version 1.5

Programming language for GEOS 64 and GEOS 128 english on 40 and 80 column display

MANUAL

Compiler and accessories by:
Falk Rehwagen

Manual by:
Denis Döhler

GEOS-USER-SOFTWARE-SACHSEN
Welcome to geoCom

We would like to take this opportunity to congratulate you on purchasing this programming system. With this system you have acquired an efficient means to program and develop your very own Geos 64/128 applications. All of the enclosed programs have been carefully developed and have been subjected to an exhausting testing procedure, by a number of experienced users, to ensure that all the programming errors have been removed. Of course experience may prove otherwise, by using different operating configurations it could happen that you find a programming error, if this should be the case - we accept NO RESPONSIBILITY for DAMAGE caused through the use of geoCom and / or the information contained in this Handbook.

If you should find an error in the programs included in the geoCom package, we would be honoured if you would help us to find and isolate the problem. To this end we would require the relevant information from yourself:

What happened, which program and Source-Code were running and what hardware configuration were you using when the problem occurred. Please send all this information to GEO-USER - SOFTWARE - SACHSEN (the address is on the next page).

The geoCom-system does not have an internal anti-copy-protection, which means that you do not need to install the software before you use it, you can also use geoCom with various Geos system-installation numbers. This is meant to enable a simplified programming-work-field but is not to be regarded as a free license to copy the software at will. In order to discourage our customers from copying the geoCom package each individual sold package has it's own internal code number thus enabling us to follow any pirate copy back to it's source ! We believe that after two years of intensive development we are offering geoCom at an attractive price, so why spoil all the good work and deprive us of our fair won reward.

Some of the Programs and their Source-Codes in this software package have been declared as Public Domain (PD) to encourage their use and circulation amongst Geos users.

Here's a list of the geoCom Copyrights:

- geoCom, ObjectEdit, Linker Copyright (C) 1992 - 1994 Falk Rehwagen
- BasicTOgeos Copyright (C) 1993 Denis Döhler
- IconEdit, ShowFont Written (W) 1993 - 94 Falk Rehwagen , PD (free to copy !)
- System-Info, PatternShow,SID-Demo Written (W) 1993 Denis Döhler, PD
  geo 3D Written (W) 1993 Denis Döhler & Falk Rehwagen , PD (free to copy !)

Apart from the above the whole Handbook is protected by Copyright. The Handbook has been prepared and written by Denis Döhler and Falk Rehwagen, the English translation was carried out by Paul MILNER, the Handbook, including extracts thereof, may not be copied.

System Requirements :

Required Hardware:
Commodore 64 or Commodore 128 (D)
Diskstation, Commodore 1541 or compatible (C 1570, C1571 . . .)
Joystick or Mouse or another input device , TV or Monitor

Recommended Hardware:
Commodore 1351-Mouse or a compatible proportional mouse
GEOS-compatible REU.
A diskestation with a higher storage capacity (C 1581, CMD HD, CMD FD)
A printer with graphic capability

Required Software:
GEOS from Version 2.0 upwards
geoWrite from Version 2.0 upwards
Assembler (MegaAssembler or geoProgrammer)
What to do when it doesn't work?

Ok so you have got a problem, it's all working nicely but not at all like you wanted it! Well the main thing is - stay cool and think, think, think!

The first priority is to check out the Hardware - if everything is working alright then the Hardware is OK! Then check out the Software - first Validate the current Disk, this will tell you that the BAM is OK (or not !). You should then erase damaged files (eg. geoWrite) from the Disk and replace them with your pre-prepared backups! So if everything else is OK then the problem must lie with your Source-Code, the only solution here is to check your program carefully and if all else fails a new study session with the geoCom Handbook is called for. When you are really stuck fast then it is possible that we may be able to help you - get in touch with us, including a stamped addressed envelope.

The left hand address is for definitive questions in regard to geoCom programming and the distribution of geoCom. The right hand address is for general questions to geoCom and Geos internal problems. When you write to us PLEASE include sufficient German postage stamps or money, our budget has been calculated very tightly!

Handbook Presentation:
We have specially selected this form of Handbook presentation in order to enable you the user to remove or add pages as required - we plan to add items as the level of experience with geoCom expands.
**Handbook, List of Contents:**

| Welcome to geoCom | Page 1 |
| System Requirements | Page 1 |
| What to do when it doesn't work? | Page 2 |
| Further Reading | Page 2 |
| Handbook, List of Contents | Page 3 |
| Disk Contents | Page 5 |

**Chapter 1:**

1.0 geoCom - general | Page 6 |
1.1 Writing the Source-Code | Page 6 |
1.1.1 The Text Editor | Page 6 |
1.1.2 The Source-Code Name | Page 6 |
1.1.3 The Source-Code Structure | Page 6 |
1.1.4 Dialog Boxes, Menu Bars - making you own | Page 7 |
1.1.5 Writing single Section Programs | Page 7 |
1.1.6 Writing Multiple Section Programs | Page 8 |
1.2 geoCom-Operating Procedures | Page 8 |
1.2.1 Calling geoCom | Page 8 |
1.2.2 The Screen Display | Page 8 |
1.2.3 The Compiler | Page 9 |
1.3 The Finished Program | Page 9 |
1.4 geoCom - Additional Programs | Page 10 |
1.4.1 Starting geoCom | Page 10 |
1.4.2 Errors, When Starting geoCom | Page 10 |
1.4.3 Start Editor | Page 10 |
1.4.4 Errors, When Using The StartEditor | Page 11 |
1.5 ObjectEdit | Page 11 |
1.5.1 Calling ObjectEdit | Page 11 |
1.5.2 The Structure | Page 12 |
1.5.3 Object Construction | Page 13 |
1.6 ICON EDIT | Page 19 |
1.7. BASIC TO GEOS | Page 20 |
1.8 LINKER | Page 21 |
1.9 Program Examples | Page 21 |
1.9.1 PATTERN SHOW | Page 21 |
1.9.2 SHOW FONT V1.4 | Page 25 |
1.9.3 SYSTEM INFO | Page 26 |
1.9.4 Geo 3-D | Page 26 |
1.9.5 SID-Demo’s | Page 26 |

**Chapter 2**

2.0 Commands, General Information | Page 28 |
2.0.1 Variable Requirements | Page 29 |
2.0.2 Pre-Select Variables | Page 29 |
2.0.3 Loading Variables From Disk. | Page 31 |
2.1 Programming-Commands | Page 33 |
2.1.1 BASIC - Oriented Commands | Page 33 |
2.1.2 Program - Loop-Commands | Page 38 |
2.2 Program - Structure Commands | Page 39 |
2.2.1 Source-Code - Parameter-Commands | Page 39 |
2.2.2 Variables - Memory Areas | Page 40 |
2.2.3 Source-Code - Module-Commands | Page 41 |
2.2.4 Dialogbox - Commands | Page 42 |
2.2.5 Menu Bars - Commands | Page 44 |
2.3 Process-, Memory-, Machine-Code - Commands | Page 44 |
2.3.1 Process - Commands | Page 44 |
2.3.2 Memory - Commands | Page 45 |
2.3.3 Machine-Code - Logging In | Page 46 |
2.4 Disk. - and File - Commands | Page 46 |
2.4.1 Disk. - Commands | Page 46 |
2.4.2 File - Commands | Page 47 |
2.5 Printer - Commands | Page 49 |
Chapter 3:
3. Description Of The GEOS-System
   3.1 Memory Occupation  Page 58
       3.1.1 The Basics  Page 58
       3.1.2 Different Versions ?  Page 58
       3.1.3 Fundamental Memory Occupation  Page 59
       3.1.4 Zero Page  Page 59
       3.1.5 GEOS Variables  Page 60
       3.1.6 Further Important Memory Positions  Page 65
       3.1.7 The Screen Memory  Page 65
   3.2 Codes  Page 65
       3.2.1 Keyboard - Codes  Page 65
       3.2.2 Character - Codes  Page 67
       3.3 Disk Format  Page 67
       3.3.1 BAM  Page 67
       3.3.2 Directory  Page 68
       3.3.3 A Directory Block  Page 68
   3.4 File Formats  Page 69
       3.4.1 InfoBlock  Page 69
       3.4.1.1 Sequential Files  Page 69
       3.4.1.2 VLIR Files  Page 69
   3.5 geoWrite Documents  Page 69
       3.5.1 InfoBlock  Page 69
       3.5.2 Index - Sector  Page 70
       3.5.3 Data Unit Structure - geoWrite V1.1  Page 70
       3.5.4 Data Unit Structure - geoWrite V2.0/V2.1  Page 70
   3.6 Further Text Formats  Page 70
       3.6.1 TextScrap - Structure  Page 71
       3.6.2 Notice Block - Structure  Page 71
   3.7 Definitions  Page 71
       3.7.1 Graphic - Escape  Page 71
       3.7.2 NewCardset  Page 71
       3.7.3 Ruler - Escape  Page 71
   3.8 Graphics  Page 71
       3.8.1 geoPaint - Picture - Structure  Page 71
       3.8.2 Photo Scraps - Structure  Page 72
   3.9 Font - Structure  Page 72
       3.9.1 InfoBlock  Page 72
       3.9.2 Structure  Page 72
   4.0 Geo - Error Messages  Page 73
       4.2 geoCom - Error Messages  Page 78
   5.0 Appendix - A  Page 80
       Conversion, Hex - Dec  Page 80
Disk Contents:

Disk A:

Disk side a:
- DeskTop Page 1
- GEOCOM
- Start geoCom
- Start Editor
- BASIC TO GEOS
- OBJECT EDIT
- LINKER
- ICON EDIT

Disk side b:
- DeskTop Page 1
- SHOW FONT
- SYSTEM INFO
- GEO 3D
- SHOW FONT_com
- SYSTEM_com
- Geo 3D_com

DeskTop Page 2
- definitions_ext
- icon_system
- definitions_ext+

DeskTop Page 2
- ICON EDIT_com
- ICON EDIT_obj
- ICON EDIT_msc
- ICON EDIT_asm
- LINKER_com
- LINKER_obj
- LINKER_msc
- LINKER_asm

Disk B:

Disk side a:
- DeskTop Page 1
- PATTERN(a)_com
- PATTERN(a)
- PATTERN(b)_com
- PATTERN(b)
- PATTERN(c)_com
- PATTERN(c)
- PATTERN(d)_com
- PATTERN(d)

DeskTop Page 2:
- PATTERN(e)_com
- PATTERN(e)
- PATTERN(f)_com
- PATTERN(f)
- PATTERN(g)_com
- PATTERN(g)
- PATTERN(h)_com
- PATTERN(h)

DeskTop Page 3
- PATTERN_com
- PATTERN SHOW

- GeoCom V1.5 - Handbook -
1.0 geoCom - general

Using geoCom you can program the following File types:
- One part Applications, like PrinterEdit
- Multiple part Applications, like geoWrite
- Self starting programs, like SET CLOCK

If you are planning to work with Geos 128 you can do this in the 80 character mode and develop programs for both the 40 and 80 character modes. geoCom includes special codes to enable you to carry out these functions but we recommend C-64 Users only to program 40 character modus programs - there is no way he can check an 80 character program for errors! geoCom works together with Geos 64 V2.0/V2.5 and Geos 128 V2.0 (40 and 80 character modes) By using geoCom it is possible to program a file that includes the 40/80 switch box.

Programming using geoCom is carried out in four steps:
a. Using the Text Editor - create the Source-Code.
b. Using geoPaint - create Graphics, Sprites etc.
c. Develop the Menu Bars and using ObjectEdit - tie in the Bitmap Graphics
d. Using geoCom - mix the ingredients to create the finished program

Warning! If you have installed GeoHexer this will lead to problems with geoCom. This means that you cannot run geoCom if GeoHexer is running!

We strongly recommend you to prepare backup discs from the original geoCom discs, for your own safety it is better to work with the copies and keep the original discs in reserve. We recommend you to make a work disc with the required files and to copy this, complete, into the REU before starting work with geoCom. The work disc should include:
- geoCom - Start geoCom - Start Editor - ObjectEdit - IconEdit
- geoWrite - Text-Printer Driver - required Fonts and Desktop

1.1 Writing the Source-Code
1.1.1 The Text Editor
You will not be able to find a text editor on the geoCom discs, you have already bought it together with your Geos system - it is geoWrite! Using geoWrite you can design a page of your Source-Code as it pleases you. With geoWrite you have such wonderful tools (CUT, PASTE, SEARCHING -> REPLACE) these are without equal in the Geos world.
You can use various fonts and styles. The only rules that you have to observe is that:
1) you may never have more than 128 characters in a line and
2) you are only allowed one Command and its relevant Parameter in a line
this includes spaces, text formatting and the fonts.

1.1.2 The Source-Code Name
When you want to write a new program, the first step is to create a new geoWrite document. You can specify a name of your choice for the new document, however you must select a name other than that of the planned program because Geos saves programs according to their names and the Source-Code would thus be overwritten during the compiling phase if the name of the the document where to be identical with that of the new program. We recommend you to give your Source-Code docu's the suffix "_com", this will enable to keep your geoCom Source-Codes separated from your other geoWrite documents.

1.1.3 The Source-Code Structure
GeoCom are entered without line numbers. This may be new for some of you but it simplifies matters. The various subroutines are not accessed with GOTO Line Number, instead they are accessed with GOTO/GOSUB LABEL. Due to the absence of the line numbers the problem with the awkward renumbering of a changed sequence has been overcome. The Source-Code is made up of three parts:

Definitions Section
The definitions section must include the name of the program, the appropriate class and the authors name. Additionally the system variable STARTFLAG, which dictates which modi the program can run in (only for C-128!) will be included. This Data is all very important for the Info-Block. You can also include the memory allocation variable at this point. In the original start up settings the following allocations are possible (if you haven't changed them!):

- PAGE 6 -
Declarations Section
In this section the available variables will be defined, which variables are to be declared as Integers, String- or Byte-Variables. All the Label Names must be included in this section. If you have created Menu Bars, Icons etc with ObjectEdit, or you want to do so, then you will have to include the declaration OBJFILE "Name_obj" and its call up in this section. Later during the compilation phase this object file will be read and slotted into the completed program. Please pay regard to the fact a maximum of 127 names for variables, labels, files etc can be held in the memory at one time (for both single and multiple-part programs).

Instructions Section
The instructions section includes the information for the screen image and mainloop followed by the labels for the menu bars, dialog boxes and their relevant sub routines. Each label is announced by the character ‘’, this character is followed by the actual label name (Eg. start). You can find this character as <SHIFT 3> It is possible to write more than one command in a single line, the commands must be separated with a ‘:’.

The instructions section is normally the largest part of the Source-Code. You can write the 3 three parts of the Source-Code directly one after another, there is no requirement to separate the parts. It is only required to write the three parts in the right order, this is a requirement of geoCom. You can see the construction best by studying the demo's.

1.1.4 Dialog boxes, Menu bars - making your own
Using ObjectEdit you can make and save all your own click boxes, menu bars, special feature boxes and user icons. The standard boxes, menu bars and file select boxes are provided for your use by geoCom. The ready-use graphics are defined within the Source-Code and inserted into the new program by geoCom. User icons (eg. Arrows) are developed first using geoPaint and then imported into ObjectEdit using the PhotoScrap-Format, they are then given certain parameters and finally tied into the _obj-code. You can see that we have deliberately tried to use as many aspects of standard Geos to enable you to make the best use of your familiar environment and also to ensure that everything fits together with the least fuss!

ObjectEdit is an Application and can be started with the familiar double click on the program icon. Start geoCom -> Objectedit or with geoCom -> ObjectEdit without Object, with Object. It is possible to move directly from Object Edit to geoCom with Start geoCom and vice versa. Please be careful when inserting the parameters, mistakes here can lead to problems later.

There are virtually no restrictions when defining menu bars. When using ObjectEdit it is possible to define parameters and data entries that can be included in your new programs. Eg. other fonts and font sizes, file headers and comparison tables etc. The whole package is tied together in a single file and must be read in at the beginning of your Source-Code.

1.1.5 Writing Single Section Programs
Single section programs are loaded complete into memory, an example for this type of program is PRINTEDIT. You have the advantage that all labels and variables are always available, the disadvantage is simple - the program is limited as to its maximum size. You are not allowed to use any module commands in a single section program. After the program has been compiled you must save the program yourself, otherwise it will only remain in the memory from geoCom and be scratched when you leave geoCom. Take a look at the program "ShowFont" for clarification.
1.1.6 Writing Multiple Section Programs

When you start a multiple section program only a small part of the program is loaded into the memory. A good example for a multiple program is geoPublish, this program is much larger than the entire memory of a C-64! Multiple programs are split into a series of modules and only the presently required section is loaded into the memory.

For this purpose a multiple section program must have a global section and a number of local sections. The global section is responsible for overseeing the whole program and contains the name of the program, its class and the author's name. The global section also contains all the repeating variables and labels. The local sections contain particular parts of the program (similar to sub routines), loops and variables that will only be required in the same local area. Please bear in mind that a second local section completely overwrites a previously active section when loaded into the memory. You can only hold a maximum of 127 different variables, label names etc. in the memory simultaneously. Don't forget to include the global variables in your calculations! A good example of a multiple section program is the original DeskTop (from BSW), eg. If you want to see the contents of an Info-Block (keycode C= Q>) the next module is first loaded into the memory.

Bear in mind when programming that when a program is started the global section is leapfrogged and the program is commenced with the module "0". Therefore module 0 must contain the screen information, important parameters and commands.

1.2. geoCom - Operating Procedures
1.2.1 Calling geoCom

Starting geoCom from the Desktop

When you start geoCom with a double click on the program icon you will be presented with a file choice box, using this box you can select the required Source-Code-Text. If there are more than 6 geoWrite documents on the disk, then the box will be complemented with a scroll bar, the first 15 files from a disk. are available. Using DISK and DRIVE you either change the floppy drive and/or the disk. With Cancel you go to the geoCom main menu. It is possible using the F-keys (F1/F3/F5/F7) to switch between the floppy drives A - D, you can also access to a dialog box which offers you the opportunity to start an error list or not. If you don't access the Cancel option the Source-Code will be read in and compiled.

Starting geoCom with Start geoCom

You can read about the operating procedure of Start geoCom at chapter 1.4.1. If you access geoCom out of geoWrite (using Start geoCom) the present document will be accepted as a Source-Code and compiled - if you have used the start editor to define a file. When starting geoCom with Start geoCom the pre-select C-128 screen modus will be activated before starting geoCom. If you have defined the option "Start compiling source" - an error list will be automatically made while geoCom is compiling the Source-Code or a choice box will appear beforehand.

1.2.2 The Screen Display

When you start geoCom with "Start compiling source" then only the lower part of the main menu will be displayed. In the first line you will see the Name of the Source-Code, the Page, how far advanced the translation is and which Character is being checked now.

<table>
<thead>
<tr>
<th>Source: Beispiel 16_com Page: 1 Cmd: 28 Errors: 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective: name Mod: 1 Obj-code: another</td>
</tr>
<tr>
<td>Code area: $2800 - $2808 (max. $284b)</td>
</tr>
<tr>
<td>Constants area: $284c - $284e (max. $28a5)</td>
</tr>
<tr>
<td>Variables area: $2806 - $2807 (max. $280a)</td>
</tr>
</tbody>
</table>

It is also possible that instead of Character, the word Errors is present. This means that you pre-selected the automatic making of an error list. In the second row can be seen the Name of the found object file, the present Module-No. and the Object Code. Under this in the third line can be seen the Code area. In the fourth line the fixed parameters can be seen, and in the fifth the variables. These values (Max. Value) : the pre-selected memory area can either be determined by the programmer in the declarations section or defined internally by geoCom. During the compiling procedure it is possible to see how geoCom is sorting the variables.

After the translation has been completed you can see how much and what memory areas have been used as a final value, thus you have the opportunity to alter the standart
memory allocations, increasing or decreasing as required. If the variable, Code or constant parameter memory area is to small then this will be indicated by an error message. It is possible using the upper dialog box to access various geoCom functions:

Other source file - With the file selection box (all geoWrite-files on the current disk) it is possible to access another Source-Code. Don't forget to save an already compiled program first !
Save program - After the program has been compiled you can either start the program or save it, we recommend you to save the program first !
Open Error file - When you have created an error protocol it is not possible to save the program. Therefore instead of "Save Program" you now have the opportunity to start the error protocol, this is a geoWrite text that includes the errors found during the compilation.
Run Program - Refer to Save Program !
To Desktop - Return immediately to the Desktop (No safety check !) a program that has not been saved beforehand will be lost !
Without text to geoWrite - You will land in the start box from geoWrite (Create, Open - Desktop).
With text to geoWrite - You will start geoWrite and the Source-Code-Text will be opened.
Without object to ObjectEdit - You will land in the start box from ObjectEdit (Create, Open, Quit)
With object to ObjectEdit - You will start ObjectEdit and the relevant data document will be opened, or the first data document on the disk.

If the text segments are shown in Italics then the functions are not available, they are only available so long as they are visible in "plain style". If, in the StartEditor, you have activated the option Set template then you will see when starting ObjectEdit or geoCom a short options list where you can choose either to start geoCom or ObjectEdit. It is possible to move between geoWrite, ObjectEdit and geoCom without having to go through the Desktop.

1.2.3 The Compiler
Compiling a text means that the Source-Code will be combined with the Data-Document, converted to Machine-Code and saved internally in geoCom as a program. During the period that the compiler is active you will see the lower part of the main menu, geoCom will load the Source-Code-Text and convert it. Single section programs are not compiled on disk and only when the Source-Code-Text is to large will it be divided into sections and deposited on the disk. After the translation you will have to do this yourself, or start the new program. Please bear in mind that the required object files and the module Source-Code (if required) must be present on the same disk.
Multiple section programs are, after the successful compilation of the second module, automatically saved to disk. Please make sure that you have sufficient space on the disk !
If required, it is possible to save any errors, that occurred during the compilation, in a new geoWrite document which will be created by geoCom. This document will be called "GuelTextname.err". After the compilation has been completed and no errors have been detected then the geoCom main menu will appear with the option "Save Program". The Compiler can be stopped at any time with the <RUN/STOP> key.

1.3 The Finished Program
During the saving period an Info-Block will be created, this includes the data from the declarations section and a standard icon. Of course you can alter this icon using the IconEditor - the IconEditor has actually been written using geoCom !
You have now received a sound foundation to geoCom and its peripherals, the time has now come to inform you about the "bad side" of geoCom. Your ready-use new program will includes ca. 10 KB of Geos standard routines, only with their help is it possible to combine all the BASIC (similar to) commands and the Machine-Code routines. Your whole Source-Code will be saved as Machine-Code, this means that the program will have an average size of 15-20 KB. This is unfortunately the only disadvantage of geoCom, it is not possible to write very small programs as is the case with pure Machine-Code - this wasn't our target, we didn't want to write a second Mega-Assembler.

1.4 geoCom - Additional Programs
1.4.1 Starting geoCom
To enable you to be able to start GeoCom or ObjectEdit out of geoWrite we have written Start geoCom is a Desk Accessory (DA). Before you activate Start geoCom you should first update your geoWrite docu. (to be found in the File-Bar), only so can you be sure that the latest version of your document is being activated by geoCom. If you forget to update your document beforehand a dialog box will appear, warning you of this fact. GeoCom will now be loaded and started, according to your pre-selects, the individual pre-select options are described at chapter 1.4.3 (Start Editor).
We advise you to always start geoCom with Start geoCom, because Start geoCom is only a loader you will only see it on screen if there is an error or a problem during the loading sequence.

1.4.2 Errors, When Starting geoCom
- If you don't carry out an update the following box will appear:

```
Start geoCom V1.3 - (C) 1992-93 Folk Rehungen
Please call first the update-point in the file-menu.
Abbruch
```

With Cancel you can rectify this problem.

- If you don't start from geoWrite you will see this box:

```
Start geoCom V1.3 - (C) 1992-93 Folk Rehungen
The program wasn't called from geoWrite V2;
Are you sure, that the document is up to date?
OK Abbruch
```

Here you can choose between OK - carry on or Cancel.

- If geoCom (or ObjectEdit) cannot be found the following box will appear:

```
Start geoCom V1.3 - (C) 1992-93 Folk Rehungen
Are you sure, that the document is up to date?
Which application do you want to call?
geoCom ObjectEdit Abbruch
```

Make sure that the required program is in either floppy A or B. In order to do this and find the required program you can leave this box with the Cancel option.

1.4.3 Start Editor
The Start Editor enables you to select various values and functions that will be initialized by Start geoCom. To set up the program you must copy geoCom and Start Editor to your work disk or REU and then start Start Editor. You can either start Start Editor from the Geo Menu Bar (also present in geoWrite) or with the usual double click on the Start Editor program icon. the individual items can be selected (de-selected) by activating the buttons. (Black button = activated)
Possibilities to call:
You can choose here, should GeoCom or ObjectEdit be started by Start geoCom. If you start with GeoCom then you will be able, directly after the start, to go straight into the compilation modus by clicking on : Start compiling source. If not you will enter the GeoCom main menu. If you select the item Start compiling source then any errors that appear during the compilation will be saved in the error text. The Main file has a special function, if you enter here the name of a particular file then this file will always be loaded and compiled, it makes no difference from which Source-Code text you start GeoCom. E.g. if you have written a program with a number of modules and you have set each module up with it's own Source-Code Text, it would be advantageous to give as standard file the name of the main program, in the main program the command INCLUDE . . . "where the blank spaces represent the name of the module Source-codes. GeoCom will here insert the Source Code and so complete the program!

C128-Screen
Here you can set up the 40/80 character modus (only with a C-128) to switch to before the start of GeoCom/ObjectEdit. With the button Switch box you can determine whether a respective switch box should appear.

Options
You must save your pre-selects. It is possible to read pre-selects that you have already saved. When you select the option Set template you can determine after the start of Start geoCom if geoCom or ObjectEdit should be started. You only really need the start editor if you want to change the Start geoCom pre-selects.

1.4.4 Errors, When Using The Start Editor
If Start geoCom cannot be found on the same disk, the following message will appear:

Check that Start Editor and Start geoCom are on the same disk.

