Get/SetRefractiveCorr</td>
<td>6-124</td>
</tr>
<tr>
<td>TMC_Get/SetRefractiveMethod</td>
<td>6-124</td>
</tr>
<tr>
<td>TMC_Get/SetStation</td>
<td>6-125</td>
</tr>
<tr>
<td>TMC_GetAngle</td>
<td>6-110</td>
</tr>
<tr>
<td>TMC_GetAngle_WInc</td>
<td>6-111</td>
</tr>
<tr>
<td>TMC_GetAngSwitch</td>
<td>6-131</td>
</tr>
<tr>
<td>TMC_GetCoordinate</td>
<td>6-107</td>
</tr>
<tr>
<td>TMC_GetDistSwitch</td>
<td>6-127</td>
</tr>
<tr>
<td>TMC_GetFace1</td>
<td>6-130</td>
</tr>
<tr>
<td>TMC_GetInclineStatus</td>
<td>6-132</td>
</tr>
<tr>
<td>TMC_GetInclineSwitch</td>
<td>6-132</td>
</tr>
<tr>
<td>TMC_GetOffsetDist</td>
<td>6-129</td>
</tr>
<tr>
<td>TMC_GetPolar</td>
<td>6-103</td>
</tr>
<tr>
<td>TMC_GetSimpleMea</td>
<td>6-116</td>
</tr>
<tr>
<td>TMC_ifDistTapeMeasured</td>
<td>6-125</td>
</tr>
<tr>
<td>TMC_ifOffsetDistMeasured</td>
<td>6-129</td>
</tr>
<tr>
<td>TMC_QuickDist</td>
<td>6-113</td>
</tr>
<tr>
<td>TMC_SetAngSwitch</td>
<td>6-130</td>
</tr>
<tr>
<td>TMC_SetDistSwitch</td>
<td>6-127</td>
</tr>
<tr>
<td>TMC_SetHandDist</td>
<td>6-126</td>
</tr>
<tr>
<td>TMC_SetInclineSwitch</td>
<td>6-131</td>
</tr>
</tbody>
</table>
TMC_SetOffsetDist .......................................................... 6-128