Disclaimer

The program has been developed and reproduced carefully, neither the author nor the distributors, however, can guarantee that the program and this guide are free of errors. Therefore GIGA- SOFT under no circumstances will be liable for any direct, indirect, incidental or consequential loss of use, stored data, profit or contracts which may arise from any error, defect or failure of this software.

GIGA- SOFT has a policy of constant development and improvement of their products. We reserve the right to change manuals and software at any time and without notice.

Instructions for the Basicextension "GIGA-BASIC V 1.00"

Introduction:

Although the Sinclair QL comes with a real good Basic, some commands are missing, which would offer the full power of the QL. This extension set should increase your motivation to program in Basic. With over 70 commands and functions, Giga-Basic is a useful extension for the QL. Before starting work with Giga- Basic you should read this manual carefully. You should never work with your original copy of Giga- Basic. To obtain working copies a backup program is included. To start this program enter: 'exec_w mdv1_clone'. You can backup Giga- Basic up to three times.

The extension includes the following groups :

- grafics
- spritehandling and spriteanimation
- base conversion
- fullscreen basiceditor
- direct access to medium
- multitasking clocks
- mousedriven screenoriented menus
- pull-down-menus
- multitasking control commands
- programmable function keys
- others

GRAFICS

PAINT #dev,x,y

Fills an irregularly bordered area of the chosen

screen with the ink colour.

dev: devicenumber of the screen
x: x- coordinate
y: y- coordinate

Direct access to medium

* *GET #dev,variable (,variable)*

Gets a value from medium and writes it into the variable. The type of the value depends on the type of the variable.

Example: GET #4,integer%

#dev: devicenumber
variable: any type and number of variables

* *BGET #dev,byte (,byte)*

Gets byte from medium and puts it into the variable.

#dev: devicenumber
byte: any type and number of variables

* *PUT #dev,variable (,variable)*

Writes value to microdrive. Any variabletype is allowed.

#dev: devicenumber
variable: Any type and number of variables.

* *BPUT #dev,byte (,byte)*

Writes byte to microdrive.

#dev: devicenumber
byte: variable which gets a byte

SET_POINTER #dev,pointer

Sets filepointer to new position. With this command it is possible to have direct access to microdrive (or FLP, HDK, FDK and so on).

#dev: devicenumber
pointer: longword containing the pointer

pointer=GET_POINTER (#dev)

Gets the pointer of the selected microdrive.

#dev: devicenumber
pointer: variable containing the pointer

Base conversion

The following functions provide an easy way to convert bases.

hexnum$=CHEX$(decimal)

Converts a decimal value into a hexadecimal string

hexnum$: string which will contain the hexnumber
decimal: variable containing the decimal number

decimal=CHEX(hexnum$)

Converts a hexadecimal string (max. 32 bit) into a decimal number.

decimal: variable which will contain the decimal number
hexnum$: string containing the hexnumber

binary$=CBIN$(decimal)

Converts a decimal number into a binary string (32 bit).

decimal: variable which will contain the decimal number
binary$: string containing the binary number

decimal=CBIN(binary$)

Converts a binary string into a decimal number.

decimal: variable which will contain the decimal number
binary$: string containing the binary number

Multitasking Control Commands

The following commands are intended to control the multitasking capabilities of the QL. Now it is possible to delete, suspend or activate jobs from Basic.

JOB_INF #dev

This command displays a list of all active jobs. A Job is a program working in the background. Additionally you can see the priority, the owner job, the baseaddress and the tagnumber. Job 0 is the Basicinterpreter. For further informations on multitasking refer to the Sinclair User Guide.

#dev: devicenumber

SUS_JOB jobnr,tagnr,timeout

Suspends a job for a period.

jobnr: jobnumber
tagnr: tagnumber
timeout: Number of frames the job being deactive (-1: infinite).

REL_JOB jobnr,tagnr

Releases a suspended job. This command is the reverse of SUS_JOB.

jobnr: jobnumber
tagnr: tagnumber

PRIOR_JOB jobnr,tagnr,priority

Sets the priority of a job. Priorities are allowed in the range from 0 to 127. 127 is the highest priority. If the priority is high, more time is available for the job.

jobnr: jobnumber
tagnr: tagnumber
priority: priority

Sprites and Animation

Giga-Basic offers a great number of efficient commands for development and animation of sprites. So it is easy to generate actiongames or programs using icons. Sprites are organized in a 32 x 20 matrix and are flickerfree.

Important definitions :

spritedatablock (sprdat): This is a memoryblock which contains the bytes for the shape (mask) of the sprite. A sprite shape contains 160 bytes. Every spritedatablock can be attached to every sprite.

spritenumber (sprnr): A sprite will be activated under a spritenumber. Under this number the sprite can be moved over the whole screen.

SPRDIM spritenr,datanr,anatenr

Reserve memory for sprites. The defaults are :

SPRDIM 4,16,4

spritenr: number of possible sprites

datanr : number of possible spritedatablocks
anatenr: number of the sprites which can be animated

SPRCLR

SPRCLR releases the memory allocated by SPRDIM. All defined Sprites are lost.

INVMASK #dev,x,y,sprdat

Prints a spritemask onto the screen. The coordinates are relative to the left upper edge of the selected window. The coordinates have pixel size (This is not a sprite. Only a mask will be drawn.).

dev: devicenumber
sprdat: spritedatablock

SPRON sprnr,sprdat

Activates a sprite with a spritedatablock.
Note: This command does not have any effect on the screen. The sprite will not be visible until it is activated by the MOVESPR command.

sprnr: spritenumber
sprdat: spritedatablock

SPROFF sprnr

Removes the selected sprite.

sprnr: spritenumber

REFRESH

Important after 'CLS'. All active sprites are refreshed.