1.5. ObjectEdit
ObjectEdit is a very important tool. With ObjectEdit you can:
- develop your own dialog boxes
- develop menu bars
- insert the data for file headers (important for creating new files)
- declare the comparison tables (eg. keyboard hotkey commands)
- tying in (made with geoPaint) mini graphics - (Bitmaps)

Everything that you develop or tie in with ObjectEdit is saved in a data file (Machine-Code) and must be tied into the geoCom Source-Code with the command OBJFILE name, byte (in the Source-Code). The tied in bitmaps are called up, shown and inserted into the Source-Code with OBJECT name and the menu bars with MENU name, byte. ObjectEdit only functions in 40 char. modus. Please bear this in mind!

1.5.1 Calling OBJECT EDIT
Calling from the Desktop
The usual start is with the well known double click on the program icon. OBJECT EDIT
starts and the main screen appears. You will see a choice box with the usual same options as in geoPaint, geoWrite etc. New Doc., Existing Doc or return to Desktop.

- Create new Document:
  If you want to create a new object document then you have to enter a name in for the new document in the follow-on box. We recommend you to choose a name with the suffix _obj, this will enable you to keep a good overview of your object documents.

- Open existing Document:
  When you wish to open an object document that already exists you must select (Open) one of the files that are presented in the follow-on box, the first fifteen files on the disk, will be shown. You can also start an existing document when you perform a double click on its icon in the Desktop.

- Quit to Desktop:
  You return to the Desktop

Starting from Start geoCom
The operating procedures of Start geoCom can be read at chapter 1.4.1. According to your pre-selects, that you have set up in the Start Editor, you will arrive at ObjectEditor. Please bear in mind that the data contents of a document cannot be printed.

1.5.2 The Structure
When you start ObjectEdit the main screen will appear. If you start ObjectEdit without an object document a Copyright logo will appear in the lower part of the main screen. If you start ObjectEdit from an object document the logo will not appear. If you have attempted to create a new object document the following message will appear at the bottom of the screen:

None objects in this file!

In the top left hand corner you will see the name of the present object document and in the top right corner is the menu bar. The menu bar options are:

- geos
  - ObjectEdit Info: The usual "author box" will appear.
  - Drive (active drive): With this option you are able to directly switch between your various drives. You can switch onto the next following drive. Please bear in mind that with this option you could receive a drive conflict, OBJECT EDIT always starts with the current drive and during the work on the object document the program will often move blocks of information back and forth between the memory and the disk, if you have changed the drive then it could happen that he program could no longer find the relevant data - ZONK !!!
  - The disk, and memory relevant DA's will be shown. It is possible using the Drive option to access DA's on other drives.

File
- close: The presently active document will be updated, saved to disk, and then closed, the options box: "New, Open, Cancel" will appear so enabling you to work with a further document.
- update: We strongly recommend you to regularly update your document, analog to geoWrite, geoPaint etc.
- preview: Shows you how many objects are in the active document.
- recover: If you have made a change to the document which you are not happy with, you can return to the document state at the last update.
- GeoCom V1.5 - Handbook -

- quit: The document will be updated, closed and you will return to the DeskTop.

Edit
- cut / copy / paste: This is the same as in geoCalc and refers to the individual objects. The functions enable you to erase, copy and insert objects that you have previously prepared in other documents, so saving you a lot of tedious parallel work. The function makes an Object_Scrap.
- drive (A): Enables you to switch to another drive, this is particularly useful when you are looking for a Scrap that is on another disk, in a drive other than the present drive. The switch is possible in the logical direction: A - B - C - D. Don't forget to switch back to your "working drive."
- delete object: Erase the actual object from the object docu. Careful, if you haven't saved the object with cut/copy as an Object-Scrap it will be lost! Excepting: You can pull the object back with back-date (as long as you have not carried out an update beforehand). - A check box appears before you can carry out this function!
- new object: A further object will be inserted into the object docu. A dialog box will appear asking you which type of object you wish to insert. You must give the object a name otherwise the object will be known as unnamed, always make a note of the object name, you have to insert the object with the command OBJECT name in the Source-Code.

Options
Here you can find the specific object declarations: BITMAP, SPRITE, ICON, FILEHEADER. Eg. Insert Fotoscrap . . . etc. It is basically possible to load all graphics as a PhotoScrap, with SPRITES and FILEHEADER it is possible to use the available Pixel Editor, this is very simple - you only need to place the mouse pointer on the required position - with a click, a pixel will either be placed or erased.
TIP: you are recommended within geoPaint to work in the colour-modus, activate the colours black and white, before you "cut" the graphic - switch back to the monochrome modus, this makes life a lot easier!

1.5.3 Object Construction
In order to create single objects, go to the option new object in the menu Edit, a dialog box will appear, make your selection by clicking on the relevant object-type. OBJECT EDIT will place the object in the object document:

Please select object type of new object:
- bitmap
- dialog box
- fileheader
- byte row
- real row
- processes
- sprite
- icons
- menu
- string kette
- integer row
- string row
- font

At the bottom of the screen you will see: left, the object type, label, the present sum total of objects in the docu. and the position no. of the actual object. Using the two arrows on the right hand side it is possible to switch between the objects. Enter as Name the name of the active object, click on the name area and the Text Cursor will appear on unnamed. After you have edited the object name enter <RETURN>. Later you will insert this object in the Source-Code with the command OBJECT name. The object docu. will be called up beforehand with the command OBJFILE "Object-Document".

fileheader (or Info.-Block)
When you want to create your own programs in GEOS you will require a Fileheader (Info.-Block). You can read about the construction and the placement of the individual bytes in chapter 3.4.1. OBJECT EDIT and geoCom can save you a lot of tedious work here! Important: in geoWrite you will have to enter a number of extra bytes (eg. the first page number). The new file will be created in the Source-Code using the command CREATE Filename,Name. Filename is the name of the file that is to be created and Name is the label that you have used under geoCom for the file header. The Screen is here split into 4 parts, upper left is the Icon-Editor. With the function read PhotoScrap . . . (to be found in the Option menu) it is possible to insert an already completed icon. Upper right can be seen the area where the bytes 68 - 70 and 96 of the
Info.-Block are given their parameters, it is possible to alter the standard parameters by adjusting the arrows.

Below left can be seen the area where the Bytes 71 - 93 and 97 - 133 of the Info.-Block are entered/adjusted, here it is important to pay attention to the special requirements of GEOS. We strongly recommend you to take a look at other GEOS Info.-Blocks, this can be done using a disk monitor and is especially relevant to the CLASS-Name and the Applications-Name. This is particularly important if want to be able to start your program with a double click on the program icon.

In the area below right you have the option to define the remaining bytes (134 - 169) of the Info.-Block. (Important for geoWrite docus. - 1st. page). It is not possible here within the program to enter the info.block text segment, you can only do this by calling up the info.block in the Desktop <C / Q>. Move the cursor over the area, click, now you can change the value.

byte-, integer-, constant- and string row
These variable sequences are for digit sequences (and strings) that you often use for calculations and displays, this enables you to save expensive Code-Area-Space. Enter first the number of data elements or use the arrows. Finally enter the required values for individual elements. Click on the respective area, the text cursor will appear and the contents of the element will be shown, format: (Name <Data element>). With the help of the menu option Option it is possible within the data element row to erase or import individual data elements:
- delete data entry = All the following data elements move up one position. The no. of elements will be reduced by one. The first data element cannot be erased.
- insert data entry = All the following data elements move down one position. The no. of elements will be increased by one.

You can simplify the input of the individual data elements by using the key combination <C= RETURN>. The cursor jumps immediately into the editor area. After you have entered your parameters press RETURN, the parameter will be accepted and the data element counter will increase by one.

sprites
The GeoCom package includes a simple sprite editor. It is also possible with the command: read Photoscrap., to import a pre-prepared Photoscrap as a sprite. The sprites are 24 x 24 GEOS points large. The creation / changing of a sprite is easy, click -
inside the editing window - on a particular point, if there was a spot it will now be erased and vice versa. Left from the editing window can be seen the sprite in its original size.

The actual sprites are activated with the sprite commands in the Source-Code.

**Bitmaps**
Bitmaps are graphics, they often help more than a whole book. Bitmaps can be used as logos (GeoHexer), eg. the use of a bitmap as an icon in a dialog box. A bitmap is created with a (artistic) graphics program (geoPaint). Simply cut the graphic out of geoPaint doc’ as a Photoscrap and import it here. For this function you will need to activate the menu option **Photoscrap**. this can be found in the menu **Option**. A dialog box will appear containing two values, **Width** and **Height**. The values are very useful eg. for the creation of a icon tables (ICON).

**Font**
Because of problems with memory capacity it is not advisable here to import other other fonts and font sizes, notwithstanding we have included this ability. With the command **Font Name** they can called up into the Source-Code and then used. For the import of fonts you will find the option **read font file** in the menu **Option**, first you should select the font and then the font size. The font size will be imported and saved as an object. If you enter here an illegal value (non existent font size) you will receive an error message. After the font (ie. the font size) has been completely imported, you will be shown a demo. of the font with all the possible characters.

**Processes**
After selecting this option a table will appear, here you must insert the values that correspond with all your processes (they are included in the Source-Code as labels). The values are : the total number of processes, the running no. of each individual process and it's corresponding label and the call up frequency. To simplify entering the data the keycodes `<C> RETURN>` and `<RETURN>` are available. Within the Source-Code the processes are called up using the command **PROCESS** and started with the command **RESTART byte**. The Process time refers to the interrupt, the standard value relates to 1/50 sec. ie. Value 50 = 1 call up per second. By activating additional commands in the Source-Code it is possible to have an added level of control over the individual processes. The value **byte** is the no. of the active process. Please bear in mind, the length of the mainloop increases analog to the number of active processes.

With the command **clear current entry** in the **option** menu it is possible to erase the current process from the process table. All the following processes move up one position in the table. Please bear this in mind when calling up the individual processes in the program. The no. of processes will reduced by one.

**String kette**
The principle is similar to that of the string sequences, the difference is that a string chain contains a pre-defined character combination. The individual character chains are placed in the memory in a continuous string, this means that the string cannot be changed - only accessed. Eg. a string chain for **STRNBOX** or logos. To enter the actual value you can again use the keycode `<C> RETURN>`. The Source-Code reads the string chains with the command **TEXT**. Using the menu option **option** it is possible to add or delete individual data elements from the string, to see how this is done - please read **String row**.

**Icons (pictograms)**
Icon tables, otherwise known as Pictograms, are the graphics contained within a program that help the user to operate the program - if you click on an icon something should happen if Eg. the toolbox in geoPaint. If you want to integrate icons in your GeoCom
programs, you will have to create them using geoPaint. The icons are placed as bitmaps in the object document using the command Bitmap. Later in the Source-Code you must tie in the objects (command : OBJECT...), first the bitmap and then the corresponding icon table. First you must enter the total number of icons. The entry Mouse position refers back to the command ICON, if this is called up the mouse pointer will be placed on the respective position. When using GEOS 128 you have an additional area : DOUBLE. Using this command it is possible to double the "X" value in the 80 character modulus, the mouse position is here only allowed to have a value of 0 to 319. If you click on the DOUBLE area it will appear inverted and an additional dialog box will appear, here you can enter the value : 0, 1 or 2. This value refers to the value for the command DBL.

```
Number of icons: 1
Mouse position: X: 0 [Double] Y: 0
```

Now follows the entry for the respective icon. Firstly the no. of the current icon will be set, followed by the name of the bitmap graphic that is due to appear as an icon. Please don't forget to tie in the graphic beforehand (and in the Source-Code). The values for the icon screen display are given in card format, for the 40 character screen the values 0 to 39 and for the 80 character screen 0 to 79 are allowed. When using Geos 128 it is again possible to use the command Double, if you use the command Double then you are restricted to the 40 character values. Because you are here working in tile format it is not possible to activate the byte movement modulus, (0,1,2).

In the next area you enter the values for Width and Height possible to activate the command Double. Last but not least the Labelname will be given, to which the program should jump to when the icon is "clicked." Using the menu option option it is possible to erase the current icon, all the following icons in the Icon-Table move up one position. The total number of icons will be reduced by one.

```
Current icon: 0
Bitmap: unnamed
Icon position: X: 0 [Double] Y: 0
Icon size: X: 1 [Double] Y: 1
Icon routine: unnamed
```

**menu Bars**

There is one restriction when using Menu-Options - per Menu. the Menu-Bar is positioned using the coordinates x1 to y2, that is, the four corners are set. Take especial care when setting up the Y coordinates, here is a tip for setting up the y2 coordinate:

```
15 + (No. of menu items * 14) = y2.
```

In the area Short-cuts you can enter, if the use of Key-Codes with the pre-fix "C=" (Commodore Key) from the menu level is allowed, the gap between the Left Screen Border and the start of the Key-Code, Eg. "---C=A. In the area Display you must select between a Horizontal or Vertical menu. i.e. Main.Menu or Sub-Menu! In the area Limitation you can select whether the menu border should be respected or not, there are two options : The option Unlimited allows the mouse pointer to leave the menu area (without making a choice) and roam freely around the screen, the option Limited limits the mouse pointer to the menu area, it can also move vertically or to the previous menu. Below this the no. of menu points must be entered.

```
Positions: X1: 0 [ ] Y1: 0 [ ]
X2: 127 [ ] Y2: 127 [ ]
Display: Horizontal [ ] Limitation: not limited [ ]
Number menu items: 1
```

Underneath the thick line on the screen (which is only there because it looks good !) the individual menu items are shown. Firstly the Active Menu Item can be adjusted, followed by Text of menu item where you can select between three options:

- Text - Normal text will be shown (naturally in BSW), this text must be entered in inverted commas , Eg. "Text".

- PAGE 16 -
GeoCom V 1.5 - Handbook

- **Text With Key-Code** - Normal text will be shown (naturally in BSW), this text must be entered in inverted commas, as in any other menu, additionally you must enter behind the text the character that is to serve as a Key-Code. Eg. "quit",Q ; You will then see in the menu: quit C=Q.

- **Name** - You can also enter the name of labels (Eg. String-Sequences or String-Chains). These are entered without "commas", additionally the position of the Character-Chain in the String-Sequence/Chain must be given. Don't forget to include the Zero-Byte in the calculation (Character-Chain + Zero-Byte). Eg. In a String-Sequence with a length of 16 the second Character-Chain will start at 17. Eg. test,17 will show the second element of test. You could utilize this trick to show all the DA's on the current disk, under the menu option geos.

<table>
<thead>
<tr>
<th>Current menu item:</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text of menu item:</td>
<td>unnamed</td>
</tr>
<tr>
<td>Type of action:</td>
<td>subroutine</td>
</tr>
<tr>
<td>Call up routine/menu:</td>
<td>unnamed</td>
</tr>
</tbody>
</table>

Finally can be seen the options: **Menu Type** and **Menu/Routine**. Under Menu type it is possible to select one of three options:

- **Menu Routine** - activates, after clicking, a routine (label) that you have entered in the area Menu/Routine. This makes it possible within the Source-Code to activate pre-defined screen masks. This label must end with the command RETMNU, followed by the name of the next sub-menu.

- **Sub-Menu** - In the following area you should enter the name of the sub-menu that you want to call up.

- **Sub-Routine** - In the following area you should enter the name of the label that is you want to jump to. (Typical switch out of a menu), the label must contain, at the beginning, the command FIRSTMNU so that the main menu can be restored.

At first glance the development of menus seems to be extremely complicated, however only through the use of the here described structuring in combination with OBJECT EDIT is it possible to guarantee the high level of flexibility that geoCom provides. We advise you to make a sketch of how the menu should look and combine this with a "scrap" flowchart so that you can see the various routings, this should make the job of making your own menus somewhat easier. Make a note of the individual labels, take care that after moving into a sub routine/menu that you have provided for a return to the main menu, you will find the commands for this function in the Handbook - otherwise: ZONK!! Also take care that you insert the menus into the Source-Code in reverse order, that is from lowest sub-menu to main-menu, then they will be tied in correctly. Ensure that you enter the correct x1 - y2 parameters, otherwise the menu will be drawn wrongly. It is possible under the option **options** to insert or erase a menu item from the current position. The following menu items will be moved accordingly.

**Dialog Boxes**
Although for a normal program dialog-boxes are not usually required, provision has been made for you to develop your own. Dialog-Boxes are a mixture of ICONS and TEXT. I.e. text is shown (with an "offer" or option) and an icon (icons) is/are provided giving you an area to "click" on. No doubt you will have seen lots of personalized dialog-boxes in other GEOS programs. It is much better to develop dialog-boxes from within the Source-Code, using ICONS. At this point we want to discuss dialog-boxes that can be developed with the command "Dialogbox-Orders" (Eg. CREATEBOX), but with extraneous icons. Therefore you must "make" the dialog-box with the aid of graphic tools.

In the Dialog-Box-Section of OBJECT EDIT you must first decide, whether you want to use a standard dialog-box-size or a custom made box. If you click on STANDARD a standard (ready use) box will be provided, otherwise you must enter the parameters for each of the four corners -> x1, y1, x2, y2. (The standard parameters are x1=64, x2=255, y1=32 and y2=127). Additionally you will have to enter the pattern for the frame, (click on the...
GeoCom V 1.5 - Handbook -

option to move to the next pattern). Next, you must declare how many elements are
go to appear in the box and then set up each individual element. The following types
of ICON Object-Document can be seen in the table, keep an eye on the parameter dbstat, it enables you to
declares within the Source-Code how you leave the box and where you go to (within the program)!!

<table>
<thead>
<tr>
<th>Type: ICON -&gt;</th>
<th>OK</th>
<th>CANCEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dbstat = 1</td>
<td>dbstat = 2</td>
</tr>
<tr>
<td>YES</td>
<td>dbstat = 3</td>
<td>NO</td>
</tr>
<tr>
<td>OPEN</td>
<td>dbstat = 5</td>
<td>DISK</td>
</tr>
</tbody>
</table>

You must enter, for the current ICON, the relative position for the upper left corner. The
X-Parameter should be given in tiles (/8) and the Y-Parameter in points. When using GEOS
128 it is possible to use the command DOUBLE, this has exactly the same function as in
ICONS, the drift can be allowed for with an additional parameter as in DBL integer,byte. Enter the label textstring - that is to be jumped to when the icon is
activated.

Further Types: ShowText

It is possible, within a dialog-box to display text. Here you must enter the relative
position, inside the dialog-box, of the character-chain to the upper left hand corner (both
parameters in points), you should then enter the text (in inverted commas (""), directional
caracters (-> / @) are possible, or the name of a character-chain if it should appear here).
The standard parameters for a "normal" dialog-box are: x=16 and y=16, 32, 48 or 64.

Further Types: MouseClick

When you select this type, you can decide whether a dialog-box can be left when the
area outside of the icon is clicked (Eg. the "infamous" OK). When using this option, the
variable dbstat has, after the function, the parameter 14.

Further Types: MouseRoutine

When you select this type, a sub-routine will be called if you "click" outside the
pre-defined icon - this is equivalent to the command ON 1 GOTO ..., you must declare
the label to which the sub-menu in the Source-Code responds.

Further Types: UserIcon

This option enables you to show personalized icons in the dialog-box. Basically this option
is parallel to the standard-icons, as far as the positioning is concerned. Simply enter the
name of the icon-table that contains your icon, this should contain only one icon, while the
option only recognizes the first icon it finds in the table. Within the icon-table the
parameter (ICON) declares the BITMAP that is in fact your icon.
One-step, Two-step : Develop your icon with geoPaint, paste this as a BITMAP in your
Object-Document, put together an Icon-Table which refers back to the Bitmap, finally you
can order a Dialog-Box to show your personalized icon!!
This sequence of events is the same when tying Objects into the Source-Code, firstly tie
the BITMAP'S in, followed by the Icon-Tables and then the Dialog-Boxes (If these include
personalized icons), finally define a routine (label) that is to be jumped to when you click
on your personalized icon.dbstat has here the parameter 13.

Further Types: Routine

Here you can declare which routine (LABEL) should be jumped to after a dialog-box has
been shown, this enables you, for example, to call up an additional Bitmap or to branch
off to a further program section.

Ok, well that's basically it as far as Dialog-Boxes and OBJECT EDIT are concerned.
Before you try to write a mammoth application, we strongly recommend you read the
section: Program Examples. You should first develop a trial concept with a few
dialog-boxes and menu-bars, for the start have a look at --> STRNBOX "Future
Developments"... .
There are a few demo programs in the circuit. You can for examples look at the
Object-Document "ICON EDIT_obj", this document shows you very nicely how a large
selection of objects works.
1.6 ICON EDIT

GeoCom saves all new programs with the same icon. Naturally you will want to decorate your own work with a personalized icon, that's why we have written Icon Edit. Icon Edit has been written with geoCom - to show how good the program is!! You can mess around with the Source-Code from Icon Edit. But how does Icon Edit work?

ICON EDIT is started with the usual double click on the program icon, the Icon Edit screen will be set up and an file selection box will appear. In this box you will see a list of all the files on the current disk. If there are more than 6 files on the disk, you can use the arrows to move within the list. Select the file whose icon you wish to change, and click on OPEN. It is possible using the options DISK and DRIVE to swap and change within your configuration.

The left hand side of the Screen:

<table>
<thead>
<tr>
<th>File Infos:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: GEO-WRITE 2.0</td>
</tr>
<tr>
<td>Class: geo-write v2.1</td>
</tr>
<tr>
<td>Author: Tony Requist</td>
</tr>
<tr>
<td>File type: Application (USR)</td>
</tr>
<tr>
<td>Applic.: only in documents</td>
</tr>
<tr>
<td>Date: 27.2.89 19:16</td>
</tr>
<tr>
<td>Size: 36 KB(bye(s)</td>
</tr>
<tr>
<td>Structure: VLIR</td>
</tr>
<tr>
<td>Start time: only 88 column</td>
</tr>
</tbody>
</table>

Here is shown all the information that can be otherwise found in the file Info.Box. Additionally you can see the file-structure (whether sequential or VLIR) and the screen module. The information is only displayed, it is not possible here to make any changes. Above this area can be seen, as always, the menu bar.

geos
- IconEdit Info: Here can be seen the usual Author Info.Box.
- Beneath can be seen the list of the current DA's.

File
- Close: The currently active Info.Box (Icon and Info.Text) will be updated and closed. The file selection box will appear and you can make a further choice.
- Recover: If you made a change that you are not happy with - you can return back to the stage of the last update, the current icon will be re-loaded.
- Quit: The current icon (+ Info.Text and Write-Protection) will be updated (disk.), IconEdit will be closed and the DeskTop will be reactivated.

Edit
This option enables you to work with PhotoScraps. i.e. you can alter or develop a new icon using geoPaint. you can copy, paste or erase scraps as in geoPaint.

The right hand side of the Screen:
The Pictograms in the Editor cause the following effects. (Please bear in mind that an UNDO command is not available!)

- Editor: From left to right:
  - Icon, mirror horizontally
  - Icon, one pixel UP
  - Invert pixel
  - Icon, one pixel to left
  - Erase all pixels
  - Icon, one pixel to right
  - Icon, mirror vertically
  - Icon, one pixel DOWN
  - Icon, 45 degree turn

Adjacent to this can be seen, to the right an enlarged view of the current icon and to the left the current icon in original size. Below the icon can be seen whether the "Write-Protection" is activated and the edit-area x & y coordinates:
- GeoCom V1.5 - Examples -

It is possible to switch the status of the "Write-Protection" by clicking on the button. If you move the mouse pointer into the editing area you will find yourself automatically in the Edit-Modus, the pointer will become a small rectangle and and the current x/y coordinate will be shown. In order to change the icon you must enter the "Paint-Erase" Modus, this works just like geoPaint in the Zoom-Edit-Modus- changes in the edit-area will be shown immediately on the original icon.

The Info. Window
In the Info.Window can be seen the current Info.Text, this can be edited, changes in the Info.Window will be registered by all 3 save commands (Close, Update and Quit).

The Iconpuffer

```
Iconpuffer: [ ] [ ] [ ] [ ] [ ]
```

The Iconpuffer is a RAM buffer, during the period when IconEditor is active it is possible to store a number of ready-use icons in the buffer. Click on the respective icon area and one of the cursor arrows - The LH arrow swaps the buffer icon with that of the edit area, the center arrow copies the current icon into the icon buffer and the RH arrow copies the buffer icon into the edit area.

On the Disks can be found, apart from the completed programs, the individual files for GeoCom - these are:

- ICON EDIT_com = the Source-Code, here you can cut out sections for your own use.
- ICON EDIT_obj = the Objekt-Document and
- ICON EDIT_asm = a Machine-Code section for MegaAssembler (not translated)
- ICON EDIT_msc = the translated Machine-Code section, must be tied in with Linker

1.7 BASIC TO GEOS
This program has been included to enable you to import BASIC (C64 V2.0, C128 V7.0) programs into geoWrite. The program does not help you with the construction of the Source-Code, it only gives a geoWrite-doc. version of the BASIC listing. The line numbers and BASIC syntax remain - no changes are made to the BASIC program, the text cannot be imported straight into geoCom.

Before the BASIC text can imported into GeoCom a lot of work has done to done:

- Remove the line no's and issue all the jumps (GOTO GOSUB etc.) with labels.
- Replace the BASIC-Commands with geoCom-Commands.
- Suit the various operations/sequences (IF/THEN) to the geoCom commando structure.
- Convert the Parameters and loops to the geoCom structure.
- Convert the Disk. and Printer commands to the geoCom structure.
- All System-Commands must be adapted to the GEOS structure (POKE, PEEK and SYS)

As you see there is a lot of work to be carried out before the BASIC-geoWrite-Text can be translated with geoCom, therefore it can be appreciated that this form of conversion from BASIC to GEOS is only really suitable for a limited number of programs. The start of BASIC TO GEOS is the same as any other (normal) GEOS program, double click on the program icon, after a short period the introduction screen will appear. The first screen offers you a Dialog-Box whether you can choose between the various (o Disk.) files, only the BASIC files are shown. Select the BASIC file that you want to import into geoWrite, a further Dialog-Box will appear asking you to enter the name of the new geoWrite docu, in the next box you are required to enter which geoWrite version the text should adopt.

```
Please select text format:
[ ] geoWrite v1.1
[ ] geoWrite v2.8
[ ] geoWrite v2.3
[ ] OK
[ ] Abbruch
```

The selection is made here by clicking on one of the Format-Options, the current format button is shaded. After you have made your selection you must decide whether the import should be shown on screen or if it should take place in memory, if you select the screen
option the current BASIC import line will be shown - otherwise the import would be to slow. (It is much faster if you don't select the screen display !)
Top right can be seen the no. of characters in the import and below this the geoWrite page no. When the import has been carried out a "success-story-message" will appear, after your acceptance (OK) you will return to the start box. The import will be aborted if the maximum no. of pages is exceeded (60), once you have returned to the beginning you can either select another BASIC program or return to the DeskTop.

You will not be able to find a Source-Code for this program
*** it is NOT PD ***

We here with call upon all programmers to help us develop a program that can import (and convert) GEOBASIC programs and GEOBASIC Source-Codes into the geoWrite format.

1.8 LINKER
Using this program it is possible to hang sequential files onto the end of a VLIR file as a new module, the new module then becomes the last module in the program and can contain Machine-Code routines that can be ordered to be read by your (made with geoCom) program as a supplementary to the geoCom routines. An example of this function is ICON EDIT.
This application is started in the usual manner with a double click on the program icon, a dialog box appears where are you are required to select the VLIR file onto which the supplementary SEQ file is to be hanged onto. Using the arrows it is possible to move through the file list until the required file is found. If you click on CANCEL you will return to the DeskTop. After you have made your selection, the program will be continued, and a further dialog box (for the selection of the .SEQ file) will appear, here the CANCEL option returns you to the first dialog box. The .SEQ files are usually either in the MegaAssembler or geoProgrammer format.
Linker now attempts to hang the .SEQ file of the end of the VLIR program. If a disk error should occur - this will be shown on-screen - and Linker will return to the first dialog box. Otherwise a "success-story-message" will appear and Linker will tell you which module number the new (ex. SEQ) module has been allocated, finally Linker returns to the first dialog box. Once you have returned to the beginning of Linker it is possible either to select a new VLIR program (and .SEQ file) or EXIT to DeskTop with the Cancel option.
Apart from the complete LINKER program you will also find a number of ancillary files for geoCom:
- LINKER_com = the Source-Code
- LINKER_obj = the relative Object-Document
- LINKER_asm = a Machine-Code section for MegaAssembler (not translated)
- LINKER_msc = the translated Machine-Code section, can be used with LINKER.

1.9 Program Examples
Before you attempt to develop your own programs we strongly recommend you to read carefully chapters 1 & 2 and this section. You are also recommended to print out all the example Source-Codes Eg. IconEdit, geo3D ... and to attach theses to the Handbook as an appendix.
All of the programs that are included in the geoCom package are complete, run-able geo programs. By studying these programs you can get an insight into Source- structureing, Command and Source-Code-Text construction. You can experiment around with these programs, change lines and/or cut sections out for your own programs. the only thing that you may not do is to try and present these programs as your own work (even as PD), have a good look at the "bracket-technik" in geoCom - to avoid making to many mistakes.

1.9.1 "Pattern Show"
By using this Source-Code as an example we would like to illustrate the basic concept behind geoCom. We will not be looking at each individual command, because of this we recommend you to take a good look at the workshops - print out all the Source-Codes (PATTERN(a)_com to PATTERNSHOW_com) and add these to your Handbook as an appendix. Place the files geoCom,Start Editor,Start geoCom, geoWrite and the demo file listings on a work disk.
- GeoCom V1.5 - Examples -

**Let's give the whole thing a frame - PATTERN(a)_com**

The first priority is to make a new geoWrite document : PATTERN(a)_com, this will be the docu. for the Source-Code. Next we need a skeleton : Declarations - Definitions - and Instructions Sections. In the Definitions-Section are found the file name, it's class and the author's name, (for Geos 126 the Startflag $00 will have to be included - switches to 40 character modus). All the above items are required by geoCom during the Compilation.

**TIP :** The leading-comma ' is used by geoCom as the BASIC command REM - everything that is included in a line after the command ' will be ignored by geoCom !! Under the command **AUTHORIZED** it is possible to enter your own name !

In the Definitions-Section can be found the Variable **COUNTER** and the LABEL **end**. Always ensure that you have entered the correct variable-type, geoCom requires this information so that it can define the variables.. The Instructions-Section must include the command **CLS** (Clear Screen), the LABEL **end** , the program is exited and returns to the DeskTop. As you can see it is possible to include your own messages and notes within a geoCom Source-Code, this has later no effect on the length of the completed program (unlike BASIC)- geoCom ignores the messages during the Compilation.

**Let's do an experiment - PATTERN(b)_com**

After we have cleared the screen we want to make an object appear on the screen - a logo. We require as structure a PATTERN 0, a rectangle **RECT 120,160,319,199** and a frame for the rectangle **FRAME 122,162,317,197**. In order to display our construction we need the **PRINT** command and the **SETPOS**- command to move the rectangle away from the left hand border. The principle of a "window" is as follows :

- Draw the rectangle - with **RECTx1,y1,x2,y2**
- Draw the frame - with **FRAMEx1,y1,x2,y2**
- Set the Cursor with - **SETPOSx,y**
- Set the Text-Style (thick) with - /B
- or normal with - /P

in order that the mouse can be utilized by a further routine(s) we now "switch" the mouse on, the command is : **MOUSEON**. If we were to use the command **MOUSEON** without a further parameter the program would immediately return to the DeskTop, therefore we insert the Label **click**. Using the command **ON 1 GOTO** means that after pressing the "click" button the program will go to a Label, the Label **ON byte GOTO ...** has a number of further possibilities. These five options enable you to cover the majority of queries and tie-ins in the Mainloop.

What or who is the **Mainloop** :

One of the great advantages with GEOS is the mouse pointer, when you move you mouse - the mouse pointer on the screen moves parallel to your hand movement, when you press the button "click" something happens (usually !). But how does this work, how does the computer know that you have moved the mouse ? The secret lies within the Mainloop, the Mainloop is a GEOS System-Interrupt that lies upon the normal System-Interrupt - it has been expanded to include the observation of the mouse (movement and click) and the keyboard. Therefore you do not need to insert these routines in your programs as they are automatically activated by GEOS. You only need to tell your program that the mouse routine is active and where to access it (ON byte GOTO ...), this is covered in our example by the labels **click** and **keyout**. The Pictogramms and Menu-Bars that you have defined with ObjectEdit will also be tied into the Mainloop. The Mainloop access can be seen so

1. Start the program, leave the Mainloop, the program will be loaded.
2. The program will run (initialising) to the command **MAINLOOP**
3. (Key and mouse access points could be built in (ON 0 GOTO click)).
4. Return to the Mainloop with the command **MAINLOOP**.
5. You will remain in the Mainloop (the mouse and keyboard are being constantly accessed) Eg. You press a key.
6. The program then jumps to the label **click** and looks to see whether the activated key has given a specific function, if YES the function will be carried through, if NO the Mainloop will be re-activated.

You will see that the points 5 and 6 represent a constantly repeating circulation, but now back to our example : After the label **end** will be found the label **click** but in between we will insert the command **GOTO end**. The running order of the program will now be:
- GeoCom V1.5 - Examples -

- Clear screen
- The window will appear in the lower part of the screen
- The Mainloop will be activated and when you press the mouse-button the program will jump to the label **click** and then the program will jump to the label **end** and return to DeskTop.

**Our computer has got a lot of keys . . . - PATTERN(c) . com**

so it would be nice if we could use them. The jump from the Mainloop for this function is ON 0 GOTO label. We are going to call our new label **keyrout** and we must define this label beforehand. Within the label **keyrout** a variable will be controlled which will show the key(s) that were pressed last --> keydata. The variable control can be best accessed with the :

**IF (keydata == parameter) GOTO label**

loop. If you additionally press the C= (Commodore) key then the parameter will be increased by 128. The normal Key-Code for quitting C=Q has the parameter 113 + 128 (q + C=) = 241 (Hex $f1), it makes no difference whether you insert keydata == 241 or keydata == $ . We strongly recommend you to stick with the well known standard GEOS Key-Codes, here are a few examples :

C= Q - QUIT
C= O - OPEN
C= P - PRINT
C= I - CALL INFO.BOX
C= S - SAVE

In our demo, we now go to the label **end** when the Key-Code C= Q is accessed, otherwise the parameter RETURN in the label **keyrout** will be activated and the program will return to the Mainloop. It doesn't matter when you access the Key-Code - Mainloop is always watching you !!

**Fine, but we want to see a Pattern - PATTERN(d) . com**

Now comes the most important part - displaying the Fill-Pattern. GEOS has 34 standard patterns, possible is an area 320 Pixels (0-319) (horizontal) by 200 Pixels (0-199) (vertical). We have divided the Pixel-Area into 8 x 4 fields: that is 8 fields horizontally in four rows. This means that there are, mathematically, 320 : 8 = 40 Pixels per pattern. Using this formula the first Pattern-Field has the Pixels 0-39, the second 40-79 and so on, vertically it's the same game: 0-39 Pixels. You can experiment with other parameters - be careful - don't forget the maximum parameters ! So that the Pixel-Fields can be called up at will, their definition must be carried out with a label, this label must be inserted before the Mainloop is accessed - therefore we must jump to the pattern just after the logo has been displayed. GOSUB **showpatt**. The display of the pattern will be achieved within the sub-menu **showpatt** by a REPEAT-UNTIL-Loop. The display loop between **REPEAT** and **UNTIL** is an abbreviation of an otherwise very complicated label. You can of course insert the display in clear text :

```
    PATTERN 0
    RECT0,0,39,39
    FRAME 0,0,39,39
    PATTERN1
    RECT40,0,79,39
    FRAME40,0,79,39
    PATTERN2
    RECT 80,0,119,39...
```

etc. etc. This would leave you with two problems, one you would use to much Code-Area and secondly you would use a great deal of Constant-area - a loop is much better.

**Well if we've got a mouse, we might as well use it - PATTERN(e) . com**

The mouse is accessed with the label **click**. The interrogation is carried out with the REGION-command and the Parameter-Access-Format: IF (REGION . . . . . . . The Logo will also be utilized to jump to the label **end**. The commands IF (mousedata == 0) and INTERRUPT OFF are important for the mouse interrogation. The Mainloop registers the Mouse-Key twice during a "click" - the first time when you press the key and the second time when you let the key go. This means that the relevant loop will be run twice if the the mousedata is not accessed. INTERRUPT ON and INTERRUPT OFF ensure that the
interrogation doesn't happen in the wrong area, the interrupt will be short term stopped, the mouse access removed from the Mainloop and so an eventual movement of the mouse during the "click" action will be ignored. - Don't forget to re-activate the interrupt. As an example remove the following line in the label click:

IF (mousedata == 0) THEN . . .

and the last ENDIF (before RETURN), start geoCom and compile the program. Start the completed program and watch the difference !!

In our program we have inserted a Bytevariable (endflag) that records a successful interrogation. If you click anywhere, the X and Y Parameters are saved, the endflag is placed at 255 and a calculation is carried out to see whether X and Y are within the allowed parameters. Take care that a comparison under geoCom always - = is !! A Dialog-Box (here the STRNBOX) will be shown with the Fill-Pattern parameter.

And we've just got to have an Info.-Block ... PATTERN(f).com
Well we all like to wave our own flag, and computer programmers are no exception. In GEOs programs there is usually an indication of the author in the Info.-Block this is very easy in geoCom and this is what we are going to do here. For this function we will need to use the command STRNBOX"text","text","text". here we can enter 3 lines of Text and let them be shown - the OK icon is already integrated if you "click" on the OK icon the program will return to the line after the jump (like GOSUB in BASIC). In the Key-Code access (label KEY) the interrogation to Key-Code C= i (Call Info) will have to be integrated. we come back to the Mainloop with the GOTO-RETURN-Loop, and the label info must also be called up. Naturally you don't need to insert the Text that is due to be shown in the STRNBOX beforehand, as in our example. You can pre-define the Text, like most of the other Dialog-Boxes, either as a string or simply insert it:

STRNBOX "TPattern Show version 1.0","IP/B for GEOS 64/128, 40Chars."
(w)1993 Denis Doepler

You can use the Text-style functions to emphasize the text.

Let's make a Hardcopy of the screen ... PATTERN(g).com

The most important thing is here the correct Key-Code, for printing this is usually C=P.

For graphics we will need the Variable-Area ROW 1920 BYTEVAR buffer and ROW 640 BYTEVAR buffer2, the first section is for the Printer-Driver the second section records the parameters for a Graphic-Line (80 x 80 points high). You can select your own names (buffer and buffer2) ours are just meant as an example. Now we go onto the label print:

To start with we open a Dialog-Box and prepare the actual printing with PRINTINIT, additionally the standard error-check must be inserted (the required Printer-Driver may not be present and GEOs would (without the error-check) crash !) The error-check iostat must be included by all attempts to access a drive or the printer ! Our example:

IF (iostat == 0) THEN
STARTPRINT . . .
ELSE
ERROR
ENDIF

The relevant File no's. can be found in Chapter-3 of the Handbook, analog to the Error-No's and your experience it is possible by a relevant Error-Code to switch to a respective sub-routine. If no error appears the printer will print, otherwise the Dialog-Box ERROR will appear. Basically there are two forms of printing:
- Normal text printing with LPRINT
- Memory-Print, a Memory-Area will be printed.

The Text-Print is similar to the Draft-Print-Mode in geoWrite, the characters are sent to the printer and these are set on paper using the printer internal codes (Personal set-up !!). Of course you can also send Printer-Specific-Commands, these must have the pre-fix "/" and should be recognized by the printer, using this trick it is possible to set up your printer from within the program.

The Memory-Area print command (and the screen print) already includes the Graphic-Print-Commands.
Odds and Ends . . . PATTERN(h)_com and PATTERN-DEMO_com

Normally our program has now been completed, but we would like to show you two extras that you quickly fit in. The first example is the Clock-Display, if the clock should be visible then it should be included in the Mainloop - SETVECTOR $849b . . It is possible to operate a screen saver without having first to install Ram-Process. But first the Clock-Display, the current time can be read from the System-Variable date<3> to <5>. The display format can be seen clearly in the Source-Code, the following line is very interesting:

RECT xpos,185,316,196

The right hand portion after the Clock-Display has been erased, you have probably seen that in TopDesk some dots remain behind the seconds display, this is because some digits are thinner than others (1). We have avoided this problem by catching the Text-Cursor after it has written the seconds digit and defining its position (xpos), then the rest of the line up to the right hand side is erased. The Clock-Display and the Screen-Saver are built into the Mainloop, thus enabling the clock to be shown independent of the current program.

The Screen-Saver is nothing more than a digit-loop with a few parameters it has so been constructed that two operations must be "true" in order to switch the screen on or off.

The first part of the Screen-Saver starts with:

IF (((PEEK $39) <=0) . .

First, the mouse is interrogated - are you moving (PEEK $39) or is the NOW mouse position (mousex/mousey) other than the memory position (oldmousex/oldmousey)? If one of the parameters are true, and only then, the program asks is the screen OFF (screenoff = 255). If the screen is still OFF then the program will switch to the old Operations-System, the screen will be activated (ON) and the variable screenoff will be set to 0. If the screen is not switched off, the digit-loop will be returned to the whole parameter (300) and return to the Mainloop. The section after ELSE will be ignored.

If the mouse is not being moved, the program moves straight on to the section after ELSE. Is the screen ON (first requirement), the counter (counter2) will be reduced by one. If the counter is now 0 (second requirement) then the screen will be switched OFF. This is registered in the parameter screenoff (=255) and the parameter from the present mouse position oldmousex/oldmousey registered. If the parameter counter2 not 0, then the present mouse position will be registered.

So as you see, very simple - but it works ! You can alter the "Switch-Time" by adjusting the counter2 in the line:

counter2 = 300

If you adjust the value to 1000, then the "Switch-Time" will take 3x longer, if you adjust the value to 100 then the "Switch-Time" will be accelerated by a factor of 3.

A possibility would be to allow the user to adjust the "Switch-Time" at the beginning of the program with an Input-Dialog-Box, INPBOX.

Now we will move on to the other Demo. programs in the geoCom software packet.

1.9.2 SHOW FONT V1.4

SHOW FONT displays all the fonts on the current disk, together with all the possible styles and in all the point sizes that can be used in geoWrite. Using the program is very simple: Copy SHOW FONT and at least one font to a work disk, and start SHOW FONT with the usual double click on the program icon. The program will start and show you a file selection box where you can select the font you wish to see. The font will be loaded and displayed in the smallest possible point size, on the right hand side you can see a selector for the various font styles. If you click on the style selector, or enter one of the geoWrite style Key-Codes, then the font will be displayed in the respective font style. Using the arrows, below right, (or the cursor keys) it is possible to change the size of the font - only as far as the font allows. Please bear in mind that SHOW FONT can only show font sizes that can be used by geoWrite, SHOW FONT can display LQ- and LW- fonts, SHOW FONT cannot display the Mega-Fonts from geoPublish.

You can exit SHOW FONT with the Key-Code C= Q. SHOW FONT shows a method of programming under GEOS without using a lot of menu
bars, make a printout of the Source-Code and place this in your geoCom Handbook as an appendix, this enables you to use the text as a help when looking for your own mistakes!

1.9.3 SYSTEM-INFO
SYSTEM-INFO is an application that shows you a lot of information from the Geos-Memory - Eg. Series No. and Screen Modus. You can have a look at the Source-Code to see how to access the various system areas, you can use the various sections for your own programs - the majority of programmers will never use all the routines but it’s nice to have them for the day when ...

SYSTEM-INFO is started with the usual double click on the program icon, the program will start - the current Fill-Pattern will be accessed and shown, a logo will appear and in the top right hand corner the System info’s will be shown. All the (four possible) current drives will be displayed, the drive type and their current disks. Of course it takes a few seconds to access all this information, the mouse pointer below right is a graphic and cannot be moved - the program can only be exited with the Key-Code C= Q.

WARNING : You must place formattted disks, in the system drives before starting SYSTEM-INFO !
Make a printout of the Source-Code and place this in your geoCom Handbook as an appendix, this enables you to use the text as a help when looking for your own mistakes!

1.9.4 Geo-3D
Using this program it is possible, for the first time in Geos 64/128, to display a 3-dimensional object on the screen. GeoCom contains the relevant mathematical commands which make this function accessible. The parameters that can be accessed are:

The following Key-Codes can be accessed :
- C= Q Quit
- C= D Display
- C= I Info.Box
- C= H Simulated help screen.
- C< Switch between 40/80 character screen (only Geos 128)
- C= P Enter Points
- C= V Enter connecting lines
- CRSR-Keys UP-DOWN = Text-Cursor control
- RETURN = Affirm Text Input

Using the mouse it is possible to directly access the various menu options. After entering a parameter you must click on "Show", your mistakes will not be corrected by the program - therefore if you make a really good error then the whole system could crash:

ZONK ! !

The principle behind Geo-3D is really very simple, to begin with you a point, the parameters x0 and y0 are directly tied into this point. You now enter new points that move further away from the first point – in the X, Y, and Z directions, as “Distance” a parameter of 1 mm is advised, you can increase the difference with scale option.

Recommended parameters for the scale option are : 0, 0.5, 1, and 2.
Finally enter which points should be connected to one another, there is no requirement for all the points to be connected, have a look at the example parameters!
Geo-3D is meant as a think-tank for all Geos programmers ! We hope that we have given you a new idea !
The Source-Code includes a method that shows you how to run the mouse and keyboard parallel. Make a printout of the Source-Code and place this in your geoCom Handbook as an appendix, this enables you to use the text as a help when looking for your own mistakes!

1.9.5 SID-Demo’s
The SID-Demo’s show a programming possibility for creating sounds by accessing the SID. Important:
There are no special commands to create noises, music or sounds the program accesses the original Commodore internal system using the well known POKE and PEEK commands.
You must inform geoCom with the commands INITIO and DONIO that you are accessing the Commodore system.
The interrupt is stopped during the access, the mouse and the keyboard cannot be
GeoCom V1.5 - Examples -

accessed (WAIT). The accessing of the SID is just like in BASIC V2.0 or V7.0, and is also displayed by PATTERN SHOW in the Screen-Saver. This is really only a function for programming professionals!

Make a printout of the Source-Code and place this in your geoCom Handbook as an appendix, this enables you to use the text as a help when looking for your own mistakes!

Remember the demo's are exactly that: DEMONSTRATIONS, they have not been consequently carried through to the last possible programming trick, there are still mistakes in the programs - false parameters, incorrect tables etc. We only want to show you some of the possibilities when programming with geoCom. We are planning to issue a disk, with complete demo's, if you want to help us with this project, we will be glad to hear from you - we may even pay you for your completed finished geoCom masterpieces! Our contact address:

Denis Doehler
Gorkistr. 18
04347 Leipzig
Germany
Command and functions are always written in CAPITAL LETTERS and their relative variables, Label's and object-name's are always written in lower case letters. GeoCom commands can also be used as definitions, the difference between a definition and a command is indicated by the case-syntax. Have a look at chapter 2.12 for a list of prepared variable names. A named-definition must always begin with a letter and may have a maximum length of 15 characters (lower case), after the first character you may enter letters (alphabetic) or digits (numeric), comments (in BASIC: REM) are indicated by the character "(the comments will later be ignored during the compiling process).

The following parameters can be hanged on the commands:

- **byte**
  - Variable or Constant -Parameter 0 - 255 ($00-$ff), 1 Byte
  - Eg: 199, 1, x1, page, load_3

- **integer**
  - Variable or Constant - parameter 0 - 65536 ($00-$ffff), 2 Byte
  - Eg: 319, h, y2, page, whole_1

- **real**
  - Variable or Constant, 5 Byte
  - Eg: 10.5, test, test_5

- **string**
  - Variable / Constant with Text, this Text must be in Inverted Commas, 1 Byte larger as the text
  - Eg: "text"

- **name**
  - as string, the name of a file will be awaited
  - Eg: "Data"

- **text**
  - as string, the "text-contents" will be displayed
  - Eg: "Please insert Disk !"

- **label**
  - Label name in the Source-Code (lower case)
  - Eg: start, click

- **file**
  - File-variable for every data-interrogation, Defined with FILEVAR. The command OPEN defines the respective file.
  - Eg: file

- **object**
  - Parameter for Object, Sprite, Menu-bar, Bitmap-graphics, Icon, Dialog-box etc., Defined with ObjectEdit.

- **memory address**
  - A variable address, Label name within the geoCom-Variable-Memory

- **byterow**
  - Data sequence within the Variable-Memory

- **introw**
  - Data sequence within the Variable-Memory

- **rearrow**
  - Data sequence within the Variable-Memory

- **strrow**
  - Data sequence within the Variable-Memory

- **data sequence**
  - Data sequence, important for Machine-Code

A few tips for beginner's, and pro's for "clean" programming:

1. Avoid jumping around to much within a program, if you program to many sub-routines this will destroy any advantage that you may have planned, apart from this the program will become to complicated.

2. Try to write short, clear program sections and save these for later programs in a Text Eg. Change Disk., Call Printer, Check Keys etc.

3. You can of course add comments to a program but it is easy to over do it ! You will probably find that after a few weeks the only way to understand a Source-Code is to read it from beginning to end. Your comments are ignored by the Compiler.

4. Use obvious names for variables. The variable quantity tells you a lot more than x2_suit!

5. Use obvious names for label names. You will have a much better overview when you use labels like: end, key, calcu_1 etc instead of: ter2ert, gerht9, dergft+2

6. Functions that are often repeated Eg, updating the screen (after a sub-Dialog-Box) or calculations are best represented with loops. You have a number of possibilities for building and activating loops: REPEAT . . . UNTIL, WHILE . . . LOOP and IF . . . THEN . . . ELSE.

7. Use all the advantages and shortcuts that geoWrite offers you.

8. Take advantage of the text formatting and the various text styles.

9. Take care with the various mathematical specialities ! x = (x+1), must be written in commas. An equals character "=" means assign. Two equals characters "==" is a comparison.

10. Normal increments or diminutions of ONE can be dealt with faster using the commands INC and DEC.
- GeoCom V1.5 - Commands etc -

11. If the same calculation has to be carried out a number of times, then define a LABEL for the calculation and jump to the label when the calculation is required.
12. before you even start to write the Source-Code, what should this program do - how should it get there!
13. Make a note of all the basic functions beforehand on paper:
   1. Screen display . . .  2. Key interrogations . . .  3. Branches off to where . . .
14. Numerical equations, addresses and variables that appear often should be integrated in loop calculations.
15. Variable that are often used should be defined before others that are used less often. They well be read earlier and the whole memory doesn't need to be accessed, this makes the compilation much faster.

2.01 Variable Requirements
Variables are names for memory positions. These memory positions can be filled with a particular parameter. Eg. with a number (global parameter). It is also possible to change the parameter during the course of a calculation, and then to redefine the variable with the changed parameter (local parameter). The basic knowledge: what are integers and constants, will not be described here - this is common knowledge and can be researched.
- Negative real numbers are defined in geoCom in brackets : (-2.01)
- The decimal is shown with a stop and not with comma : (2.01)
- Real numbers require 5 Bytes of memory (4 Bytes + pre-character)
- Integers require 2 Bytes of memory
- Byte numbers require 1 Byte of memory
- String variables require as many Bytes of memory as characters, plus an end-character.

2.02 Pre-select Variables
Generally a variable can have any name, including geoCom commands. To simplify matters and to avoid very complicated memory interrogations it is usual practice to define the parameter of a variable as it's name, this then means that certain variables will be pre-defined. Please don't redefine the following variable names with your own parameters in a program. In addition to the here pre-defined geoCom variables there are the up-load geoCom variables that can be seen in chapter 2.03, these variables are not often used and if required can be called up with the command "INCLUDE definitions.exe" which must (if required) be inserted in the Declarations-Section of your Source-Code - The geoWrite doc. "definitions.EXE" must be on the same disk. geoCom variables from this docu. can be, if not required, be erased - only from a copy of the docu., never from the original ! An explanations text is also provided : "definitions.exe+". The geoCom variables from chapter 2.02 are always available, the geoCom variables from chapter 2.03 are only available after they have been tied in during the compilation.

```
geoCom-Variable  Geo Name  Memory Position
backbyte         geoCom-internal
Gives the quantity of not occupied Data-Elements at the command FCLASS.
backword         geoCom-internal
Receives the address of the last-read Bytes at the command :
READfile,startaddress.size.
bootflag         firstboot   $88c5
After booting Geos the parameter of the variable is = $ff (255) otherwise $00 (0).
complflag        c128Flag   $c013
Displays the computer type : complflag = $80, it is a C128 ; $00, it is a C64.
(Using this variable it is possible to let the program switch the screen modus automatically.)
curdrive         curDrive    $8489
Receives the machine No. of the current drive as variable. No's. 8 - 11 are allowed.
date<0> to date<5> year - second $8516
Receives the parameter of the current Date/Time. The individual parameters are :
date<0>=Year  date<3>=Hour(24h)  date<2>=Day
```
- GeoCom V1.5 - Commands etc -

dbstat DBstat geoCom-internal
Receives after leaving a Dialog-Box the No. of the on-clicked icon. The following
parameters are standard, they can be changed in your self made Dialog-Box(es):
1 = OK
2 = CANCEL
3 = YES
4 = NO
5 = OPEN
6 = DISK
13 = RETURN

direnty dirEntryBuf $8400-$841d
30 character large variable with the contents of the File-Data-Block of the current file. Eg:
direnty<0> = CBM-Typeee
direnty<22> = Geo-File type
direnty<23> = Year of the last update
This will be updated when you, with the commands FINDFILE, OPEN access a file and a
read error is not shown.
doctype dataFilename geoCom-internal
Receives the name of a Data-File + 1 zero byte. Eg. You double click a geoWrite docu.,
geoWrite will be loaded first and doctype receives the name of the docu. that you
clicked so that that the docu. can be loaded after geoWrite.
drtype<8>—<11> drivetyp $848c-$8491
Drive-type, drtype<8> to drtype<11> supply a byte-variable with the following
parameters:
Bit 0 - 5 = 0 = no drive
1 = 1541, 1541 C, 1541 II
2 = 1581
Bit 6 = 1 = shadowed drive
Bit 7 = 1 = REU-drive
Bit 6 and 7 never together!

docom geoCom-internal
Receives the current Versions-No. of geoCom.
iconflag iconSelFlag $84b5
Is iconflag = 64, then an icon should be inverted after being "clicked".
Is iconflag = 128, then an icon should blink after being "clicked".
Other parameters will not be accepted.
iostat geoCom-internal
Receives during Disk. accessing in the completed program the Disk.-Error No. , this can
then be used to check errors.
Normal parameter = $00 (no error !)
fault faultData $84b6
If you have reduced the mouse movement-availability with the command mouseLeft ..
then fault will tell you whether the mouse pointer has "hit" the allowed border. The
variable fault will also be accessed when you leave the current menu. Parameters:
Bit 7 = 1, when the mouse pointer hits the upper edge of the allowed area.
Bit 6 = 1, when the mouse pointer hits the lower edge of the allowed area.
Bit 5 = 1, when the mouse pointer hits the left hand edge of the allowed area.
Bit 4 = 1, when the mouse pointer hits the right hand edge of the allowed area.
Bit 3 = 1, when the mouse pointer is no longer in the current menu.

keydata keyData $8504
Receives the ASCII parameter of the last-pressed key.

menu menuNumber $84b7
Receives the No. of the current menu, the first menu option has the parameter zero (0).

men unm geoCom-internal
Receives the No. of the menu option that has been clicked. This must be read as the first
item in the MenuRoutine.
mousedata mouseData $8505
Receives the parameter 255 ($ff), when the Fire-Button is pressed, otherwise the
parameter is zero (0).

- PAGE 30 -
mousex     mousexPos  $003a-$003b
Integer parameter, x-coordinate of the mouse position.

mousey     mouseyPos  $003c
Byte parameter, y-coordinate of the mouse position.

nation      nationally  $c010
Nation-parameter, $01 for German version, $00 for US version.

numdrives   geoCom-internal
No. of system drives (1-4).

pmod        geoCom-internal
Important for LINE or POINT. Is pmode = 0 - set dot, is pmode = 1 - erase dot.
Is pmode = 255 will copy a area from the background to the foreground. here must
scrbuf = 192!

ramflag     sysRamFig  $88c4
Is required when a REU (Bank 0) is to be accessed by the Geos-Kernal.
Bit 7 = $0000-$7ffff the MOVEData-Routine is to be used for memory access
Bit 6 = $8000-$bfff receives the drive-driver (Drive A - C)
Bit 5 = $c000-$7ff will be loaded with the area $8400-$88ff, when Geos is exited to
BASIC.
Bit 4 = $7e00-$82ff and $b900-$fc3f are required for Re-booting.

scrbuf      dispBufferOn
This is for graphic and text. The variable has the parameter:
128, only write in the foreground; 64 only write in the background; 192 write in both
fore- and background.

scrcol      screencolours  $851e
Is the parameter for the current fore- and background colour.

scrflag     geoCom-internal
Displays the current screen modus:
0 = 40 Characters, 128 = 80 Characters

stringx     stringX  $84be
Receives as an integer-No. the X-coordinate of the character-width of a character-chain.

stringy     stringY  $84c0
Receives as Byte-No. the Y-coordinate of the character-height of a character-chain.

style       geoCom-internal
Defines the text-style for text.
Bit 3 = Outline Bit 4 = italic Bit 5 = inverted
Bit 6 = Bold Bit 7 = Underline.

sysflag     c128Flag  geoCom-internal
Receives the computer type:
Parameter = 128 it is a C128, at 0 it is a C64.

sysreg      $5000     geoCom-internal
Together with CALLSYS defines a data-sequence (integer) for the system-register.

version     $c00f
Defines the Geos-versions-No.:
$12 = Version 1.2 $13 = Version 1.3 $20 = Version 2.0

2.03 Loading Variables From Disk.
After the definition and the address follows the geoCom variable name. You only need to
use the variable no. The available name-memory reduces to 100 after tying in these
definitions.
**GeoCom V1.5 - Commands etc.**

External Definitions Version 1.0 - Copyright 1993 Falk Rehwagen

*** geoCom-internal Variables ***

**INTVAR AT $0400; xpos**  
Variable: xpos  
Receives the current screen cursor position (x-parameter).

**BYTEVAR AT $0402; ypos**  
Variable: ypos  
Receives the current screen cursor position (y-parameter).

**BYTEVAR AT $0403; pattern**  
Variable: pattern  
Receives the current pattern No.

**BYTEVAR AT $0407; bamstat**  
Variable: bamstat  
Receives the current drive No. of the current BAM (0=no BAM).

**BYTEVAR AT $040a; printflag**  
Variable: printflag  
Is the Variable<>0, then the printer-driver is active, special screen modus !

**ROW 4 BYTEVAR AT $040b; numfiles**  
Variable: numfiles  
Gives the no. of open files in the current drive (insert 8 - 11)

**BYTEVAR AT $040c; docflag**  
Variable: docflag  
Is the Variable<>0, then a docu. should be opened at START; if Bit7 = 1 then print docu.

*** GEOS-internal Variables ***

**BYTEVAR AT $30; mouseflag**  
Variable: mouseflag  
This Variable has the parameter : 128 when the mouse-pointer is visible ;  
64 when die menus are available; 32 when the Icons are available.  
Combinations are possible, Eg. 96, when menus and Icons are available.

**BYTEVAR AT $39; pressflag**  
Variable: pressflag  
This Variable has the parameter : 128 when a "new" key is pressed;  
64 when the input device is moved ; 32 when the fire button is pressed.  
(Combination = sum of the individual parameters)

**BYTEVAR AT $33; win_top**  
Variable: win_top  
is the parameter for the upper edge of a Text-Box.

**BYTEVAR AT $34; win_bottom**  
Variable: win_bottom  
is the parameter for the lower edge of a Text-Box.

**INTVAR AT $35; win_left**  
Variable: win_left  
is the parameter for the left hand edge of a Text-Box.

**INTVAR AT $37; win_right**  
Variable: win_right  
is the parameter for the right hand edge of a Text-Box.

**BYTEVAR AT $84b3; selection**  
Variable: selection  
Blink speed for menus and icons  
Standard : 10

**BYTEVAR AT $84b4; alphaflag**  
Variable: alphaflag  
Says whether the text cursor visible is, this then influences the cursor blink-speed :  
Bit7=1 Cursor is blinking; Bit6=1 Cursor is now on; Bit0-5 Blink-speed

**BYTEVAR AT $84b6; faultdata**  
Variable: faultdata  
Says whether the mouse-pointer has "hit" the pre-installed border, or if the mouse-pointer has left the current menu, Parameters :  
Bit7=1 top, Bit6=1 bottom, Bit5=1 left, Bit4=1 right, Bit3=1 outside

**BYTEVAR AT $84b6; mouse_top**  
Variable: mouse_top  
Defines in a mouse-window, the upper border.
- GeoCom V1.5 - Commands etc -

BYTEVAR AT $84b9; mouse_bottom Variable: mouse_bottom
Defines in a mouse-window, the lower border.

INTVAR AT $84ba; mouse_left Variable: mouse_left
Defines in a mouse-window, the left hand border.

INTVAR AT $84bc; mouse_right Variable: mouse_right
Defines in a mouse-window, the right hand border.

ROW 63 INTVAR AT $84c1; mousepic Variable: mousepic
Mouse-pointer in sprite format.

BYTEVAR AT $8501; max_speed Variable: max_speed
Is the maximum mouse speed.

BYTEVAR AT $8502; min_speed Variable: min_speed
Is the minimum mouse speed.

BYTEVAR AT $8503; accel Variable: accel
Is the mouse acceleration parameter.

BYTEVAR AT $8506; inputdata Variable: inputdata
Gives the direction of movement:
0= right, 1= right - up, 2= up, 3= left - up, 4= left
5= left - down, 6= down, 7= right - down.

INTVAR AT $850a; random Variable: random
Integer parameter, lucky no. Is set from new with each interrupt.

BYTEVAR AT $851c; alarm Variable: alarm
If $ff has been set, then the alarm is active.

2.1 Programming-Commands
2.1.1 BASIC - Oriented Commands

ADD string,byte
This command enables an ASC-ll code character (byte) to be added to the end of a
caracter-chain (string). If the maximum length of the string has already been achieved,
then the additional character will be ignored.
Eg.: string="Hello": ADD string,65 : PRINT string : displays = "HelloA".

ADR byte (integer)
This operator shows the address (integer) of the variable (byte) in the main memory.
Eg.: integer=(ADR(byte)) : the integer receives the memory address of the byte.

ADR integer (integer)
This operator shows the address (integer) of the first byte of the variable (integer) in the
main memory.

ADR real (integer)
This operator shows the address (integer) of the first byte of the variable (real) in the
main memory.

ADR string (integer)
This operator shows the address (integer) of the first character of the variable (string) in the
main memory.

ADR label (integer)
This operator shows the address (integer) of the first memory-position, that the address
of the jump-table (label) receives in the main memory.

ADR file (integer)
This operator shows the address (integer) of the first byte of the data-segment (integer)
for files (file) in the main memory.

ADR object (integer)
This operator shows the address (integer) of the first byte of the object (object) in the main memory.

ADR byterow (integer)
This operator shows the address (integer) of the first byte of the first element of the data-sequence (byterow) in the main memory.
Eg: integer=(ADR byterow) : The integer receives after accessing the memory-address of the first data-element of "byterow".

ADR introw (integer)
This operator shows the address (integer) of the first byte of the first data-element of the data-sequence (introw) in the main memory.

ADR realrow (integer)
This operator shows the address (integer) of the first byte of the first data-element of the data-sequence (realrow) in the main memory.

ADR stringrow (integer)
This operator shows the address (integer) of the first character of the first data-element of the data-sequence (stringrow) in the main memory.

ASC string,byte1 (byte2)
This operator shows the ASCII-Code (byte2) of the character byte1 (positions.no.) from the character-chain string as byte-variable. The no. of the first character is zero (0)
Eg.: test=(ASCII"Begin",0) : Gives the first parameter = 66

BYTE AT integer (byte)
This operator gives the sum (byte) of the contents of a memory-position at a particular address (integer). This operator can also be defined with a definite parameter so that a byte can be saved.
Eg.: (BYTE AT $5000) = 10 : This command defines at the memory-position the decimal parameter 10.

BYTEROW AT integer (byterow)
This operator acts as a simulation for a declared data-sequence (byterow) from a defined memory-position (integer) and can therefore be used as a reserve.
Eg.: (BYTEROW AT $5000)<5> = 10 : This command defines in the data-element no.5, which is the data-sequence starting at memory-position $5000, the parameter 10.

BYTE real (byte)
This operator converts a real-no. (real) into the byte-format and returns the sum (byte). If the no., that is to be converted, outside the the permitted parameters (0-255) then a error-parameter will be given !

CHR byte (string)
This operator shows a character-chain (string) that contains a character whose ASCII-Code is given as parameter (byte).
Eg.: text=("Hello"-(CHR55)) : After the command the text is "HelloA".

DEC byte
Reduces the parameter of the variable byte by a factor of one. If the variable had the parameter zero (0) then it will receive, after the command, the parameter 255.
Eg.: test=10 : DEC test : test = 9.

DEC integer
Reduces the parameter of the variable integer by a factor of one. If the variable had the parameter zero (0) then it will receive, after the command, the parameter 65535.
Eg.: test=1025 : DEC test : test = 1024.
END
The end character of a program. The screen will be erased, the program exited and you return to the DeskTop, therefore all previously opened files must be closed. Geos will search for the DeskTop on the drives A and B, and when found - load it.

GET byte
This command gets a character from the keyboard. The keyboard will be interrogated, if no key is pressed then byte = 0!

GOSUB label
This command instructs the program to jump to the label label. It is a sub-program that can in turn be exited with the command RETURN. If only the label-name is given, then this will be interpreted as the command GOSUB label.
Eg.: GOSUB key : jumps to the label key.

GOTO label
This command instructs the program to jump to the label label. - There is no command for the return jump.

HEXSTR byte
This command converts the variable byte in a HexDec no. and gives the sum on to a character-chain. Eg.: Number = (HEXSTR 10) : number is then "0a".

HEXSTR integer
This command converts the variable integer in a HexDec no. and gives the sum on to a character-chain.
Eg.: Number = (HEXSTR 10) : number is then "0a".

HIGH integer
This function gives the high-parameter of the variable integer as a byte-variable.
The opposite pole to this command is LOW.

INC byte
Increases the parameter of the variable byte by the factor one (1). If the variable had the parameter 255 then after the command the variable will have the parameter 0.
Eg.: test=10 : INC test : test => 11.

INC integer
Increases the parameter of the variable integer by the factor one (1). If the variable had the parameter 65535 then after the command the variable will have the parameter 0.
Eg.: test=1024 : INC test : test => 1025.

INPUT string
This command reads the character-chain string ! If the string has been previously defined, then the string will be displayed and the text-cursor appears at the end of the string (which can now be edited) press RETURN to confirm the string.

INPUT text,string
This command displays the character-chain text and waits for the confirmation of the character-chain string : RETURN.

INT byte
The variable byte is converted to an integer-variable.
Eg.: integer = (INT 10) : integer = 10

INT real
The variable real is converted to an integer-variable.
Eg.: integer = (INT 10.5) : integer = 10

INT byte,byte 1
The variables byte (LOW-parameter) and byte 1 (HIGH-parameter) are combined to an integer-variable. Eg. for addresses.
Eg.: test=(INT byte,byte 1)
INT AT integer (integer)
This operator gives the sum (integer) of the contents of a memory-position with a given address (integer).

INTROW AT integer (introw)
This operator is used as a simulation of a declared data-sequence (introw) from a defined memory-address (integer) and be used as a reserve.
Eg.: ((INTROW AT $5000)<10>) = 1024 : This command defines in the data-element no.10, which is the data-sequence starting at memory-position $5000, the parameter 1024.

LEFT string,byte
This command takes from the character-chain string - from left (the beginning) - the standing (byte) character and gives the sum as string.
Eg.: text = "Test" : text1 = (LEFT text,0) : text1 = "T".

LEN string
This function calculates the no. of characters in the character-chain string and gives the sum as byte-variable.
Eg.: x=(LENstring) : x = 6

LOW integer
This function defines as byte-variable the LOW-parameter of the variable integer.
The opposite pole is HIGH.

MID string,byte,byte1
This command removes from the character-chain string from the position byte the no. of byte1-characters and gives them as character-chain.

PLEN string
Calculates the length of the character-chain string in graphic-screen-points and gives the sum as an integer-parameter. This is useful for formatted displays.
Eg.: test = (PLEN"contents")

PRINT string
This command gives the string, i.e., displays the string on the screen. If a "window" has been defined beforehand, then the display will appear within the "window." The display will be ended with screen-CR, the text-cursor can be pre-defined for text entry with the command SETPOS... within the screen (window). The command ? string functions in the same way. Using this commando structure it is possible to show various text-styles, the text-style will be announced with the character T, the following parameters are allowed:
/P = normal /B = bold  /I = italic   /O = outline   /U = underlined
The display of the command-characters in HexDec. format is therefore possible.
Eg. 160 displays the Commodore (C=) character on the screen. If a fore-slash (/) (Shift-7) is wanted, then 2 slashes must be written into the command, otherwise the program will be waiting for a style change : " / / ".

PRINT string;
Has a similar function to the command PRINT, here the text-cursor remains behind the string in the same line. There is no screen-CR. The command ? string has the same function.

PUT byte
Displays on the screen the ASC-ll character byte, the command characters are allowed.

PUTDEC integer,byte
Displays on the screen a faster number (integer), this doesn't need to be first converted to character-chain. byte has the following functions:
Set Bit 7 = the display is tied to the LH border, otherwise RH.
Set Bit 6 = without zero (0) display, otherwise 5 digits with a zero (0) prefix when required.
Bit 0 - 5 = width of the required display-field, if the display is tied to the RH border (Bit7=0)
REAL byte
Converts the variable byte to a real-no.
Eg.: byte=10 : real=(REALbyte) : real=10.

REAL integer
Converts the variable integer to a real-no
Eg.: integer=1024 : real=(REALinteger) : real=1024.

REAL string
Converts the variable string to a real-no
Eg.: string="10" : real=(REALstring) : real = 10

REAL AT integer (real)
This operator gives as sum (real) the contents of a memory-position of a particular address (integer).

REAL AT integer (real)
This operator serves as a simulation for a declared data-sequence (realrow) from a pre-defined memory-position (integer) and can be used as a reserve.

RETURN
This command jumps from a sub-program, that has been previously called with the GOSUB label, back to the position behind the command GOSUB.

RIGHT string,byte
This command takes from the character-chain string - from right (the end) - the standing (byte) character and gives the sum as string.
Eg.: text = "Test" : text1 = (RIGHT text,1) : text1 = " t ".

SETHIGH integer,byte
Converts the as integer existing high-value-byte in byte.

SETLOW integer,byte
Converts the as integer existing low-value-byte in byte.

STR byte
This function converts a byte-no. in a string.
Eg.: string = (STR98) : string = "98".

STR integer
This function converts an integer-no. in a string.
Eg.: string = (STR98) : string = "98".

STR real
This function converts a real-no. in a string.
Eg.: string = (STR10.94) : string = "10.94"

STR byte AT integer
Converts a byte-long character-chain from address integer to a string. byte is the no. of characters and integer the start-address.
Eg.: test = (STR 16 AT $8465 ) gives the current Printer-Name.

STRROW byte AT integer (strrow)
This operator serves as a simulation of a declared data-sequence (strrow) from a defined memory-position (integer) and can be used as a reserve. byte is here the length of the character-chain.

VAL string
Converts a string in a real-variable. VAL is the opposite function to STR. Letters and special-characters lead to errors.
Eg.: string = "123" : real = (VALstring) : real = 123.
WAIT byte
Waiting-loop, controlled by the interrupt. The interrupt is activated every 1/50 sec. The parameter 50 = 1 sec. If the interrupt is in-active then WAIT doesn't function!

2.1.2 Program - Loop commands
One of the most serious deficiencies when programming in BASIC is that there isn't a command that allows a structured flow in complex programs. The commands GOTO and GOSUB make a program complicated and diffuse. When programming with geoCom you should try and place large, complex program's in a number of individual modules, not withstanding the following commands and loop definitions should be fairly useful!

IF...
The command IF is very wide-reaching, it can be used for many functions.

IF......ENDIF
Is IF true, then everything between IF and ENDIF will be carried out.
Eg. IF (y == 10) : e = 2 : GOTO end : ENDIF -> when y = 10, then e = 2 and goes to end.
ENDIF must be at end!

IF......GOSUB label
Is IF true, then the program will jump to the sub-program label.

IF......: GOTO label
Is IF true, then the program will go to the label label.

IF......: THEN....:ENDIF
Is the parameter to IF true, then the command to go to THEN will follow. Everything between THEN and ENDIF will be carried out.

IF......THEN......ELSE......ENDIF
Is the parameter IF true, then the command to go to THEN will follow. Otherwise the command ELSE... will be carried out. The function can be compared to IF...THEN... but this time you have the un-true option.

This IF... loop format is used often.
Eg. Keyboard interrogations : IF (keydata == $f1) GOTO end
or by mouse interrogations (area-interrogation) : IF (REGION10,100,100,200) : x=(x+3) : ENDIF

According to the individual situation it is possible to select the relevant loop-format. Take care that with Eg. Keyboard-interrogations, that the interrogation should be as short as possible, before the return to the Mainloop - otherwise the program will appear to run slowly.

REPEAT......UNTIL
This loop (REPEAT) will repeats itself until the required parameter (UNTIL) is achieved. It is a similar loop to FOR...NEXT. The loop will be left when the required parameter has been affirmed. It is possible to use the section UNTIL separately, this would then be a waiting-loop that would remain active until the required parameter is affirmed.
Eg.: REPEAT : a =(a +1) : UNTIL (a> 71)

WHILE......LOOP
This loop will repeats itself until the required parameter is declared un-true. It is a similar loop to FOR...NEXT. It is possible to use the section LOOP separately, this would then be a waiting-loop that would remain active until the required parameter is declared un-true.
Eg.: WHILE(x=101) : x =(x+1) : LOOP. The loop will be left at parameter 101.

Due to technical requirements there are no FOR...NEXT-commands within the geoCom-command-area. A FOR...TO...Next-loop can be easily set up with a REPEAT..UNTIL-loop. The variable counter, the start-counter-variable begin the end parameter end and the increment-variable step should all be defined as an integer-variables.
The start-counter parameter is the number before TO, and the end-parameter is the number after TO. STEP is, when not defined = 1. The loop is then: counter = begin
REPEAT : followed by the relevant commands (Eg. POKE..)
counter = (counter + step) : UNTIL counter >= end
As a comparison :
FOR begin TO end STEP step : followed by the commands : NEXT

Generally : is the so-called counter 1 that is x=(x+1), ie. -1, then can optionally INCx or DECx be used. This is faster and saves memory space.
Eg.: counter=0 : REPEAT : INCcounter : UNTIL(counter==100)

2.2 Program - Structure Commands
2.2.1 Source-Code - Parameter-Commands
Program structure commands are important for the compilation, geoCom uses the information contained within these commands for memory assignment, the Info.-Block and for defining labels and variables. A total of 127 label commands may be defined, this includes in multiple section programs the Global- and other, individual modules.

AUTHOR name
Must be included in the Definitions-Section. The name is entered as a string, the string may be up to 20 characters long - a string that is to long will be cut short! If the name-string is not entered, the standard parameter "Falk Rehwagen" will be used.
Eg.: AUTHOR"Denis Doehler"

BYTEVAR byte
Must be included in the Declarations-Section. This command is used to define the variable-names. All the byte-variables must be entered here. A maximum of 16 byte-variables can be entered behind one BYTEVAR command.
Eg.: BYTEVAR ab,numb,test,xa,xc,xe

CLASS name
Must be included in the Definitions-Section. The file-class of the program must be entered in this string, the string may be up to 16 characters long, - a string that is to long will be cut short! usually the first 12 characters are used for the program name, the characters 13-16 are then used for the versions-no. If the name-string is not entered, the Source-Code-ext name will be defined as the class !!
Eg.: CLASS"Pattern Show V 1.0"

CODE integer,integer1
Must be included in the Definitions-Section. This command is used to define the "from - to" memory-area for the Code-Area, if you don't apply a definite set of parameters then geoCom will define the standard parameters.
Eg.: CODE$2800,$4200 (defines the memory-area : $2800 to $4200).

CONST integer,integer1
Must be included in the Definitions-Section. This command is used to define the "from - to" memory-area for the constant-parameters, if you don't apply a definite set of parameters then geoCom will define the standard parameters. Max. parameter is $5000!
Eg.: CONT$4000,$4900 (defines the memory-area : $4000 to $4900 as Constant-Parameter-Memory-Area).

FILETYP byte
Must be included in the Definitions-Section. This command is used to define the file-type, (according to the geo-Standard), whether application = 6 or auto-exec = 14. If you don't include this command, then geoCom sets the standard parameter : application.
Eg.: FILETYP 6

INTVAR integer
Must be included in the Definitions-Section. This command is used to define the required integer names. All the integer-variables must be entered here. A maximum of
LABEL label
Must be included in the Declarations-Section. This command is used to define the
required label names. All the label-names must be entered here. A maximum of 16
label-names can be entered behind one LABEL command.
Eg.: LABELEnd, begin, loop, key

NAME name
Must be included in the Definitions-Section. This string is used to define the name of
the program which is to be created. The string may be up to 16 characters long.
Eg.: NAME "Pattern Show"

OBJECT name
Must be included in the Declarations-Section. This command ties in the in the OBJFILE
incarcerated graphics, bitmaps etc. The name is the in ObjectEdit defined label-name

OBJFILE object
Must be included in the Declarations-Section. This command ties in during the
compilation the, with ObjectEdit, created objects. The string is the name of the
self-created Object-file and may be a maximum of 16 characters long.

REALVAR real
Must be included in the Declarations-Section. This command is used to define the
required real-number-variables names. All the real-number-variables must be entered
here. A maximum of 16 real-number-variables can be entered behind one REALVAR
command.
Eg.: REALVARw2, w3, w4

STARTFLAG byte
Must be included in the Definitions-Section. This command is used to set the
screen-modus in Geos 128. The following parameters are permitted:
$00 = only 40 Characters  $40 = 40 and 80 Characters
$80 = can only run under Geos 64  $c0 = only 80 Characters

STRLen byte
Defines the maximum length of character-chains, that can be read, carried over,
calculated or otherwise not be pre-defined by the user. If you do not define this
command, then geoCom will set a standard parameter of 254 Characters per
character-chain.
Eg.: STRLEN 60 -> Sets all the following strings a maximum of 60 characters.

STRVAR byte; string
Must be included in the Declarations-Section. This command is used to define the
required string-variable-names and their size. All the string-variable-names must be
entered individually here. Byte is the size of the individual string.
Eg.: STRVARZ; dummy - dummy is 2 characters large.

VAR integer, integer1
Must be included in the Definitions-Section. This command is used to define the
required "from to" memory-area for all the variables. If you do not define this
command, then geoCom will set a standard high-parameter of $0000, because the
area upwards of $6000 is required for the background-memory. ({$7900 max top limit
for programs!}
Eg. VAR$5000,$5900 : defines memory-area $5000 to $5900 as variable- memory-area.

2.2.2 Variables - Memory Areas
It is possible to directly define a memory-area for your variables. The development of
single-dimensional data-sequences is also possible: a<0>, a<1>, a<2>.

ROW quantity BYTEVAR byte
Creates a data-sequence on the variable byte with a quantity quantity, where a
number of variables can be defined.
Eg. ROW 1920 BYTEVAR printerbuffer : creates a data-sequence with 1920 Bytes for the variable printerbuffer.

ROW quantity INTVAR integer
Creates a data-sequence on the variable integer with a quantity quantity, where a number of variables can be defined.
Eg.: ROW 10 INTVAR test : creates a data-sequence with 10 Integer-no's. for the variable test. The variables can be accessed with : test<0> to test <9>.

ROW quantity REALVAR real
Creates a data-sequence on the variable real with a quantity quantity, where a number of variables can be defined.

ROW quantity STRVAR byte; string
Creates a data-sequence on the variable string with a quantity quantity, where byte is the length of a character-chain.

ROW quantity BYTEVAR AT memoryaddress; byte
Creates a data-sequence on the variable byte with a quantity quantity, from the address memory address. This can be entered in either the Hex or Dec. numeric form.

ROW quantity INTVAR AT memoryaddress; integer
Creates a data-sequence on the variable integer with a quantity quantity, from the address memoryaddress.
Eg.: ROW 10 INTVAR AT $6a00;test : Creates a data-sequence with 10 integer-no's from the variable test from memory-address $6a00. The variables can be accessed with : test<0> to test <9>.

ROW quantity REALVAR AT memoryaddress; real
Creates a data-sequence on the variable real with a quantity quantity, from the address memoryaddress.

ROW quantity STRVAR byte AT memoryaddress; string
Creates a data-sequence on the variable string with a quantity quantity, from the address memoryaddress. byte is a character-chain.

2.2.3 Source-Code - Module Commands

GLOBALEND
End label for the main part of a program. Everything that is in front of this command always remains in memory. - Global Section.

GOTOMOD byte
Jumps to the beginning of the module "byte", where the first module is the module 0 and stands behind the Global-Section. In this case byte must be a number between 0 and 127.

INCLUDE string
If this command is included in a Source-Code, then geoCom will exactly at this position - during the compilation - tie a second Source-Code (Eg.MODULE1) in (the additional Source-code must be on the same disk.). The string must be the name of the additional Source-Code. This enables you, when writing a long and complicated Source-Code, to divide the individual modules into separate, logical, Source-Codes. The command INCLUDE may not be included in a further modular Source-Code ! !

OVERLAYMOD
End label for a single module. Everything that comes after this command, will be assigned to the following module. - Local Section.
The last module does not require this end-label.
Eg. for a multiple-section-program.
- GeoCom V1.5 - Commands etc -

NAME...CLASS...AUTHOR...CODE...CONST...VAR...
BYTEVAR...INTVAR...ROW.BYTEVAR
LABEL...
Main-Global-Section program-text

GLOBALEND

'Module 0'
BYTEVAR...-Module variables
LABEL...-Module label names
1st. Module program-text
including GOTO
MODIFICATION

'OVLAYERMOD'

'Module 1'
BYTEVAR...-Module variables
LABEL...-Module label names
2nd. Module program-text

2.2.4 Dialog Box - Commands
GeoCom is supplied with 6 standard-dialog-boxes that can be accessed with commands.
A 7th. command is provided for accessing your own dialog-boxes that you have created
with Objectedit. The standard-boxes are so set-up that no new parameters are required,
the dialog-boxes will always be correctly positioned to accept the defined character-modus
and all entries and displays (incl. mouse) are only relevant to the latest dialog-box.

CREATEBOX name,text,byte
This command watches over an oft used dialog-box, the parameter "name" should be that
of your yet-to-be created program. This dialog-box should allow the user to change the
disk and / or drive. The file name, that is to be entered or edited should be passed to a
character-chain (name). if this character-chain already includes a name before the
dialog-box appears then this will be shown in the dialog-box. The character-chain-name
should not exceed a length of 16 characters, longer names are not allowed under Geos.
In order to make the dialog-box more flexible, the query-text (text) should be a variable.
Finally a number (byte) closes the command, this no. decides whether the drive disk. may
be changed or not, the no. also defines whether the drive-icon should be displayed in the
dialog-box or not. Normally is a Disk. change in REU drives or in the drive where the file is
open not possible. The bits of the command "byte" have the following definitions :
Bit 0 = 1  Drive A has no Disk. Icon
Bit 1 = 1  Drive B has no Disk. Icon
Bit 2 = 1  Drive C has no Disk. Icon
Bit 3 = 1  Drive D has no Disk. Icon
Bit 7 = 1  No Disk. Icon should be displayed
You can combine the bits as required. It is possible to activate the F-Key interrogation
within the dialog-box - to select the drives A-D. The programmer can, using the
system-variable dbstat, test what action causes the dialog-box to be exited. A 16 byte
long string must be at first like name. This must be create before.
Eg. CREATEBOXname."/BPlease insert new name.";"0

Drive B: RAM 1571
Please insert new name:
text edit

Drive Abbruch

DBRET byte
This command enables the exit from dialog-boxes (your own). The command should stand
at the end of sub-routines that are called by the dialog-box-mainloop. After calling this
command the box will be removed from the screen and the system relevant data will be
re-activated - the system data was saved in the memory as long as the dialog-box was
active - the control is given back to the program Mainloop. The parameters from byte
are given over to the internal system-variable dbstat, enabling the main-routine to
determine what action, if any, was defined within the dialog-box.
Eg. DBRET 31
DIALOG object
This command displays a dialog-box on the screen, that has been developed previously using ObjectEdit. We recommend here the use of the command DBRET and dbstat. object as the name of the ObjectEdit dialog-box.

DISK
Displays on screen the dialog-box "Please insert a new disk. !", together with the current drive. This command can only be accessed when no files are open!

ERROR
Displays a dialog-box with an error-number and an error-text, this should appear (if required) after a disk-file operation. The error-number will be called from the system-variable ISTAT. An error-box will also be displayed by call-ups when no errors have been registered.

INPBOX text;string 1
Displays a dialog-box for entering / editing text. The variable text will be shown and the loop will wait until string 1 has been confirmed, string 1 is confirmed with RETURN the loop is then ended. The semicolon between text;string is very important!
Eg.: INPBOX "$Btextshow/P"; "text edit"

OPENBOX name,byte,string 1, byte 1
This command displays a file-selection-box. The following parameters must be pre-defined otherwise they will be ignored:
- name Includes the file-name that is chosen by clicking on OPEN in the file-choice list.
- byte Geos-file-type
- string 1 Geos-Class-Type in string format
- byte 1 Same parameter as CREATEBOX.

STRTNX text,text,text
This command displays on the screen a dialog-box with three strings and an OK-box.
the strings must be divided by commata, whereby each string will be displayed in a unique line. By defining /B,/P it is possible to display the text in the various styles. The box can be exited by clicking on the OK area. The box can be used for various functions, Eg. Info.-boxes. Should a string remain empty, the inverted commas should still be given. It is possible to activate the various text-styles. Variables that have been converted to strings can also be displayed.

Eg.: STRBOX*/Otext /P","/B text 2","/Ptext 3"

2.2.5 Menu Bars - Commands
FIRSTMENU
If, in your program a menu-routine should be activated with a click in a menu, then this command should be the first command - so the the previous menu can be erased.

MENU object,byte
The variable byte places the mouse-pointer on the defined object of a horizontal menu-bar.
Eg. byte = 1, the mouse-pointer will placed on the first object.

REDRAW
This command is similar to FIRSTMENU, the menu is not erased but re-drawn and re-offered for your selection.

RETMENU object
It is possible to insert, between menu and sub-menu, a routine.
Eg. To erase the colours in the menu-area.
The command RETMENU must stand at the end of the routine, the object must be the sub-menu.

2.3 Process-, Memory-, Machine Code-, Commands
The Geo-Mainloop ensures that the keyboard, mouse, and clock etc. are regularly interrogated and updated, the Mainloop is also responsible for running the various system-internal processes Eg. RAM Process... It is possible, using the following commands, to integrate your own home-made processes in the Mainloop.

2.3.1 Process - Commands
BLOCK byte
Blocks the process with the number byte. This doesn't affect the Timer, the process is no longer interrogated.

ENABLE byte
The process byte will definitely be called upon in the next Mainloop run through.

FREEZE byte
Blocks the process with the number byte. The Timer will be "frozen", when the process is re-activated the Timer will be re-started.

INTERRUPTOFF
Switches the computer interrupt OFF, this is especially important for area-interrogations.
Commands that are driven by the interrupt (WAIT) are now disabled !

INTERRUPTON
Switches the computer interrupt ON, this is especially important for area-interrogations.
Commands that are driven by the interrupt (WAIT) are now enabled !

MAINLOOP
Using the command MAINLOOP it is possible to directly access the Geo-Mainloop. The
control over menu's, pictogram's and processes etc can be directly tied into the Geo-Mainloop, it is advisable to ensure the interrogation of the mouse and keyboard beforehand with the command: ON byte GOTO label.

ON byte GOTO label
Geo has a very important command: MAINLOOP. This loop ensures that all drivers and processes are called-up (running the clock). The command: ON byte GOTO label is a programmers tool, sing this command it is possible to ensure that after forcing the Mainloop to carry out "strange" tasks that Mainloop is also directly commanded to carry out the usual tasks: mouse & keyboard. The parameters:
ON 0 GOTO label, forces the jump to label label, when a key is pressed. The actual interrogation (which key has been pressed) must be inserted by the programmer, the parameter of the latest key-function can be found in keydata.
ON 1 GOTO label, branches the program to the label label when the mouse-button is pressed. The actual interrogation (which button has been pressed) must be inserted by the programmer.
ON 2 GOTO label branches the program to the label label when the text display runs over the right-hand border of the current "window".
ON 3 GOTO label branches the program to the label label when a dialog-box is being exited.
ON 4 GOTO label branches the program to the label label when a "System-Error $XXXX" appears.

PROCESS object.byte
Ties in the process-table object, byte is the no. of processes. Please bear in mind that the speed of the program will be affected parallel to the no. of processes that you activate: more processes = less speed.

RESTART byte
Re-starts a blocked or frozen process (FREEZE or BLOCK).

UNBLOCK byte
Is the opposite command to BLOCK byte. The process byte will no longer be blocked, the process will run normally after the next interrupt.

UNFREEZE byte
Is the opposite command to FREEZE byte. The "frozen" Timer will be re-activated, the process will be accessed.

2.3.2 Memory - Commands
AT
AT address l Enables the saving and reading of variables and variable-sequences after a defined address in the variable-memory.
Eg. BYTEVAR AT INTVAR AT, REALVAR AT STRVAR 16 AT BYTEROW AT
INTROW AT REALROW AT STR 18, ROW AT

INITIO
Switches to the memory-area $d000-$dff i.e. the old Commodore-System. Following this command it is only possible to use POKE and PEEK commands, the occupation-tables for this memory-area can be found in every C 64/128 BASIC-Handbook. It is not possible, following this command, to set the interrupt with the Mainloop. Commands such as PROCESS and WAIT will not work, if you don't respect this rule - the program could hang-up.

DONEIO
Returns the Internal-System back to Geo. Commands that act on the Commodore-System are de-activated.

FILL integer, integer 1 byte
Fills the memory with the character byte from the memory-address(HEX) integer 1 with the quantity integer.
Eg. FILL 1000,$7000,255: Fills from address $7000 1000 Bytes $ff
MOV E integer,integer 1 ,integer 2
Moves the quantity integer 2 internally from the start-address integer to target-address integer 1. (Address assignments in HEX).
Eg.: MOVE$6000,$7000,100: Moves 100 Bytes from memory $6000 to $7000.

PEEK integer
Gets a with integer defined memory-address a parameter in Hex. The integer-variable can be written in Hex ($56f4) or Dec (45678).  
Eg. a = (PEEK$56f4): a includes the contents of the memory-position $56f4.

POKE integer,byte
Fills the memory-position integer with the parameter byte. Syntax as in PEEK.

SETVECTOR integer,label
In the Geos-memory are a number of addresses that are at certain times automatically accessed. Using this command it is possible to force these vectors to access you own labels.
Eg., SETVECTOR $84a1,return: should the address $84a1 be accessed then jump to label return.

TEXT (strrow,byte)
This command enables your, created with OBJECT EDIT, string-chains to be accessed where strrow defines the string-chain and byte the relative data-element.
Eg. ICON EDIT.

2.3.3 Machine-Code - Logging In
CALLSYS integer,introw
This command is a tool for calling-up the various, powerful Geos-internal-systems-routines from a particular memory-address (integer). These routines are usually offered as options, using the Geos-Register, to the Zero-Page or Processor-Register. Unfortunately geoCom cannot directly access the routines, therefore they must be set before geoCom calls the relative sub-routine. This function is covered by the command CALLSYS, the programmer must enter the relevant information for the Geos-Processor-Register using a data-sequence (introw). This contains the register-data in the following format:

<table>
<thead>
<tr>
<th>Data-sequence elements</th>
<th>includes the GEOS-Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 . . . 15</td>
<td>GEOS-Register r0 . . . r15</td>
</tr>
<tr>
<td>16 low parameter byte</td>
<td>Flag-register</td>
</tr>
<tr>
<td>16 high parameter byte</td>
<td>Accumulator</td>
</tr>
<tr>
<td>17 low parameter byte</td>
<td>X-Register</td>
</tr>
<tr>
<td>17 high parameter byte</td>
<td>Y-Register</td>
</tr>
</tbody>
</table>

These parameters must be, as required, set before the command is called. If you access the wrong parameter then this could lead to a System-Crash. To set the required bytes use SETLOW and SETHIGH.

CALL integer
This command is used to call-up a Machine-Code routine from a defined address (integer). This command should be handled with extreme care, it can lead very easily to a System-Crash ! !

2.4 Disk. - and File - Commands

These commands should always be "handled with care" to avoid possible damage to disks, please don't open a file and allow the disk, to changed directly afterwards! A number of the commands are supplied with this protection built in, thus some functions are not available when a file is open!

2.4.1 Disk. - Commands
CAPACITY byte (integer)
This operator shows (integer) the quantity of free, occupied and total space on the disk. in the current drive. The parameter (byte) defines the display modus:
0 = total disk. capacity (blocks), 1 = No. of free blocks, 2 = No. of occupied blocks
- GeoCom V1.5 - Commands etc -

Eg.: A=(CAPACITY0) : A includes the disk. capacity (Integer no. )

DEVICE byte
byte = current drive. The parameters are always:
8 = Dr. A  9 = Dr. B  10 = Dr. C  11 = Dr. D

DISKNAME
This command gives, in string-format, the current disk. name.
Eg.: name = (DISKNAME)

DRIVE
This command changes, in logical order (A - B - C - D), the current drive, the non-active drives are ignored.

OPENDISK
This command initializes a new disk in the current drive, simultaneously a number of memory-addresses and the function DISKNAME are updated. If errors occur they will be recorded in the variable iostat.

2.4.2 File - Commands
APPENDREC file
This command hangs a data-sequence on the current VLIR-file. It will be attached to the current data-sequence and so becomes the current data-sequence. A maximum of 127 data-sequences (records) are allowed, if the parameter 127 is exceeded an error-message will be displayed.

CLOSE file
This command closes the defined file (file). A VLIR-file where a further data-sequence is still open cannot be closed. A possible error will be recorded in iostat.

CREATE name,object
This command creates, in the current drive, a file that is specified by the file-Header - developed under ObjectEdit (includes Class etc.) The file-name must be entered as a character-chain (name), it is therefore possible to create a number of identical files with different names. A possible error will be recorded in iostat. VLIR files do not receive after the "creation" a data-sequence, these must be created using APPENDREC.

DELETE name
The file name will be erased from the current disk in the current drive, it is not possible to erase a file that is currently open - an error-message will not be displayed. Information such as Filetype or Write-Protection have no bearing on the command ! A possible error will be recorded in iostat.

DELETEREC file
This command erases the current data-sequence from an open VLIR-file (file). All the following data-sequences move up one (1) position. A possible error will be recorded in iostat.

FCLASS name,byte,byte 1 ,byte row
Using this command it is possible to read all the files on the current disk. with the same file-type. Parameters:
name = Class, byte = File-type, byte 1 = Quantity, byte row = includes the "found" files.
A string-variable with 16 characters must be defined beforehand.
backbyte gives the quantity of not-occupied data-elements.
Eg.: ROW 8 STRVAR16; da_s -> 8 Strings with length 16 and Name da_s. Can be called up later with da_s<0> to da_s<7>.
FCLASS "-5,8,da_s" --> searching for 8 Files of type 5 (Desk Access) and lays the answer in da_s<0> to da_s<7>.

FINDFILE name
Searching for the file name on the disk. in the current drive. If name is found, the parameter will be placed in the variable-area direntry, if name is not found the the error-message will be placed in the variable iostat, this can be accessed.
FILEVAR file
Must be included in the Definitions-Section. This command is used to define a variable-name for the usage of files. The relevant file will be defined by the OPEN command.
Eg.: FILEVAR datafile

GET file.byte
This command accesses from the SEQ.file file a character as byte-variable, where the command GET reads continuously. An internal pointer for this command is available, the pointer is set back using the command RESET file. It is important to respect the variable-format.

GET file.integer
This command accesses from the SEQ.file file a character as integer-variable, where the command GET reads continuously. An internal pointer for this command is available, the pointer is set back using the command RESET file. It is important to respect the variable-format.

GET file.real
This command accesses from the SEQ.file file a character as real-variable, where the command GET reads continuously. An internal pointer for this command is available, the pointer is set back using the command RESET file. It is important to respect the variable-format.

GET file.string
This command accesses from the SEQ.file file a character as string-variable, where the command GET reads continuously. An internal pointer for this command is available, the pointer is set back using the command RESET file. It is important to respect the variable-format.

GETBLOCK byte,byte 1,byterow
This command reads a complete disk block (Sector= 256 Bytes). The parameters are:
byte = Track  byte 1 = Sector  byterow = Address-area (Begin)

GETFILE name,byte,byte 1,string
Universal main-routine for loading and starting programs. The parameters are:
name = File name  byte = Geos-File-type  byte 1 = Drive no.
string = here can be found the required docu. name (when you want to open the doc. and go to geowrite) or an empty string!

INITFILE file
This command erases all the data-variables Eg. Record-no's.

INSERTREC file
This command inserts a record before the current data-sequence, this is then the current data-sequence. A maximum of 127 data-sequences (records) are allowed, if the parameter 127 is exceeded an error-message will be displayed.

OPEN file,name
This command opens the file name and defines the file-variable file to the respective file. name is the name of the file in the DeskTop and file the FILEVAR-iable in your program.

OPENREC file,name
This command opens the current data-sequence as a .SEQ file of an existing VLIR-file that must be on the current disk in the current drive.

PTREC file,byte
This command specifies which data-sequence of an open VLIR-file should be back-accessed. byte is the record-no. The first data-sequence has the no. 0.

PUT file,byte
Writes a byte byte in a previously opened .SEQ file file. An internal pointer will be used, it is the opposite command to GET. The pointer can be set-back with the command
RESET file.

PUT file, integer
Writes an integer-no. integer in a previously opened SEQ file file. An internal pointer will be used, it is the opposite command to GET. The pointer can be set-back with the command RESET file.

PUT file, real
Writes an real-no. real in a previously opened .SEQ file file. An internal pointer will be used, it is the opposite command to GET. The pointer can be set-back with the command RESET file.

PUT file, string
Writes an character-string in a previously opened .SEQ file file. An internal pointer will be used, it is the opposite command to GET. The pointer can be set-back with the command RESET file.

PUTBLOCK byte, byte 1 ,byterow
This command writes a complete disk. block (256 bytes) back to a disk, byte is the respective track and byte 1 the sector. The memory-areabyterow must be previously defined within the Definitions-Area (256 bytes). The file must already be open. This is a command for .SEQ files.

READ file, start-address, size
This command reads the maximum quantity size of characters and places them at the specified address in the variable-memory. **backward** receives, as parameter, the address (main-memory) of the last-read bytes.

Eg. **READfile,(ADRbuffer),254** - reads from the file 254 characters (Sector) and places them in the variable buffer. Additionally, ADR reads the start-address of the variables. The variable buffer then receives the complete sector (without chain-bytes).

RECCINFO byte
Delivers the open VLIR-file's information to the records. byte had the following parameters:

0 = not changed yet (<> 0 change has occurred)
1 = current data-sequence
2 = used data-sequence

RENAME name TO name 1
This command renames the file name as name 1.

RESET file
Sets the GET / PUT pointer back to 0.

SETWRITE file
This command enables you to “write” into a .SEQ file, this command is possible because when you open a .SEQ file the system switches to READING.

WRITE file, start-address, size
This command writes the maximum quantity size of characters (EG. after setting a record) from the specified address in the variable-memory to disk.

Eg. **WRITEfile,(ADRbuffer),254** - writes in the file file a total of 254 characters (Sector) from the variable buffer. The respective record must have been set beforehand, additionally, the BAM will be re-written.

2.5 Printer - Commands
2.5.1 Printing- Preparations
If you want to make a printer-hardcopy of you geoCom programs, then you may only fill the variable-memory up to $7900. The area $7900 - $7fff is reserved for the printer-driver and any variables that you place in this area will erased when you load the printer-driver.
PRINTINIT bytevar
This command loads the printer-driver, the printer-driver must be on the current disk. If
this is not the case and you haven't included a respective error-routine in your program
then the program will crash! After the printer-driver has been loaded then the screen
display will be switched into the foreground (refer to scrbf !) and all the required
printer-information will be loaded into the variable bytevar. The variable bytevar must
be prepared beforehand in the Declarations-Section with the following parameters:
ROW 1920 BYTEVAR bytevar
The variable bytevar can of course have a different name. Eg. printer-buffer.

PRINTDONE
The memory-area $7900 is freed, the printer-driver is erased and the screen display is
switched back to the normal background modus.

STOPPRINT
Stops the printout and is followed by a FORMFEED (Paper throw-out).

2.5.2 Printing - Text
LPRINT string
The character-chain string will be printed, followed by a LINE-FEED. Control-Codes can
be sent to the printer in Hex-Format preceded by a fore-slash (Shift 7), it is possible to
send more than one Control-Code, if the last Control-Code a zero (0) then the following
Control-Code will not be accepted - 0 = string end.
Eg.: LPRINT"1b/78/01" - activates the LQ-Modus, in BASIC CHR27,103,1

LPRINT string;
The character-chain string will be printed, without a LINE-FEED. The following
character-chain will be then printed without a break (back-to-back). Control-Codes can
be sent to the printer in Hex-Format preceded by a fore-slash (Shift 7), it is possible to send
more than one Control-Code, if the last Control-Code a zero (0) then the following
Control-Code will not be accepted - 0 = string end.
Eg.: LPRINT"1b/78/01"; - activates the LQ-Modus, in BASIC CHR27,103,1

SETNLO
Switches from Draft-Modus in the NLQ-Modus, the relevant printer-parameters will be
taken from the current printer-driver.

STARTASCII
adjusts the line-spacing to the printer X-parameter and switches to ASC-II Modus
(Text-Modus): - New Page
The printout "order of battle" is:
PRINTINIT : Error-check at variable iostat.
STARTASCII
LPRINT .. LPRINT
STOPPRINT
PRINTDONE

2.5.3 Printing - Graphics
DIMX and DIMY
The parameters for Print-Width (max. 80 (X) characters at 8 points = 640 points) and
Print-Height (max. 90 (Y) characters at 8 points = 720 points) will be read from the
printer-driver. The parameters are accessed with x=(DIMX) and y=(DIMY).

HARDCOPY byte1,byte2,byte3,byte4
Gives a hard-copy of the screen:
byte1 = screen-start-line LH paper side.
byte2 = screen-start-line RH paper side.
byte3 = quantity of lines (at 8 points)
byte4 = variable name, is defined by : ROW 640 BYTEVAR in the Definitions-Section.
The screen is split into 25 lines (Commodore standard). You can print the fore- and
back-ground screens separately. The parameters for byte1 and byte2 are then:
Foreground 0 - 24 Background 25 - 49

- PAGE 50 -
If the parameter 128 stands in byte1 or byte2 then this will not be printed:
HARDCOPY0,128,25,byteName = Print the foreground-memory on the LH side.
If byte1 or byte2 are greater than 128 then this will be calculated as tab-space:
HARDCOPY0,(128+10),25,byteName = Print with a 10 character gap from the LH border.

PRINTBUF data
Prints the 640 byte graphic-data (ROW 640 BYTETVAR byte) in tiled-format.

STARTPRINT
Defines the line-spacing for the Graphic-Modus :- New Graphic Page.

2.6. Font - Commands

FONT object
This command activates the font and font-size that you have defined in object with ObjectEdit. the font has already been read and saved by ObjectEdit and must not be present on the current disk. You are not recommended to use "excessive" font sizes - this could lead to memory-space problems.

SYSFONT
De-activates all other fonts and switches back to the BSW-9 System-Font.

2.7 Graphic - Commands

The 40 Character (char.) Modus (C 64, C-128-40 char. modus) screen is divided into 320 (0-319) horizontal points and 200 (0-199) vertical points. The 80 char. C-128 (only Geos 128) screen has 640 (0-639) horizontal points. Graphic commands require that the x and y coordinates are set for the start and a possible end position: x,y to x1,y1. The LH upper corner has the parameter 0,0 and the RH lower corner 319,319 (ie. 639,199).

Colour-Codes are defined with the parameters: 0-39 and 0-24.

Due to the usual problems with memory-space (or rather lack of it !) there are particular rules that apply directly to the definition of the graphic-parameters x,y,x1 and y1. The parameters x and x1 are integer-variables and the parameters y and y1 must be entered as byte-variables ! ! As long as you don't define any variables as words and only as numbers then you shouldn't have any problems: LINE0,0,319,199, is OK but when you enter the variables: UP, LEFT,DOWN.RIGHT you are going to get a lot of trouble ! You must then define the variables LEFT and RIGHT as integer-variables INTVAR and the variables UP and DOWN as BYTETVAR in the Declarations-Section in your Source-Code and I bet that this will clash somewhere ! !

BITMAP byte,byte1,object
This command is used to display bitmaps (graphics) that you have previously tied in with ObjectEdit. The graphic will be displayed at the position that you have defined with the x-coordinate (byte) and y-coordinate (byte1), these two coordinates define position of the upper LH corner of the graphic. The x-coordinate is set in tiles (each 8 points) in the area from 0-39 (or 79 in 80 char. modus) and the y-coordinates are defined in points in the area 0-199. object is the name of the graphic and must be included in the Declarations-Section.
Eg.: BITMAP 10,100,arrow : defines the graphic "arrow" at the position (80,100)

CLRSCOL
Erases the current colour information, the colours return to the scrcol parameters or the initial settings of you Geos-System. This does not include the screen-border-colour, there are a number of DA's (Desk Accessories) that require a particular border-colour.

CLS
Erases the current screen and fills the screen with pattern no.2. If a "window" had previously defined with the command: WINDOW integer,byte, integer1, byte1 then this "window" will be erased.

CLS byte
Erases the current screen and fills the screen with pattern byte. If a "window" had previously defined with the command: WINDOW integer,byte, integer1,byte1 then only this "window" will be erased.
Eg. CLS 0
- GeoCom V 1.5 - Commands etc. -

**COLBOX** byte,byte 1,byte 2,byte 3,byte 4
This command fills a rectangular screen area with a pre-defined colour. Colour can only
be defined in 8x8 Pixel blocks, therefore you can only define 8x8 Pixel-format-areas.
byte,byte 1 is the LH upper corner, byte 2 the height in lines (0-25), byte 3 refers to
the width in tiles (0-40) and byte 4 is the colour, parallel to the sprite-colours.
Eg.: COLBOX5,2,10,10,4

**DBL** integer
This command doubles the x-coordinate parameter in tile-format.
Eg. SETPOS(DBL15),10

**DBL** integer,byte
This command doubles the x-coordinate parameter. It is possible using the variable byte
to shove the point around a bit. The following parameters are permitted:
byte = 1, 1 Pixel will be added
byte = 2, 1 Pixel will be subtracted
byte = 0 Normal
Eg.: SETPOS(DBL8,0),10 : Doubles the absolute parameter for the 80 char. modus, without
letting a single Pixel slip.

**FRAME** integer,byte, integer 1,byte 1
Draws a rectangular frame on the screen with the coordinates: x,y,x1,y1.
Eg. FRAME10,100,120,150

**FRAME** integer,byte, integer 1,byte 1,byte 2
Draws a rectangular frame on the screen with the coordinates: x,y,x1,y1 byte 2 is the
frame draw-pattern.
Eg.: FRAME10,100,120,150,0

**GET** byte,byte 1,byte 2,byte 3,byterow
Cuts the area byte, byte 1,byte 2,byte 3 out of the foreground-memory and lay's it in
the variable-area byterow. byte and byte 2 must be defined in tile-format (divide by 8).
byterow is calculated so:
((byte2 - byte ) divided by 8) multiplied by (byte3 - byte1)
and must be inserted in the Declarations-Section. This command is especially useful if a
screen-area must be moved, or to use the background-memory as variable-memory.
An example: Dialog-Box 40 char. modus - parameters:
byte = 64 / byte 1 = 32 / byte 2 = 255 + frame = 263 / byte 3 = 127 + frame = 135
263 - 64 = 199 divided by 8 = 25 (rounded)
135 - 320 = 104 (rounded)
25 x 104 = 2600 = byterow

**ICONS** object
Displays the pictogram object, that has been previously defined with ObjectEdit. An
automatic icon-interrogation-tie-in is only possible when the Geos-Mainloop has been
previously activated with the command MAINLOOP.

**INVERT** integer,byte, integer 1,byte 1
Inverts a rectangular area, with the coordinates x,y,x1,y1, on the screen. Defined points
are white, un-defined points are displayed black.
Eg. INVERT10,100,120,150

**IMPRINT** integer,byte, integer 1,byte 1
Moves an area from the foreground to the background. This command can only be used
when the background-memory is not being used as variable-memory.
Eg. IMPRINT20,50,100,100

**LINE** integer,byte, integer 1,byte 1
Draws a line with the start-position x,y to the end-point x1,y1 on the screen.
Eg. LINE 0,10,319,10

**PATTERN** byte
The pattern-fill No. for the command WINDOW,RECT,FRAME .. is defined by PATTERN
byte. This command must be defined before the draw-commands. Geo recognizes 34 fill-patterns (Pattern), of which geoPaint can show 32. Allowed are the parameters: 0-33.

POINT integer,byte
This command either sets or erases a point on the coordinate-point x,y. The pointmode is here very important.

PUT byte,byte,byte,byte,byte,byte row
Takes the area byte,byte,byte,byte,byte,byte out of the variable-area byte row and draws the area on the foreground-screen. See GET byte,byte,byte for more informations.

RECOVER integer,byte,byte,byte
Recover a DA to the screen. The background-memory (from $6000) may not be used as variable-memory. The variables integer - byte define the screen-area (x & y coordinates) that must be copied from the background to the foreground area (in a perfect world: 0,0,319,199), the screen colour may have to be re-defined. The command RECOVER..must follow the command GETFILE... (Geo returns here after exiting a DA).

RECT integer,byte,byte,byte
Draws a rectangle on the screen with the coordinates: x,y,x1,y1.
Eg. RECT 10,100,120,150

REGION integer,byte,byte,byte
Interrogates a with integer,byte,byte,byte defined rectangular screen-region, whether the mouse-pointer is "clicking" in the area. After the interrogation the program will receive a byte parameter as answer, is the parameter = 1 then there was a click, otherwise = 0. Here is it recommended before the interrogation to turn the interrupt off and afterwards back on! Have a look at "Pattern show" for a Demo. view.

SWITCH
Switches the screen-modus -> 40 in 80 and 80 in 40 char modus. Only for C 128!

TEST integer,byte
Checks whether the coordinate integer,byte a point sets or not. After the interrogation the program will receive a byte parameter as answer, is the parameter = 1 then there was a point, otherwise = 0.

WINDOW
Defines a "window" with the maximum parameters (40 char. = 0,0,319,199). All the following text-displays will appear within the "window".

WINDOW integer,byte,byte,byte
Defines a "window" with the parameters integer,byte to integer,byte. All the following text-displays will refer back to this "window".
Eg: WINDOW 10,10,309,189: defines a "window", that is 10 points smaller as the screen.

2.8 Sprite-, Mouse- and "The Same as ..." Commands
The mouse-pointer and the text-cursor are sprites, they occupy the sprite-positions 0 and 1. You may not re-define these positions. Sprites are created with ObjectEdit, it is possible to create sprites directly in the memory-area $d000-$dff. In this section can be seen all the commands that have anything to do with sprite-control.

2.8.1 Sprite - Commands
POSSPR byte,byte,byte
Sets the Sprite byte on the x-y coordinate byte,byte of the screen. The x-y coordinates are identical to the graphic-screen.

SETSPR byte,object
Sets the Sprite byte on the, with ObjectEdit, defined position of the object object.

SPRCOL byte,byte
Sprites can contain a maximum of 3 colours, 2 of which are the same for all sprites. The colours are defined with this command. The parameters are the same for the other
"colour-commands":
0 - black  1 - white  2 - red  3 - green  4 - purple  5 - dark green  6 - blue
7 - yellow  8 - orange  9 - brown  10 - pink  11 - dark grey  12 - grey
13 - light green  14 - light blue  15 - light grey

SPRITEOFF byte
Switches the sprite-no. byte off.

SPRITEON byte
Switches the sprite-no. byte on.

2.8.2 Mouse - Text Cursor - Commands

DUddy
De-activates icons and switches them all off. A reaction following a "click" is no longer possible.

HOME
Positions the text-cursor in the upper LH corner of the text-window.

MOUSEOFF
Switches the mouse-pointer (Sprite 0) off.

MOUSEON
Switches the mouse-pointer (Sprite 0) on.

MOUSEWIN
The mouse-pointer is "restricted" to the whole visible screen.

MOUSEWIN integer,byte,integer 1,byte 1
The mouse-pointer is restricted to the area defined within the parameters, the pointer can only leave the area by "clicking". This is very useful when defining menu reactions.
Eg.: MOUSEWIN0,0,319,199: = command MOUSEWIN

PROMPTINIT byte
Initials and defines the text-cursor to the size byte. The standard size (Eg. font BSW) is 9. The text-cursor must be initialized before it is activated.
Eg.: PROMPTINIT9

PROMPTOFF
Switches the text-cursor off.

PROMPTON
Switches the text-cursor on.
Eg. After the cursor has been defined to a particular position.
This command is only required when the commands PRINT and / or INPUT are not being activated.

QUITINP
Switches the with SETINP allowed enter-format prematurely off with a simulated RETURN.

SETINP string,label
This command enables the simultaneous access to mouse and keyboard. The pre-selected parameters of the variable string are valid and the program will jump to the label label after a RETURN confirmation.
Eg. The OK icon, it can be "clicked" and accessed with the RETURN key.

SETPOS integer,byte
Sets the text-cursor on the screen coordinate : integer,byte (x-y)

2.9 Mathematical Functions
(Translator's notes : I'm not a mathematician, some of the explanations may be a bit odd but I think the Demo's are alright!!!)
It is very important that negative real numbers are shown in brackets: (-.5)

2.9.1 Mathematical Commands

- real1 (real2)
  This operator gives the negative parameter (real2) of a defined number (real1).
  Eg. real= (-19.14) : real is then -10.34.

ABS real
  Gives the absolute whole-number parameter of a number. Is the number or parameter
  positive then the ABS function has no effect. Is the number or parameter negative then
  the function converts the num. / para. to it's positive equivalent.
  Eg. (ABS(-4.1)) is 4.1

AND
  Logical combination of two byte-variables. The following mathematical rules are binding:
  \[ 0 \cdot 0 = 0 \quad 0 \cdot 1 = 0 \quad 1 \cdot 0 = 0 \quad 1 \cdot 1 = 1 \]
  Eg. IF ((a<>b) AND (c==d)) THEN ....

ATN real
  This function calculates the tangent of the parameter real and gives the answer as
  real-number.
  Try: ATN "the angle with the tangent as (real)

COS real
  This function calculates the cosine of a real number and gives the answer as real-number.

EXP real
  The EXP function calculates the parameter of the constant e (circa 2.71828), that with
  the argument of the function is being given a potential calculation!
  Eg. (EXP 2) gives the parameter 7.3890 or 2.71828 x 2.71828

EXOR
  Logical combination of two byte-variables. The following mathematical rules are binding:
  \[ 0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 0 = 1 \quad 1 \oplus 1 = 0 \]

FLOOR real
  This function removes the positions after the decimal point:
  Eg. x=(FLOOR115.01) : x=>115

LOG real
  Calculates the logarithm at base 10 and gives the answer as a real-number:
  Eg. LOG (100) = parameter 2.

MOD
  Gives the left over of divisions calculation and defines this as the MOD byte.
  Eg. 11 / 4 = 2 , left over = 3 is then to be found in the MOD byte.

NOT
  Logical combination of byte-variables. The following mathematical rules are binding:
  \[ 0 = 1 \quad 1 = 0 \]

OR
  Logical combination of two byte-variables. The following mathematical rules are binding:
  \[ 0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 0 = 1 \quad 1 \oplus 1 = 1 \]
  Eg. IF ((a<>b) OR (a<>-1)) THEN ....

SIN real
  This function calculates the Sine of the parameter real and gives the answer as a
  real-number.
  Eg. x=SIN(4) , x=-0.7566802495

- PAGE 55 -
SGN real
This function will give you a byte-parameter when:
real > 0 is = 1; real = 0 is 0 or real < 0 is = 2

SQR real
The SQR function calculates the square root of the parameter real an gives the answer as a real-number.
Eg. (SQR (16)) = 4

TAN real
The SQR function calculates the tangent of the parameter real an gives the answer as a real-number.
Eg. y=(TAN2) y=> -2.18503986

2.9.2 Mathematical Equations
geoCom can carry out many different forms of calculations (more than me ! ) from addition to square roots and much more ! Take care when working with equations Eg. x=x+1 must be written so that the RH side stands in commata : x=(x+1). Further an integer-variable + byte-variable calculation is not possible, one of the variables must be converted beforehand.

2.10 Music- and Sound-Commands
One of the most interesting specialities of the Commodore 64 and 128's are the extensive options that these, relative, small computers offer. With practice and experience it is possible to imitate a large number of musical instruments, GeoCom allows you to access the computer SID chip (GeoBasic : VOICE, SOUND) by first leaving the Geo environment and paying a visit in the BASIC environment, the commands are not directly available in geoCom. GeoCom accesses the BASIC environment through the commands INITIO and DONEIO, the parameters are "Poked" and Peeked" to the memory-position $54272.
All the programmer has to do, is using INITIO and DONEIO to tell geoCom that you are accessing the "old" Commodore-System ($d000-$dff). If you have never programmed Commodore-Sound-Commands before you will have to consult a Commodore-SID-Handbook!
3. Description Of The Geos System
3.1 Memory Occupation
3.1.1 The Basics
In the computer world numbers are usually entered in one of three formats, Decimal, Hexa-Decimal or Binary. The Decimal-Format is possibly the easiest format for the human brain - the structure is simple and based on the number 10, therefore there are 10 digits (0-9). The computer can only internally work with two states - ON or OFF - therefore the "natural" numeric structure for a computer is even simpler and based on only 2 digits (0 or 1), this is known as the Binary-Format and is pre-fixed with the character %. The Binary-Format is slow, difficult to read and binary numbers take up a lot of space, because of this and various other reasons the Hexa-Decimal-Format was developed. The Hexa-Decimal-Format (will be referred to as HEX.) works with a 16 characters (0-9 and a-f), HEX. numbers combine 4 numbers from the Binary-System to one HEX. number. HEX. numbers are pre-fixed with the $ character.

Numbers are stored in a computer in Binary-Format but can be displayed (visual/sound/print) as HEX. or Decimal numbers. The computer always groups 8 happenings (On / OFF each 1 bit) together - this is a byte, the bits in a byte are numbered from left to right - 0-7. In a byte it is possible to show all the decimal numbers from 0-255, obviously this will not always be sufficient therefore a number will then be displayed using 2 bytes. This enables an area 0-655353 or -32768 to 32768 to be displayed, the combination of 2 bytes is known as a word.

The computer processor splits the available memory, which includes many bits into bytes. To enable the processor to find the individual bytes, each byte becomes an individual address, these addresses are given a definitive number. The number will be known as a word, therefore 65535 bytes can be addressed, this in turn gives a 64 KB large memory (65535/1024). A word that has been saved in the memory and displays the address of another memory-position is known as a pointer.

<table>
<thead>
<tr>
<th>BYTE</th>
<th>% 10110101</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% 1011</td>
</tr>
<tr>
<td></td>
<td>% 0101</td>
</tr>
<tr>
<td>High-Nibble</td>
<td>Low-Nibble</td>
</tr>
</tbody>
</table>

Memory-positions or -addresses are shown as 2 bytes because 1 byte can accept the maximum value of 255 ($ff). Specialists also define these two bytes as a word, a word consists of a low- and a high-byte.

<table>
<thead>
<tr>
<th>Adresse</th>
<th>$c0f0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$c0</td>
</tr>
<tr>
<td>High-Byte</td>
<td>$f0</td>
</tr>
<tr>
<td></td>
<td>$f0</td>
</tr>
<tr>
<td>Low-Byte</td>
<td>$c0</td>
</tr>
</tbody>
</table>

The low-byte is always displayed first and the low-byte always comes after the low-byte, therefore the low-byte will always be given in the first address-byte and the high-byte in second address-byte (). The complete address can be calculated with:

Highbyte × 256 + Lowbyte

The parameters are stored in the memory-positions as bytes, ie max. 255 ($ff).

3.1.2 Different Versions ?
Although there are various Geos versions available (V1.0, V1.2, V1.3, V1.5, V2.0) we haven’t been able to find any great differences in the memory-positions between the various versions V1.3 to V2.0. Your own program must therefore work together with the various versions. Memory differences between the the national versions, US and German version, are few and far between and are mainly concerned with the keyboard definitions.

- PAGE 58 -
3.1.3 Fundamental Memory Occupation

GeoCom allows various positions within the memory to be read and or re-written with new parameters. The geoCom programming system only requires to access the memory-positions that are listed here. GeoCom divides the memory according to the following structure. The geoCom memory-occupation (while your programs are running) is also displayed.

$0000 - Zero Page, Stack . . .
$0400 - geoCom - standard routines
$2800 - The routines of your geoCom program(s).
$4000 - The constants from your geoCom program(s).
$5000 - The variables from your geoCom program(s).
$6000 - Memory for the background screen.
$8000 - Geo-System variables.
$9000 - Drive drivers and other Geo-routines.
$a000 - Memory for the foreground screen.
$c000 - Geo-System routines.
$ffff

GeoCom allows, using the System-Variables, a direct access to selected memory-positions using defined-labels. The HEX address will always be given as the first item in the following description of the individual memory-positions, if the address includes more than 1 byte with a definite function then the start- and end-address will be given. Berkeley Softworks, the developers of GeoCom, have given all the "important" memory-addresses a name. This is very useful when programming in Assembler, the names are given directly behind the respective memory-positions. If the memory-position has been assigned a geoCom System-Variable then this is displayed on the RH margin, if the System-Variable is displayed with a * then this is an External-System-Variable that must be tied in using an extra function. The memory-positions are shown in ascending order.

3.1.4 Zero Page

$0022-$0023 curPattern
Points to the current fill-pattern, must be stored as 8 bytes. The respective bits show the points of the 8 graphic-lines of the pattern.

$002e currentMode
Text-modus byte. Has the byte been set (=1) then the style is activated. Multiple-combinations are permitted! ie the byte = 0 then the modus is not activated, = normal (plain) text.

Bit 0  not used
Bit 1  subscript (high)
Bit 2  superscript (low)
Bit 3  outline
Bit 4  italic
Bit 5  invert
Bit 6  bold
Bit 7  underline

Bit 1 and 2 find are only available in geoWrite.

$002f dispBufferOn
The byte defines, whether a text or graphic display should be written in the back- and / or fore-ground.

Bit 6  background
Bit 7  foreground

$0030 mouseOn
Multiple-byte for an active mouse-pointer and menu's, icon(s) active or not.

* mouseflag

PAGE 59
GeoCom V1.5 - Memory etc.

Bit 5 = Icon(a) active
Bit 6 = Menu(s) active
Bit 7 = Mouse-pointer on

$0031-$0032  mseaPicPtr
This Word is a pointer for the mouse-pointer graphic-data, the address $84c1 can
normally be found at this position.

$0033  topMargin
Y-Coordinate for the upper border for text-displays. Must lie between 0 and 199!

$0034  bottomMargin
Y-coordinate for the lower border for text-displays. Must lie between 0 and 199!

$0035-$0036  leftMargin
X-coordinate for the LH border for text-displays. If the text-display includes a CR, then
the following text will again start at the LH border. Must lie between 0 and 319 (639
when C128-80 char. modus)!

$0037-$0038  rightMargin
X-coordinate for the RH border for text-displays. The RH collision can be controlled by
a routine, that points at the StringFaultVector. Must lie between 0 and 319 (639 when
C128-80 char. modus)!

$0039  pressFlag
Multiple-byte for Press-Key, Input-driver and Fire-(Mouse)-Button
Bit 5 = Input-Driver fire-button has been pressed
Bit 6 = Input-Driver has been changed
Bit 7 = A different key has been pressed

$003a-$003b  mouseXPos
Shows in low- and high-byte format the x-coordinate of the mouse-pointer on the screen
(not the true position).

$003c-$003d  mouseYPos
Shows the y-coordinate of the mouse-pointer on the screen (not the true position).

$003c  graphMode
This byte shows, with it’s bit 7, under Geo 128 the current screen-modus. Is the bit set,
then is the 80 char. modus active.

3.1.5 GEOS variables
$8000-$80ff  diskBlkBuf
General buffer for disk. blocks. Is the same size as a disk. sector = 256 Bytes.

$8100-$81ff  fileHeader
The File-info. Block is stored here, is a VLIR-file open, then the File-Index-Block will be
stored here. Refer to chapter 4.4!

$8200-$82ff  curDirHead
The current disk. BAM is stored here. Larger drives (1571/1581) also use from $8900 a
second area (curDirHead).

$8300-$83ff  fileTrScTab
This area is used to store the Spur/Sector-Sequence of .SEQ files and / or
data-sequences.

$8400-$841d  dirEntryBuf
dirent
30 byte large block is used to store the file directory information. Refer to chapter 4.2!

$841e-$842f  DrACurDkNm
Here can be found the disk.name from drive A. Is the name shorter than 18 bytes, then
the remainder will be filled with $a (160), therefore this area cannot be read directly as
a string - the end-label 0 is missing! Drive A is always Drive-No. 8!
$8430-$8441  DrBCurDkNm
As DrACurDkNm, only for drive B, Drive-No., 9

$8442-$8452  dataFileName  docname
Here can be found the name of data-file + 1 zero-byte. Eg. When you click on a geoWrite
docu., geoWrite will first be loaded - geoWrite then looks at this address to see which
file was specified.

$8453-$8464  dataDiskName
Here can be found the disk. name to the above mentioned data-file. 16 bytes + 2 bytes
$A are stored. Cannot be directly accessed.

$8465-$8475  PrntFilename
The name of the current printer-driver is stored here, 16 bytes + 1 zero-byte.

$8489  curDrive  curDrive
Current Drive-No., always 8 or 9 while Geos programs can only be started from either 8
or 9 (DeskTop).

$848a  diskOpenFlag
If $848a = $ff then current disk. / drive = OK.
If $848a = $00 then current Disk. / drive = ERROR

$848e-$8491  driveType  drtype
These 4 consecutive bytes include the current drive-types of the drives A-D
Bit 0 = no drive
Bit 1 = 1541,1541c,1541 II
Bit 2 = 1571
Bit 3 = 1581
Bit 4 = not occupied
Bit 5 = not occupied
Bit 6 = shadowed drive
Bit 7 = RAM-drive
Bit 6 and 7 never together.

$8496  curRecord
Here can be found the no. of the current data-sequence of an open VLIR-file.
No data-sequence available = $ff, otherwise = 1

$8497  usedRecord
Here can be found, the number of data-sequences within the current VLIR-file.

$849b-$849c  appMain
A Word that applies to a routine that should be accessed by every run-through of the
Mainloop. Is the parameter = 0, then NO (0) routines will be accessed.

$84a1-$84a2  mouseVector
Points to a (self-written) routine that should be jumped to when the fire-button
(mouse/joystick ("click")) is activated.

$84a3-$84a4  keyVector
Points to a (self-written) routine that should be jumped to when a key is pressed. In
geoCom this function can be accessed with the command "ON 0 GOTO label ".

$84a5-$84a6  inputVector
Points to a routine when the bit 6 of the pressFlag is set. The pointer is usually set to
$0000, therefore a routine will not be accessed.

$84a7-$84a8  mouseFaultVector
When the mouse-pointer is outside a (displayed) menu or another pre-defined area, then
the program will jump to a routine that is stored at this position.

$94a9-$94aa  otherPressVector
If the fire-button is activated outside a (displayed) menu or another pre-defined area,
then the program will jump to a routine that is stored at this position. In geoCom this
function can be accessed with the command "ON 1 GOTO label ".

- PAGE 61 -
$84ab$-$84ac$

StringFaultVector
If a character is written beyond the RH border (defined in rightMargin), then the contents of the address will be used as a pointer to a (self programmed) routine. In geoCom this function can be accessed with the command "ON 2 GOTO label ".

$84ad$-$84ae$

alarmTmtVector
When the alarm clock sets off an alarm, then the program will jump to this address. The standard parameter is $0000$ but can be changed and / or accessed by an application.

$84af$

BRKVector
This routine will be jumped when Geos comes across the Assembler-Command brk. Geos will then display the error-message : "System error at $..... In geoCom this function can be accessed with the command "ON 4 GOTO label ".

$84b1$-$84b2$

RecoverVector
Points to a routine that rebuilds the background when the background has been damaged after a menu(s) or dialog-box(es) has been cancelled. In geoCom this function can be accessed with the command "ON 3 GOTO label ".

$84b3$

selectionFlash
Blink speed for menu-objects and icons. The standard parameter is = $0a$ (10).

$84b4$

alphaFlag
Blink speed for the text-cursor. The standard parameter is = $00$, because the cursor should not blink within the destTop.
Bit 0-5 = Time lapse before the cursor blinks
Bit 6 = Cursor visible
Bit 7 = Here you can only enter alpha-numeric characters (no control characters (Eg:C=)).

$84b5$

iconSelFlag
Bit 6 = The icon should not be inverted after being "clicked-on"
Bit 7 = The icon should be inverted after being "clicked-on"
One or the other ! Bit 7 has priority ! Bit 7 is the standard parameter !

$84b6$

faultData
When the mouse-pointer leaves a with : mouseTop,mouseBottom,mouseRight or mouseLeft defined parameter (border), then this parameter (faultdata) will be activated.
Bit 3 = The mouse-pointer is no longer within the current menu.
Bit 4 = The mouse-pointer has "collided" with the RH border-parameter.
Bit 5 = The mouse-pointer has "collided" with the LH border-parameter.
Bit 6 = The mouse-pointer has "collided" with the lower border-parameter.
Bit 7 = The mouse-pointer has "collided" with the upper border-parameter.

$84b7$

menuNumber
The no. of the current menu. The first menu object has the parameter zero (0).

$84b8$

mouseTop
Maximal upper mouse-pointer-position. Standard parameter = 0, Max = 199
Refer also to : faultData

$84b9$

mouseBottom
Maximal lower mouse-pointer-position. Standard parameter = 199
Refer also to : faultData

$84ba$-$84bb$

mouseLeft
Maximal LH mouse-pointer-position. Standard parameter = 0
Refer also to : faultData

$84bc$-$84bd$

mouseRight
Maximal RH mouse-pointer-position.
Refer also to : faultData

- PAGE 62 -
The mouse-pointer sprite is stored in this, 64 byte large, area. Sprites are 63 bytes large, therefore the 64th byte = 0. The sprite is 24x21 pixels, x-direction = always 8 pixels = 1 byte.

The highest speed of the mouse-pointer is stored at this position. Standard parameter = $7f = 127

The lowest speed of the mouse-pointer is stored at this position. Standard parameter = $1e = 30

The acceleration ratio of the mouse-pointer is stored at this position. Standard parameter = $7f = 127

ASCII-parameter for the last-pressed key, that hasn't been called upon by another routine.

Has the fire-button been pressed, then is the parameter = $ff, otherwise = $00

Registers the movement of the Input-Device:
0 = right
1 = upper-right
2 = up
3 = upper-left
5 = left-down
6 = down
7 = right-down
255 = no movement registered

Current mouse-speed

Lucky number, is re-defined after each interrupt.

Last two digits of the current year.

Current month

Current day

Current hour(s)

Current minute(s)

Current second(s)

Parameter = $ff, when the defined alarm-time is NOW.
Parameter = $00, when the defined alarm-time has not been arrived at.

Bit 0 - 3 = current foreground colour
Bit 4 - 7 = current background colour
ramExpSize
Parameter = size of the current REU in 64 kb blocks.

sysRamFlag
is only relevant by a connected REU:
Bit 4 = the areas $7e00-82ff and $b900-fc3f are set so that after a re-boot from the
System-Disk, the Geo-Kernel is immediately available.
Bit 5 = the area $7900-7dff will be loaded by the area $8400-88ff when Geo is exited to
Basic.
Bit 6 = the area $3000-b8ff will be loaded by the disk-drivers from the drives A-C.
Bit 7 = the area $0000-78ff will be used by the MoveData-routine for the high speed
memory-block transfer function.

firstBoot
After the initial SystemStart the parameter $ff can be found here, otherwise = $00.

curType
Relevant information about the current drive.
Copy of driveType!

ramBase
If a shadowed drive is included in the current System, then this parameter indicates in
each individual address which banks in the REU are occupied by the RAM-disk or
shadowed disk.
Available for drive A ($88c7) to drive D ($88ca).

DrCCurDkName
Same as DrACurDkName for Drive A ! For drive C.

DrDCurDkName
Same as DrACurDkName for Drive A ! For drive D.

dir2Head
Second BAM-block for 1571/81 drives - size = 256 bytes.

spr0pic
64 byte large area for sprite 0. Sprite = (63) Bytes + zero (0) byte.

spr1pic
64 byte large area for sprite 1. Sprite = (63) Bytes + zero (0) byte.

Attention! The sprites 0 and 1 are already occupied by Geo, sprite 0 is the
mouse-pointer and sprite 1 is the text-cursor.

spr2pic
64 byte large area for sprite 2. Sprite = (63) Bytes + zero (0) byte.

spr3pic
64 byte large area for sprite 3. Sprite = (63) Bytes + zero (0) byte.

spr4pic
64 byte large area for sprite 4. Sprite = (63) Bytes + zero (0) byte.

spr5pic
64 byte large area for sprite 5. Sprite = (63) Bytes + zero (0) byte.

spr6pic
64 byte large area for sprite 6. Sprite = (63) Bytes + zero (0) byte.

spr7pic
64 byte large area for sprite 7. Sprite = (63) Bytes + zero (0) byte.
3.1.6 Further Important Memory Positions

$\text{\textcolor{red}{\textit{Sc00f}}} \quad \text{version} \\
\text{Geos System Version-No.} \\
\text{Version V1.2 = $12$} \\
\text{Version V1.3 = $13$} \\
\text{Version V2.0 = $20$}

$\text{\textcolor{red}{\textit{Sc01o}}} \quad \text{\textit{nationally}} \\
\text{Nationality Label : By Geos V1.2 = $00$, otherwise (German) = $01$.}

$\text{\textcolor{red}{\textit{Sc013}}} \quad \text{\textit{c128Flag}} \\
\text{Computer-type-Flag : By C64 = $00$, By C128 = $80$. First from version V1.3!}

$\text{\textcolor{red}{\textit{D000-\textit{dfff}}}}$ \\
The memory-area $\text{\textcolor{red}{\textit{D000-\textit{dfff}}}}$ includes the old Commodore-System, sprite-interrogation, and the switch-on-message etc. This area must first be activated with \textit{INITIO}, after you have finished accessing this area you must de-activate it with \textit{DONEIO}. Further information that relates to this area can be found in most C64/128 Handbooks.

3.1.7 The Screen Memory

When working with the C64 and 128 computers we must always bear in mind that we have two distinct screen memory formats: Geos 128 supports both the 40 and 80 character screen modes. The two formats have many differences because they are addressed by two different processors.

When using the 40 char. display format the screen has a 320 x 200 dot matrix, in order to save this screen require 8000 bytes - there are 64000 picture dots and in a single byte it is possible to store 8 picture dots. This memory area is reserved within the main memory area and starts at $\text{\textcolor{red}{\textit{D000}}}$. At $\text{\textcolor{red}{\textit{D000}}}$ a copy of GEOS controls the Dialog-Box and Menu requirements, this area has the same size and structure as that at $\text{\textcolor{red}{\textit{D000}}}$. Many applications require this area for an additional variable buffer, these applications make a note of the changes and restore the area to its previous status after the function has been completed. This doesn't usually require much memory because the Dialog-Boxes and Menus don't normally occupy the whole screen area, in order to access this function \textit{geoCom} uses the commands: \textit{GETbyte1,byte2,byte3,byte4,byte4,byte4,byte4,byte4,byte4,byte4} and \textit{PUTbyte1,byte2,byte3,byte4,byte4,byte4,byte4,byte4,byte4,byte4}. The 40 char. screen memory is internally divided into a grid tiled pattern - 8 bytes are equivalent to a tile. This tiled pattern of 8 * 8 points appear adjacent to each other on the screen - a new row is started when the previous row has been filled.

The 80 char. screen is stored in a different format, it is twice the size (640 * 200) of the 40 char. screen and therefore occupies twice as much memory space. This is not taken from the main memory, a separate screen memory is provided. This can only be accessed (In & Out) using two memory positions - for further information, please read the relevant handbooks. The 80 char. screen saves 8 dots - from one line - in a single byte, therefore a line is equivalent to 80 bytes in the screen memory. The screen memory is split into two equal parts and copied to the main memory at $\text{\textcolor{red}{\textit{D000}}} \text{and $\text{\textcolor{red}{\textit{D000}}} - where the 40 char. screen would normally be located. The background-screen-memory can be used as a variable memory by using the same commands as in the 40 char. modus.}

3.2 Codes

Codes are used in GEOS to define a key or a character to one or more of the computer-internal-operated characters. For the purposes of entering (input) information there are keyboard - and Input - Codes, these define for each key or key combination a particular code, for the display and output functions there are the so-called Output Codes. This means that each character that appears on the screen or is sent to the printer has it's own (code) number.

3.2.1 Keyboard - Codes - Geos ASCII ($\text{\textcolor{red}{\textit{01-3f}}}$)

By accessing the memory position $\text{\textcolor{red}{\textit{8504}}}$ (or keydata) you will receive the ASCII-parameter of the last-pressed key(s), thus enabling you - using geoCom - to access
the keyboard at any time. Using the same option it is also possible to access the
last-pressed key-combination (e.g. C=q) and thus - using the label command - to branch
off into a further program section. (If the Commodore-Key is held down, then the
parameter will be incremented by the factor of 128).

<table>
<thead>
<tr>
<th>Key</th>
<th>Hex</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>f1</td>
<td>$01</td>
<td>1</td>
</tr>
<tr>
<td>F2</td>
<td>$02</td>
<td>2</td>
</tr>
<tr>
<td>f3</td>
<td>$03</td>
<td>3</td>
</tr>
<tr>
<td>F4</td>
<td>$04</td>
<td>4</td>
</tr>
<tr>
<td>f5</td>
<td>$05</td>
<td>5</td>
</tr>
<tr>
<td>F5</td>
<td>$06</td>
<td>6</td>
</tr>
<tr>
<td>no scroll</td>
<td>$07</td>
<td>7</td>
</tr>
<tr>
<td>Cursor left</td>
<td>$08</td>
<td>8</td>
</tr>
<tr>
<td>tab</td>
<td>$09</td>
<td>9</td>
</tr>
<tr>
<td>LINE FEED</td>
<td>$0a</td>
<td>10</td>
</tr>
<tr>
<td>Enter</td>
<td>$0b</td>
<td>11</td>
</tr>
<tr>
<td>No Geos</td>
<td>$0c</td>
<td>12</td>
</tr>
<tr>
<td>RETURN</td>
<td>$0d</td>
<td>13</td>
</tr>
<tr>
<td>f7</td>
<td>$0e</td>
<td>14</td>
</tr>
<tr>
<td>F8</td>
<td>$0f</td>
<td>15</td>
</tr>
<tr>
<td>Cursor Up</td>
<td>$10</td>
<td>16</td>
</tr>
<tr>
<td>Cursor Down</td>
<td>$11</td>
<td>17</td>
</tr>
<tr>
<td>HOME</td>
<td>$12</td>
<td>18</td>
</tr>
<tr>
<td>CLR</td>
<td>$13</td>
<td>19</td>
</tr>
<tr>
<td>Back</td>
<td>$14</td>
<td>20</td>
</tr>
<tr>
<td>No Geos</td>
<td>$15</td>
<td>21</td>
</tr>
<tr>
<td>STOP</td>
<td>$16</td>
<td>22</td>
</tr>
<tr>
<td>RUN</td>
<td>$17</td>
<td>23</td>
</tr>
<tr>
<td>Brit.Pound</td>
<td>$18</td>
<td>24</td>
</tr>
<tr>
<td>HELP</td>
<td>$19</td>
<td>25</td>
</tr>
<tr>
<td>OLD</td>
<td>$1a</td>
<td>26</td>
</tr>
<tr>
<td>ESC</td>
<td>$1b</td>
<td>27</td>
</tr>
<tr>
<td>INS</td>
<td>$1c</td>
<td>28</td>
</tr>
<tr>
<td>DEL</td>
<td>$1d</td>
<td>29</td>
</tr>
<tr>
<td>Cursor right</td>
<td>$1e</td>
<td>30</td>
</tr>
<tr>
<td>SHIFT/CTRL</td>
<td>$1f</td>
<td>31</td>
</tr>
<tr>
<td>SPACE</td>
<td>$20</td>
<td>32</td>
</tr>
<tr>
<td>!</td>
<td>$21</td>
<td>33</td>
</tr>
<tr>
<td>*</td>
<td>$22</td>
<td>34</td>
</tr>
<tr>
<td>#</td>
<td>$23</td>
<td>35</td>
</tr>
<tr>
<td>$</td>
<td>$24</td>
<td>36</td>
</tr>
<tr>
<td>%</td>
<td>$25</td>
<td>37</td>
</tr>
<tr>
<td>&amp;</td>
<td>$26</td>
<td>38</td>
</tr>
<tr>
<td>(</td>
<td>$27</td>
<td>39</td>
</tr>
<tr>
<td>)</td>
<td>$28</td>
<td>40</td>
</tr>
<tr>
<td>+</td>
<td>$29</td>
<td>41</td>
</tr>
<tr>
<td>-</td>
<td>$2a</td>
<td>42</td>
</tr>
<tr>
<td>.</td>
<td>$2b</td>
<td>43</td>
</tr>
<tr>
<td>,</td>
<td>$2c</td>
<td>44</td>
</tr>
<tr>
<td>;</td>
<td>$2d</td>
<td>45</td>
</tr>
<tr>
<td>:</td>
<td>$2e</td>
<td>46</td>
</tr>
<tr>
<td>/</td>
<td>$2f</td>
<td>47</td>
</tr>
<tr>
<td>0</td>
<td>$30</td>
<td>48</td>
</tr>
<tr>
<td>1</td>
<td>$31</td>
<td>49</td>
</tr>
<tr>
<td>2</td>
<td>$32</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>$33</td>
<td>51</td>
</tr>
<tr>
<td>4</td>
<td>$34</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>$35</td>
<td>53</td>
</tr>
<tr>
<td>6</td>
<td>$36</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>$37</td>
<td>55</td>
</tr>
<tr>
<td>8</td>
<td>$38</td>
<td>56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Hex</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$41</td>
<td>65</td>
</tr>
<tr>
<td>B</td>
<td>$42</td>
<td>66</td>
</tr>
<tr>
<td>C</td>
<td>$43</td>
<td>67</td>
</tr>
<tr>
<td>D</td>
<td>$44</td>
<td>68</td>
</tr>
<tr>
<td>E</td>
<td>$45</td>
<td>69</td>
</tr>
<tr>
<td>F</td>
<td>$46</td>
<td>70</td>
</tr>
<tr>
<td>G</td>
<td>$47</td>
<td>71</td>
</tr>
<tr>
<td>H</td>
<td>$48</td>
<td>72</td>
</tr>
<tr>
<td>I</td>
<td>$49</td>
<td>73</td>
</tr>
<tr>
<td>J</td>
<td>$4a</td>
<td>74</td>
</tr>
<tr>
<td>K</td>
<td>$4b</td>
<td>75</td>
</tr>
<tr>
<td>L</td>
<td>$4c</td>
<td>76</td>
</tr>
<tr>
<td>M</td>
<td>$4d</td>
<td>77</td>
</tr>
<tr>
<td>N</td>
<td>$4e</td>
<td>78</td>
</tr>
<tr>
<td>O</td>
<td>$4f</td>
<td>79</td>
</tr>
<tr>
<td>P</td>
<td>$50</td>
<td>80</td>
</tr>
<tr>
<td>Q</td>
<td>$51</td>
<td>81</td>
</tr>
<tr>
<td>R</td>
<td>$52</td>
<td>82</td>
</tr>
<tr>
<td>S</td>
<td>$53</td>
<td>83</td>
</tr>
<tr>
<td>T</td>
<td>$54</td>
<td>84</td>
</tr>
<tr>
<td>U</td>
<td>$55</td>
<td>85</td>
</tr>
<tr>
<td>V</td>
<td>$56</td>
<td>86</td>
</tr>
<tr>
<td>W</td>
<td>$57</td>
<td>87</td>
</tr>
<tr>
<td>X</td>
<td>$58</td>
<td>88</td>
</tr>
<tr>
<td>Y</td>
<td>$59</td>
<td>89</td>
</tr>
<tr>
<td>Z</td>
<td>$5a</td>
<td>90</td>
</tr>
<tr>
<td>[</td>
<td>$5b</td>
<td>91</td>
</tr>
<tr>
<td>]</td>
<td>$5c</td>
<td>92</td>
</tr>
<tr>
<td>^</td>
<td>$5d</td>
<td>93</td>
</tr>
<tr>
<td>_</td>
<td>$5e</td>
<td>94</td>
</tr>
<tr>
<td>`</td>
<td>$5f</td>
<td>95</td>
</tr>
<tr>
<td>-</td>
<td>$60</td>
<td>96</td>
</tr>
<tr>
<td>a</td>
<td>$61</td>
<td>97</td>
</tr>
<tr>
<td>b</td>
<td>$62</td>
<td>98</td>
</tr>
<tr>
<td>c</td>
<td>$63</td>
<td>99</td>
</tr>
<tr>
<td>d</td>
<td>$64</td>
<td>100</td>
</tr>
<tr>
<td>e</td>
<td>$65</td>
<td>101</td>
</tr>
<tr>
<td>f</td>
<td>$66</td>
<td>102</td>
</tr>
<tr>
<td>g</td>
<td>$67</td>
<td>103</td>
</tr>
<tr>
<td>h</td>
<td>$68</td>
<td>104</td>
</tr>
<tr>
<td>i</td>
<td>$69</td>
<td>105</td>
</tr>
<tr>
<td>j</td>
<td>$6a</td>
<td>106</td>
</tr>
<tr>
<td>k</td>
<td>$6b</td>
<td>107</td>
</tr>
<tr>
<td>l</td>
<td>$6c</td>
<td>108</td>
</tr>
<tr>
<td>m</td>
<td>$6d</td>
<td>109</td>
</tr>
<tr>
<td>n</td>
<td>$6e</td>
<td>110</td>
</tr>
<tr>
<td>o</td>
<td>$6f</td>
<td>111</td>
</tr>
<tr>
<td>p</td>
<td>$70</td>
<td>112</td>
</tr>
<tr>
<td>q</td>
<td>$71</td>
<td>113</td>
</tr>
<tr>
<td>r</td>
<td>$72</td>
<td>114</td>
</tr>
<tr>
<td>s</td>
<td>$73</td>
<td>115</td>
</tr>
<tr>
<td>t</td>
<td>$74</td>
<td>116</td>
</tr>
<tr>
<td>u</td>
<td>$75</td>
<td>117</td>
</tr>
<tr>
<td>v</td>
<td>$76</td>
<td>118</td>
</tr>
<tr>
<td>w</td>
<td>$77</td>
<td>119</td>
</tr>
<tr>
<td>x</td>
<td>$78</td>
<td>120</td>
</tr>
</tbody>
</table>
3.2.2 The Character (output) - Codes

The character - output - codes differ only from char. $00$ to $1f$. The codes are otherwise identical to the keyboard codes - therefore only the first codes (command characters) are shown here.

<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Definition</th>
<th>Hex</th>
<th>Dec</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$00$</td>
<td>0</td>
<td>Zero String end</td>
<td>$11$</td>
<td>17</td>
<td>ESC_Ruler</td>
</tr>
<tr>
<td>$01$</td>
<td>1</td>
<td>don't implement</td>
<td>$12$</td>
<td>18</td>
<td>REVON</td>
</tr>
<tr>
<td>$02$</td>
<td>2</td>
<td>don't implement</td>
<td>$13$</td>
<td>19</td>
<td>REVOFF</td>
</tr>
<tr>
<td>$03$</td>
<td>3</td>
<td>don't implement</td>
<td>$14$</td>
<td>20</td>
<td>GOTOX (2 Byte)</td>
</tr>
<tr>
<td>$04$</td>
<td>4</td>
<td>don't implement</td>
<td>$15$</td>
<td>21</td>
<td>GOTOY (1 Byte)</td>
</tr>
<tr>
<td>$05$</td>
<td>5</td>
<td>don't implement</td>
<td>$16$</td>
<td>22</td>
<td>GOTOXY (3 Byte)</td>
</tr>
<tr>
<td>$06$</td>
<td>6</td>
<td>don't implement</td>
<td>$17$</td>
<td>23</td>
<td>NEWCARDSET</td>
</tr>
<tr>
<td>$07$</td>
<td>7</td>
<td>don't implement</td>
<td>$18$</td>
<td>24</td>
<td>BOLDON</td>
</tr>
<tr>
<td>$08$</td>
<td>8</td>
<td>BACKSPACE</td>
<td>$19$</td>
<td>25</td>
<td>ITALICON</td>
</tr>
<tr>
<td>$09$</td>
<td>9</td>
<td>FORWARDSPACE/TAB</td>
<td>$1a$</td>
<td>26</td>
<td>OUTLINEON</td>
</tr>
<tr>
<td>$0a$</td>
<td>10</td>
<td>LF (CRSR down)</td>
<td>$1b$</td>
<td>27</td>
<td>PLAINTEXT</td>
</tr>
<tr>
<td>$0b$</td>
<td>11</td>
<td>HOME (x=0,y=0)</td>
<td>$1c$</td>
<td>28</td>
<td>don't implement</td>
</tr>
<tr>
<td>$0c$</td>
<td>12</td>
<td>UPLINE(CR SR up)</td>
<td>$1d$</td>
<td>29</td>
<td>don't implement</td>
</tr>
<tr>
<td>$0d$</td>
<td>13</td>
<td>CR (Carriage Return)</td>
<td>$1e$</td>
<td>30</td>
<td>don't implement</td>
</tr>
<tr>
<td>$0e$</td>
<td>14</td>
<td>ULINEON</td>
<td>$1f$</td>
<td>31</td>
<td>don't implement</td>
</tr>
<tr>
<td>$0f$</td>
<td>15</td>
<td>ULINEOFF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$10$</td>
<td>16</td>
<td>ESC_GRAPHICS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The further characters are identical to the Keyboard-Codes II.

3.3 Disk. Format

Geos can work together with the following Commodore disk, drives: 1541, 1570 (1541 emulation), 1571 and 1581, it is also possible to use the following RAM-Expansion Unit(s) (REU's): (GEORAM, REU 1746/1750, RAM-Link...) and therefore simulate the above mentioned drives. GEOS sticks fairly closely to the standard DOS-floppy-format - a few characters and options have been added. For the 1581 drives a simulated BAM is provided, the BAM (Block Availability MAP) includes information such as the disk-name, DISK.ID, and a list of the occupied and free blocks. The BAM is always at the first position in the disk. directory - 1241 and 1571, Track-18 Pos.0, 1581 Track-40 Pos.3.

3.3.1 BAM

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>Track-/Sector combinations; is shown at the first directory block</td>
</tr>
<tr>
<td></td>
<td>(1541/1571 -&gt; 18/1, 1581 -&gt; 40/3)</td>
</tr>
<tr>
<td>2</td>
<td>DOS-Format-label (1541/1571 -&gt; a ($41) - 1581 -&gt; d ($44)</td>
</tr>
<tr>
<td>3</td>
<td>Flag for double sided disk. (only 1571)</td>
</tr>
<tr>
<td>4-143</td>
<td>Block-occupation-table (check the Floppy-Handbook)</td>
</tr>
<tr>
<td>144-159</td>
<td>Filled with 160 (a0) Disk. name</td>
</tr>
<tr>
<td>160,161</td>
<td>$a0$,$a1$ not used ($a0$)</td>
</tr>
<tr>
<td>162-166</td>
<td>$a2$,$a6$ Disk-ID and Format ($2a$ for 1541,70,71 and $3d$ for 1581)</td>
</tr>
<tr>
<td>167-170</td>
<td>$a7$,$a9$ not used ($a0$)</td>
</tr>
<tr>
<td>171,172</td>
<td>$ab$,$ac$ Track/Sector for the Border block</td>
</tr>
<tr>
<td>173-188</td>
<td>$ad$,$bc$ Text &quot;GEOS format V.1.0&quot;</td>
</tr>
<tr>
<td>189</td>
<td>$bd$ Disk-Byte: $42$ -&gt; System disk., $50$ -&gt; Main disk., $00$ -&gt; Work disk.</td>
</tr>
<tr>
<td>190-220</td>
<td>$be$,$dc$ not used ($a0$)</td>
</tr>
<tr>
<td>221-255</td>
<td>$dd$,$ff$ only for 1571 Bloc-Occupations-Table, track 36 - 70</td>
</tr>
</tbody>
</table>
3.3.2 Directory

Directly following the BAM can be found the Disk Directory. It contains 8 file entries in each block. All Commodore drives can control a maximum of 144 file entries. The basic format of every block is as follows:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>$00,$01</td>
</tr>
<tr>
<td>2-31</td>
<td>$02-$1f</td>
</tr>
<tr>
<td>32</td>
<td>$20</td>
</tr>
<tr>
<td>33</td>
<td>$21</td>
</tr>
<tr>
<td>34-63</td>
<td>$22-$3f</td>
</tr>
<tr>
<td>64</td>
<td>$40</td>
</tr>
<tr>
<td>65</td>
<td>$41</td>
</tr>
<tr>
<td>66-95</td>
<td>$42-$5f</td>
</tr>
<tr>
<td>96</td>
<td>$60</td>
</tr>
<tr>
<td>97</td>
<td>$61</td>
</tr>
<tr>
<td>98-127</td>
<td>$62-$7f</td>
</tr>
<tr>
<td>128</td>
<td>$80</td>
</tr>
<tr>
<td>129</td>
<td>$81</td>
</tr>
<tr>
<td>130,159</td>
<td>$82-$9f</td>
</tr>
<tr>
<td>160</td>
<td>$a0</td>
</tr>
<tr>
<td>161</td>
<td>$a1</td>
</tr>
<tr>
<td>162-191</td>
<td>$a2-$bf</td>
</tr>
<tr>
<td>192</td>
<td>$c0</td>
</tr>
<tr>
<td>193</td>
<td>$c1</td>
</tr>
<tr>
<td>194-223</td>
<td>$c2-$df</td>
</tr>
<tr>
<td>224</td>
<td>$e0</td>
</tr>
<tr>
<td>225</td>
<td>$e1</td>
</tr>
<tr>
<td>226-255</td>
<td>$e2-$ff</td>
</tr>
</tbody>
</table>

3.3.3 A Directory-Block

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$00</td>
</tr>
<tr>
<td>1-2</td>
<td>$01,$02</td>
</tr>
<tr>
<td>3-18</td>
<td>$03-$12</td>
</tr>
<tr>
<td>19-20</td>
<td>$13,$14</td>
</tr>
<tr>
<td>21</td>
<td>$15</td>
</tr>
<tr>
<td>22</td>
<td>$16</td>
</tr>
<tr>
<td>23</td>
<td>$17</td>
</tr>
<tr>
<td>24</td>
<td>$18</td>
</tr>
<tr>
<td>25</td>
<td>$19</td>
</tr>
<tr>
<td>26</td>
<td>$1a</td>
</tr>
<tr>
<td>27</td>
<td>$1b</td>
</tr>
<tr>
<td>28,29</td>
<td>$1c,$1d</td>
</tr>
</tbody>
</table>

These parameters can also be directly accessed with the `direntry<byte>` command.

- PAGE 68 -
3.4 File Formats
3.4.1 Info. - Block
Here can be found the same file information that can be accessed from within the Desktop with the C=Q command. There is some other additional information that can only be accessed using the Diskmonitor:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$01 always $00,$ff (not a followon block)</td>
</tr>
<tr>
<td>2</td>
<td>$02      Icon width in Cards (eg. 24/8 = 3)</td>
</tr>
<tr>
<td>3</td>
<td>$03      Icon height : 21</td>
</tr>
<tr>
<td>4</td>
<td>$04      Icon header-byte : $bf (= 191 = 62 unpacked data units)</td>
</tr>
<tr>
<td>5-67</td>
<td>$05-$43 Icon-Bitmap (unpacked format)</td>
</tr>
<tr>
<td>68</td>
<td>$44      Commodore file-type (eg. $82 = PRG)</td>
</tr>
<tr>
<td>69</td>
<td>$45      Geos file-type</td>
</tr>
<tr>
<td>70</td>
<td>$46      File structure (0 = seq; 1 = VLIR)</td>
</tr>
<tr>
<td>71,72</td>
<td>$47,$48 Load address for Appli. / Desk acc. -&gt; Low,High</td>
</tr>
<tr>
<td>73,74</td>
<td>$49,$4a End address (Low, High)</td>
</tr>
<tr>
<td>75,76</td>
<td>$4b,$4c Start address (Low, High)</td>
</tr>
<tr>
<td>77-93</td>
<td>$4d-$5d Class :</td>
</tr>
<tr>
<td>94,95</td>
<td>$5e,$5f not used</td>
</tr>
<tr>
<td>96</td>
<td>$60      Screen flag for Geos 128</td>
</tr>
<tr>
<td></td>
<td>$00 = only 40 Char.-Modus</td>
</tr>
<tr>
<td></td>
<td>$40 = 40 and 80 Char.-Modi</td>
</tr>
<tr>
<td></td>
<td>$80 = only for Geos 64</td>
</tr>
<tr>
<td></td>
<td>$c0 = only 80 Char.-Modus</td>
</tr>
<tr>
<td>97-116</td>
<td>$61-$74 Author</td>
</tr>
<tr>
<td>117-133</td>
<td>$75-$85 Appli. Class, (that the file has created)</td>
</tr>
<tr>
<td>134-159</td>
<td>$86-$9f Free for Data, eg. as in geoWrite</td>
</tr>
<tr>
<td>160-255</td>
<td>$a0-$ff Info. text, finishes with a null-byte</td>
</tr>
<tr>
<td>77-88</td>
<td>$4d-$58 Permanent Doc. name, ie. doc. label</td>
</tr>
<tr>
<td>89-93</td>
<td>$59-$5d Version reference (V.x.x)</td>
</tr>
</tbody>
</table>

3.4.1.1 Sequential Files
This format was already available in the original C64/128 DOS. The track and sector position of the file start block are indicated in byte 1 and 2. The first two bytes of these blocks refer to the track and sector of the next - following - block, the other 254 bytes store the data. Is the 1st byte of a block = 0, then the 2nd byte indicates the position of the last character in the file. GEOS restricts the size of a sequential file to 1277 blocks (3258 bytes).

3.4.1.2 VLIR Files
This file structure has been specially developed for GEOS and has a different structure to the file formats recognized by Commodore 64/128 DOS. The first block of a VLIR file, indicated by byte 1 and 2 of the file entry, contains an index-block. This is structured, by means of 2 byte indicators, to show the start track and sector of the, up to, 127 data units, the individual units, are in turn, stored in sequential format. Byte 0 and 1 of the index - block always include the parameters $00 and $ff, the following byte pairs always indicate the start of data unit. The data units are consecutively numbered 0 - 126. If a byte combination where to have the parameter = $00, Sff then the data unit is available - but it is empty. If both bytes have the parameter = 0 then there are no following units in the file.

3.5 geoWrite Documents
3.5.1 Info. - Block:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$01 always $00,$ff (no following block)</td>
</tr>
<tr>
<td>2</td>
<td>$02      Icon width in cards eg. 24/8 = 3)</td>
</tr>
<tr>
<td>3</td>
<td>$03      Icon height : 21</td>
</tr>
<tr>
<td>4</td>
<td>$04      Icon header-byte</td>
</tr>
<tr>
<td>5-67</td>
<td>$05-$43 Icon-Bitmap (Format = unpacked)</td>
</tr>
<tr>
<td>68</td>
<td>$44      Commodore file type ($83)</td>
</tr>
<tr>
<td>69</td>
<td>$45      Geos file type (07 = Document)</td>
</tr>
<tr>
<td>70</td>
<td>$46      File structure (1 = VLIR)</td>
</tr>
<tr>
<td>71,72</td>
<td>$47,$48 Load address ($0000)</td>
</tr>
</tbody>
</table>
- GeoCom V1.5 - Disk. format etc. -

73.74 $49,34a End address ($fff)
75.76 $4b,34c Start address ($0000)
77.88 $4d-$58 Permanent File name (Write Image)
89-92 $59-$5c Version (V.x.x)
93-116 $5d-$74 filled with $00
117-132 $75-$84 Class: geoWrite...
133-136 $85-$88 filled with $00
137,138 $89-$8a from V2.0 - Page No.: 1st page in LOW-HIGH-Format, otherwise $01
139 $8b only from V2.0 - 64, is NLQ-spacing active, then = 128 is the function "Title Page" active then =, both parameters are possible
140,141 $8c,$8d only from V2.0 - Height of the page header (screen dots)
142,143 $8e,$8f only from V2.0 - Height of the page foot note (screen dots)
144,145 $90,$91 Page length, is dependent on the printer parameters
146-159 $94-$9f not used ->$00
160-255 $a0-$ff Info. text, finishes with a zero byte

3.5.2 Index - Sector:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$01 always $00,$ff - no following block</td>
</tr>
<tr>
<td>2,3</td>
<td>$02,$03 Data units 0 - 60 = Text pages 0 - 60, if Data unit 0 = Page 1. Le. fixed &quot;1. Page No.&quot; = set page. Indicates TR/SE of the individual pages. Max. 119,120. Data unit = 60 = Page 61</td>
</tr>
<tr>
<td>122,123 $7a,$7b Page header track/sector (= Data unit 61)</td>
<td></td>
</tr>
<tr>
<td>124,125 $7c,$7d Page foot note track/sector (= Data unit 62)</td>
<td></td>
</tr>
<tr>
<td>126,127 $7e,$7f not used</td>
<td></td>
</tr>
<tr>
<td>128,129 $80,$81 Data units 64 - 127 = Photo scraps 0 - 64, whereby Data unit 64 = byte 128, 129 = Scrap No.1 in packed format to 254, 255 = Data unit 127</td>
<td></td>
</tr>
</tbody>
</table>

3.5.3 Data Unit Structure (0 - 60) - geoWrite V1.1

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$01 LH Border in screen dots (0-144, in 8x steps)</td>
</tr>
<tr>
<td>2,3</td>
<td>$02,$03 RH Border in screen dots (224-479, in 8x steps)</td>
</tr>
<tr>
<td>4-19</td>
<td>$04-$a3 Tabulators, 8x 2 Bytes. = not set = parameter the RH border</td>
</tr>
<tr>
<td>20-23 $a4-$a7 NEWCARDSET</td>
<td></td>
</tr>
<tr>
<td>24-End $a8-End Text in Geo-char. code with following exceptions: $01 - Page end $09 - Tab. Jump $0c - RETURN $10 - GRAPHIC-ESCAPE-commence $17 - NEWCARDSET-commence $00 - Document-End</td>
<td></td>
</tr>
</tbody>
</table>

The Char. and Command Codes are interchangable from 31 onwards.

3.5.4 Data Unit Structure (0-60) - geoWrite V2.0/2.1

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-27</td>
<td>$00-$1b RULER-ESCAPE-Define</td>
</tr>
<tr>
<td>26-31 $1c-$1f NEWCARDSET-Item</td>
<td></td>
</tr>
<tr>
<td>32-End $20-End Text in Geo-char. code with following exceptions: $09 - Tab. Jump $10 - GRAPHIC-ESCAPE-commence $17 - NEWCARDSET-commence $00 - Document-End</td>
<td></td>
</tr>
</tbody>
</table>

The Char. and Command Codes are often interchangable! A new page begins with an "ESC-Ruler"!

3.6 Further Text Formats

3.6.1 Text Scrap - Structure

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$01 No of following bytes in the file.</td>
</tr>
<tr>
<td>2</td>
<td>$02 NEWCARDSET: start byte = $17</td>
</tr>
</tbody>
</table>

- PAGE 70 -
- GeoCom V1.5 - Disk. format etc. -

3.4  $03$04  Font-ID
      Bit 0-5 : Point size
      Bit 6 - 15 : Font-ID No.
5    $05       Style (if all = 0 = normal)
      1 = Subscript  2 = Superscript  3 = Outline
      4 = Italic     5 = Reverse     6 = Bold
      7 = Underlined
6 - End $06-End  Text in ASCII Format with following exceptions:
      $09  -  Tab. jump
      $11  -  RULER-ESCAPE-commence (from V2.0)
      $17  -  NEWCARDSET-commence

The Char. and Command Codes are often interchangable !

3.6.2 Notice Block - Structure
The Notice block structure is very simple. A Data unit is provided in the the index block
for each existing page. The individual data units can contain a max. of 253 bytes, the
byte 255 is always $00. The only permissible command char. is $0d (CR = RETURN) !

Index-Block:
Byte 0/1 $00$01  no following block
2,3 $02$03  Page 1 of the notice block, data unit = 0
to 254  Byte $255 = $00

3.7. Definitions
3.7.1 GRAPHIC-ESCAPE (Byte $10 + 5)
Byte  Contents:
0       $10 - Introduction
1       Width
2,3     Height (in screen lines)
4,5     Data unit No. (only 64 - 127 allowed)

3.7.2 NEWCARDSET (Byte $17 + 3)
Byte  Contents
0       $17 - Introduction
1,2     Bit 0 - 5 = Point size of the relevant Font
       Bit 6 - 15 = Font ID.
3       current style

3.7.3 RULER-ESCAPE (from Geowrite V2.0)
Byte  Contents
1       $11 - Ruler- Introduction
2,3     LH border in screen points (0-392 by V2.0 / 0-552 by V2.1)
4,5     RH border in screen points (80 - 479 by V2.0 / 80-639 by V2.1)
6-21    Tabulators, pro Tab. 2 Bytes available with following parameters:
       Bit 15 set = Decimal tab.
       Bit 15 erased = Text-Tab.
       Bit 0 - 14 = Tab. position in screen points
22,23   LH current paragraph border spacing from LH margin (Parameter same as LH
       border).
24      Formatting / Line spacing as FLAG
       0,1 = $00 - LH sustained  $01 - centered
       %00 - RH sustained  %01 - full block
       %10 = single space
       %01 - 1 1/2x space
       %11 = double space
25      Text colour (at present = not used)
26,27   not used

3.8. Graphics
3.8.1 geoPaint - Picture - Structure
geoPaint documents are stored as VLIR files, such a document is always 640 * 720 points

- PAGE 71 -
large, equivalent to 80 * 90 tiles. Two tile rows are always saved in one data unit, therefore each geoPaint document has 45 data units. Each data unit includes graphic data followed by the colour information, this data is stored in a format similar to that of the Bitmap-Pack-Format. Data entries between $00 and $3f declare that "x" packed data follow - defined by the introduction-byte. If packed data follows then bit 7 is set and the bits 0 - 6 state the required number of repeats - that must follow - before the following byte appears. As previously stated, the data in a data unit is stored in a 160 tile format, 8 following bytes are combined in a single 8 * 8 point block.

At the end of the data unit 8 extra bytes follow the last of the 160 tiles - these 8 bytes are reserved for future developments and are at present without a useful function. The 8 spare bytes are followed by the colour information for the 160 tiles, this can be stored either in packed or unpacked form. The foreground colour data is stored in the bits 0 - 3 and the background colour data is stored in the bits 4 - 7.

3.8.2 Fotoscapes

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$00</td>
</tr>
<tr>
<td>1-2</td>
<td>$01,$02</td>
</tr>
<tr>
<td>3-3</td>
<td>$03-x</td>
</tr>
<tr>
<td>x+1</td>
<td></td>
</tr>
</tbody>
</table>

Width in bytes (8-pixel format)
Height - in screen rows.
Modus-Byte + Data (interchangeable)
Colour table in packed format - but refers to an unpacked Bitmap.

Modus-Byte:
0 = End of the graphic data
1 - 127 = defines how often the next byte should be repeated, by an unpacked graphic = 1 + Data + 1 + Data + 1 . . .
128 - 220 = minus 128, defines how much data follows
221 - 255 = minus 220, defines how many bytes - in total - the following pattern has.

3.9. Font - Structure

3.9.1 Info. - Block :

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$00,$01</td>
</tr>
<tr>
<td>2</td>
<td>$02</td>
</tr>
<tr>
<td>3</td>
<td>$03</td>
</tr>
<tr>
<td>4</td>
<td>$04</td>
</tr>
<tr>
<td>5-67</td>
<td>$05-$43</td>
</tr>
<tr>
<td>68</td>
<td>$44</td>
</tr>
<tr>
<td>69</td>
<td>$45</td>
</tr>
<tr>
<td>70</td>
<td>$46</td>
</tr>
<tr>
<td>71,72</td>
<td>$47,$48</td>
</tr>
<tr>
<td>73,74</td>
<td>$49,$4a</td>
</tr>
<tr>
<td>75,76</td>
<td>$4b,$4c</td>
</tr>
<tr>
<td>77,78</td>
<td>$4d-$5d</td>
</tr>
<tr>
<td>94,95</td>
<td>$5e,$5f</td>
</tr>
<tr>
<td>95</td>
<td>$60</td>
</tr>
<tr>
<td>97-121</td>
<td>$61-$74</td>
</tr>
<tr>
<td>122-127</td>
<td>$75-$7f</td>
</tr>
<tr>
<td>128,129</td>
<td>$80,$81</td>
</tr>
<tr>
<td>130-154</td>
<td>$82-$9a</td>
</tr>
<tr>
<td>155-159</td>
<td>$9b-$a0</td>
</tr>
<tr>
<td>160-255</td>
<td>$a0-$ff</td>
</tr>
</tbody>
</table>

Size of the individual points in LOW-HIGH format in bytes - from the smallest -> largest point size.

Free

Font ID LOW-HIGH-format (0-1023 possible) -> BSW9 =$00, BSW128=$20

max. 12 Data units in LOW-HIGH-format for the individual points sizes from the smallest -> largest point size. These include in the bits 6-15 a copy of the ID and in bit 0-5 the point size.

Free

Info. text, ends with a null-byte.

3.9.2 Structure

Fonts have a VLIR structure, the Index Block of a font is structured as follows :

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$01</td>
</tr>
<tr>
<td>2,3</td>
<td>$02,$03</td>
</tr>
</tbody>
</table>

This byte sequence is followed by a pointer that indicates the point size on each individual Track/sector (that holds information relevant to the respective font). Point sizes that are not available are not
- GeoCom V1.5 - Disk. format etc. -

set. Eg. Point size 10 - data unit 10 and is displayed as $20.321.

The 1st. block of the data unit can also be found in the memory (without Byte 0,1):

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>$00,$ff</td>
</tr>
<tr>
<td>2</td>
<td>$02</td>
</tr>
<tr>
<td>3,4</td>
<td>$03,$04</td>
</tr>
<tr>
<td>5</td>
<td>$05</td>
</tr>
<tr>
<td>6,7</td>
<td>$06,$07</td>
</tr>
<tr>
<td>8,9</td>
<td>$08,$09</td>
</tr>
<tr>
<td>10-10a-x</td>
<td>$0a-x</td>
</tr>
</tbody>
</table>

4.0 GeoCom - Error Messages

You could receive one of these error messages when running one of your programs, not during the compilation. Using the geoCom variable istat or the command ERROR it is possible to make a record of the program errors - if no errors occurred, then istatb = 0.

<table>
<thead>
<tr>
<th>Number</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>$00</td>
</tr>
<tr>
<td>01</td>
<td>$01</td>
</tr>
<tr>
<td>02</td>
<td>$02</td>
</tr>
<tr>
<td>03</td>
<td>$03</td>
</tr>
<tr>
<td>04</td>
<td>$04</td>
</tr>
<tr>
<td>05</td>
<td>$05</td>
</tr>
<tr>
<td>06</td>
<td>$06</td>
</tr>
<tr>
<td>07</td>
<td>$07</td>
</tr>
<tr>
<td>08</td>
<td>$08</td>
</tr>
<tr>
<td>09</td>
<td>$09</td>
</tr>
<tr>
<td>10</td>
<td>$0a</td>
</tr>
<tr>
<td>11</td>
<td>$0b</td>
</tr>
<tr>
<td>12</td>
<td>$0c</td>
</tr>
<tr>
<td>13</td>
<td>$0d</td>
</tr>
<tr>
<td>14</td>
<td>$0e</td>
</tr>
<tr>
<td>15</td>
<td>$0f</td>
</tr>
<tr>
<td>32</td>
<td>$20</td>
</tr>
<tr>
<td>33</td>
<td>$21</td>
</tr>
<tr>
<td>34</td>
<td>$22</td>
</tr>
<tr>
<td>35</td>
<td>$23</td>
</tr>
<tr>
<td>37</td>
<td>$25</td>
</tr>
<tr>
<td>38</td>
<td>$26</td>
</tr>
<tr>
<td>39</td>
<td>$27</td>
</tr>
<tr>
<td>41</td>
<td>$29</td>
</tr>
<tr>
<td>46</td>
<td>$2e</td>
</tr>
<tr>
<td>115</td>
<td>$73</td>
</tr>
</tbody>
</table>

geoCom provides the following additional error messages :

<table>
<thead>
<tr>
<th>Number</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>$80</td>
</tr>
<tr>
<td>129</td>
<td>$81</td>
</tr>
<tr>
<td>130</td>
<td>$82</td>
</tr>
<tr>
<td>131</td>
<td>$83</td>
</tr>
<tr>
<td>132</td>
<td>$84</td>
</tr>
<tr>
<td>133</td>
<td>$85</td>
</tr>
<tr>
<td>134</td>
<td>$86</td>
</tr>
<tr>
<td>135</td>
<td>$87</td>
</tr>
</tbody>
</table>

4.2 geoCom - Error Messages

These error messages are shown when or saved in the error protocol (document) when an error occurs during the compilation. These error messages are designed to help you to design your program with a minimum of errors.
- GeoCom V 1.5 -Disk. format etc. -

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$01$</td>
<td>Command too long.</td>
</tr>
<tr>
<td>2</td>
<td>$02$</td>
<td>Help: Enter only (max) 127 chars. in a line.</td>
</tr>
<tr>
<td>3</td>
<td>$03$</td>
<td>Pictures aren't allowed in the source-text.</td>
</tr>
<tr>
<td>4</td>
<td>$04$</td>
<td>Help: Under geoCom you cannot integrate graphics/pictures in the Source-Code, you must integrate your graphics with OBJECTEDIT.</td>
</tr>
<tr>
<td>5</td>
<td>$05$</td>
<td>- Brackets to deep (max. 7)</td>
</tr>
<tr>
<td>6</td>
<td>$06$</td>
<td>Help: Split the equation into two parts.</td>
</tr>
<tr>
<td>7</td>
<td>$07$</td>
<td>Variable not declared</td>
</tr>
<tr>
<td>8</td>
<td>$08$</td>
<td>Help: All your variables must be defined in the Declarations-Section.</td>
</tr>
<tr>
<td>9</td>
<td>$09$</td>
<td>Too many variables with the command (max. 16).</td>
</tr>
<tr>
<td>10</td>
<td>$0a$</td>
<td>Help: You may use a max. of 16 variables behind a variable command - use the command more than once.</td>
</tr>
<tr>
<td>11</td>
<td>$0b$</td>
<td>Full constants area</td>
</tr>
<tr>
<td>12</td>
<td>$0c$</td>
<td>Help: Set the start of the C-M-A lower, or the end higher.</td>
</tr>
<tr>
<td>13</td>
<td>$0d$</td>
<td>Binar (%) or hexadecimal ($) expect</td>
</tr>
<tr>
<td>14</td>
<td>$0e$</td>
<td>Help: You have probably pressed the &quot;$&quot; key instead of &quot;$&quot;.</td>
</tr>
<tr>
<td>15</td>
<td>$0f$</td>
<td>Syntax unknown</td>
</tr>
<tr>
<td>16</td>
<td>$10$</td>
<td>Help: Take care when entering the commands that you use the correct geoCom syntax incl. brackets, commas and colons.</td>
</tr>
<tr>
<td>17</td>
<td>$11$</td>
<td>Full code area</td>
</tr>
<tr>
<td>18</td>
<td>$12$</td>
<td>Help: Expand the memory-area or shorten the Source-Code.</td>
</tr>
<tr>
<td>19</td>
<td>$13$</td>
<td>Full variables area</td>
</tr>
<tr>
<td>20</td>
<td>$14$</td>
<td>Help: Expand the memory-area, use the program variables more than once.</td>
</tr>
<tr>
<td>21</td>
<td>$15$</td>
<td>Operator as command</td>
</tr>
<tr>
<td>22</td>
<td>$16$</td>
<td>Command as operator</td>
</tr>
<tr>
<td>23</td>
<td>$17$</td>
<td>Too many variables names (max. 127)</td>
</tr>
<tr>
<td>24</td>
<td>$18$</td>
<td>IF/WHILE to deep (max. 40).</td>
</tr>
<tr>
<td>25</td>
<td>$19$</td>
<td>No object file existing</td>
</tr>
<tr>
<td>26</td>
<td>$1a$</td>
<td>Help: Either you have forgotten the Object-Document or the names are incorrect.</td>
</tr>
<tr>
<td>27</td>
<td>$1b$</td>
<td>Variables name not declared</td>
</tr>
<tr>
<td>28</td>
<td>$1c$</td>
<td>Object not in the object file</td>
</tr>
<tr>
<td>29</td>
<td>$1d$</td>
<td>Help: You have attempted to call upon an object, that cannot be found in the Object-file or the names are incorrect.</td>
</tr>
<tr>
<td>30</td>
<td>$1e$</td>
<td>Types aren't equal (object, program)</td>
</tr>
<tr>
<td>31</td>
<td>$1f$</td>
<td>Too many object addresses (max. 128).</td>
</tr>
<tr>
<td>32</td>
<td>$20$</td>
<td>File not found or false file class</td>
</tr>
<tr>
<td>33</td>
<td>$21$</td>
<td>Pointer buffer for direct variables overflow</td>
</tr>
<tr>
<td>34</td>
<td>$22$</td>
<td>Code-area not over $5000$</td>
</tr>
<tr>
<td>35</td>
<td>$23$</td>
<td>Start adress higher than the end adress</td>
</tr>
<tr>
<td>36</td>
<td>$24$</td>
<td>Command only in the definition part</td>
</tr>
<tr>
<td>37</td>
<td>$25$</td>
<td>Help: Take care when placing the commands, not every command can be placed everywhere!</td>
</tr>
<tr>
<td>38</td>
<td>$26$</td>
<td>Command only in the declaration part.</td>
</tr>
<tr>
<td>39</td>
<td>$27$</td>
<td>Help: Take care when placing the commands, not every command can be placed everywhere!</td>
</tr>
<tr>
<td>40</td>
<td>$28$</td>
<td>Fals text version of INCLUDE-file</td>
</tr>
<tr>
<td>41</td>
<td>$29$</td>
<td>Help: The INCLUDE-file must have the same geoWrite format as the Main-Source-Code.</td>
</tr>
<tr>
<td>42</td>
<td>$2a$</td>
<td>Stringlength only in range 1-254</td>
</tr>
<tr>
<td>43</td>
<td>$2b$</td>
<td>Unimplemented string-control-code</td>
</tr>
<tr>
<td>44</td>
<td>$2c$</td>
<td>ROW-area only in range of 1 to 65534</td>
</tr>
<tr>
<td>45</td>
<td>$2d$</td>
<td>Globar area had to defined first</td>
</tr>
<tr>
<td>46</td>
<td>$2e$</td>
<td>INCLUDE-texts maximum of 1</td>
</tr>
<tr>
<td>47</td>
<td>$2f$</td>
<td>Including text not found</td>
</tr>
<tr>
<td>48</td>
<td>$30$</td>
<td>Rekuservs including of text not allowed</td>
</tr>
<tr>
<td>49</td>
<td>$31$</td>
<td>Compiling throught STOP brokeed</td>
</tr>
<tr>
<td>50</td>
<td>$32$</td>
<td>Help: You have pressed the RUN/STOP key during the compilation!</td>
</tr>
<tr>
<td>51</td>
<td>$33$</td>
<td>Program-/Block-end with open IF/WHILE-command</td>
</tr>
</tbody>
</table>
| 52  | $34$ | Help: You have forgotten an ENDIF/LOOP-command. Check your IF.
- GeoCom V1.5 - Disk. format etc. -

```
/ WHILE...-Structure.
36  $24
   - Code area not under $2800
   Help: The area from $0400 to $2800 is reserved for geoCom,
        therefore you must start your Code-Area above $2800.
37  $25
   - ENDF/ELSE/LOOP without IF/WHILE
   Help: You have forgotten an IF/WHILE...-command. Check your IF
         / WHILE...-Structure.
38  $26
   - Error buffer overflow
   Help: Remove some errors from the Source-Code - there are more
        than 35 errors - IF...ENDIF-command errors appear most often!!!
39  $27
   - Disk error appeared
```

5.0 Appendix - A

Conversion - set bit, Hex - Dec.
At particular addresses it is possible to set or interrogate up to 8 bits (0-7), they can
either be set/interrogated singley or as a group. The individual bits have the following
parameters:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Hex</td>
<td>$80</td>
<td>$40</td>
<td>$20</td>
<td>$19</td>
<td>$08</td>
<td>$04</td>
<td>$02</td>
<td>$01</td>
</tr>
</tbody>
</table>

Is the parameter = 0 or has 0 been poked then = no bit set.
To set bit 6 and 7 the respective parameters must be added together = 64 ($40) and 128 ($80). You can "Poke and Peek" "write and read" in both HEX and DEC form, take
care to observe the correct variable format and the pre-select char. ($)