INVSPRITE sprnr

The chosen sprite is inverted.

sprnr: spritenumber

MOVESPR sprnr,x,y (,sprdat)

Sets a sprite to a new position. The optional spriteparameter is intended to change the appearance of the sprite. If no spritedatablock paramater is given, the sprite image does not change.

sprnr: spritenumber
x,y: absolute pixel coordinates

sprdat: spritedatablock

Several sprites can be moved using only one command. This type of motion is named animation. It is a really easy task to move rockets, men, cars and other things now.

SETANIMATE sprnr,sprdat(,sprdat1)(,sprdat2)

This command has as many parameters as you want to. The given spritedatablocks are connected in series.

Note: Before using ANIMATE, you have to initialize the routine with the SETANIMATE command. A maximum of 16 spritedatablocks may be connected.

sprnr: spritenumber
sprdat: spritedatablock

CLRANIMATE sprnr

The selected spriteentry will not be animated after the use of CLRANIMATE.

sprnr: spritenumber

STEPSPRITE sprnr,xstep,ystep,statx,staty

This command can be used after every SETANIMATE. You can change the direction and speed of the animation in your basic progras.

sprnr: spritenumber
xstep: stepsize x
ystep: stepsize y

statx: 0 After reaching the border of the screen invert the x-direction.
1 After reaching the border appear at the otherside.
2 After reaching the border kill the sprite.

staty: Same as statx but refering to the y-direction.

ANIMATE

Moves all sprites which are declared with the SETANIMATE command over the screen.

sprite=COLLISION(sprnr)

Asks whether two sprites are overlaid. If it is true COLLISION returns the spritenumber; otherwise -1.

sprnr: spritenumber
sprite: If the sprite isn't in contact with another sprite -1 will be returned, otherwise the spritenumber.

Spritedefinition commands 1

Sprites can be defined for MODE 4 or MODE 8. Following an example of an eightcolour sprite:

```
100 :
110 SPRDEFBLOCK starship
120 :
130 SD8 "................"
140 SD8 "................"
150 SD8 ".......11......."
160 SD8 "......1111......"
170 SD8 "....22222222...."
180 SD8 "..3333333333.."
190 SD8 "...33333333..."
200 SD8 ".....7....7....."
210 SD8 "....7......7...."
220 SD8 "...7........7..."
```

The fourcolour example :

```
100 :
110 : SPRDEFBLOCK disk
120 :
130 SD4 "................................"
140 SD4 ".################################."
150 SD4 ".################################..."
160 SD4 ".################################."
170 SD4 ".################################."
180 SD4 ".###############....###############."
190 SD4 ".############........############."
200 SD4 ".##########..........##########."
210 SD4 ".############........############."
220 SD4 ".###############....###############."
230 SD4 ".################################."
240 SD4 ".################..################."
250 SD4 ".################..################."
260 SD4 ".################################."
270 SD4 "................................"
```

The colours are set in the following form:

MODE 4

```
red       : '1'
green     : '2'
white     : '3','0'
black     : all other characters
```

MODE 8


```
blue      : '1'
red       : '2'
magenta   : '3'
green     : '4'
cyan      : '5'
yellow    : '6'
white     : '7','0'
black     : all other characters
```

SPRDEFBLOCK sprdat

Clears the selected spritedatablock and prepares it for a new definition.

sprdat: spritedatablock

D4 defblock$

Command to define a fourcolourspsrite. Up to 20 commands can be used after a SPRDEFBLOCK command. The string must be 32 characters long.

SD8 defblock$

Command to define an eightcolourspsrite. Up to 20 commands can be used after a SPRDEFBLOCK command. The string must have a length of 16 charactars.

SPRLOAD name$

With this command you can load previously defined spritedatablocks. Before you use this command enough space must be reserved by SPRDIM.
Example : SPRLOAD "MDV1_PACMAN_SPR"

SPRSAVE name$

If you want to save the allocated sprite area you can use this command. Only the area for spritedatablocks will be saved.

flag=SPRACTIVE(sprnr)

With SPRACTIVE you can ask whether a sprite is active. 1 is true and 0 is false.
sprnr: spritenumber

x=SPRXPOS(sprnr)
y=SPRYPOS(sprnr)

With these functions you can find out the location of a sprite.
sprnr: spritenumber

MENU CONTROL COMMANDS

The following commands support userfriendly screenorientated menus. Now you can program mousedriven menus as with the APPLE MACINTOSH (R).
The handling is very simple. With commands like MENUPR or MENUBLOCK you define a BLOCK. This block can be manually inverted or selected with the MOUSE function. Possible inputmedia are the cursorkeys or a mouse with the ABC-interface (included in the big ABC package).

Example :

```
100 :
110 REMark small example menu
120 :
130 CLS
140 PRINT "M E N U"
150 PRINT:PRINT
160 MENUPR 1," Start a program"
170 MENUPR 2," List a program"
180 MENUPR 3," End"
190 :
200 a=MOUSE
210 :
220 SELect on a
230   =1:start
240   =2:LIST
250   =3:STOP
260 END SELect
```

After entering and starting the program, the menupoints appear as if they were printed with the PRINT command. An arrow appears, too. This arrow can be moved over the whole screen. If the arrow is in range of a menupoint this will be inverted. So you can see exactly what you have chosen. By pressing the button or space the selected menunumber will be returned.

SETMDEV mode

Selects inputmedium for the menucommands.

mode: 0: keyboard (cursorkeys/ space)
1: mouse

MENUDIM number

Reserves memory for the menupoints. Space for pull-down-menus will be automatically allocated.

number : the maximum number of menupoints

MENUBLOCK #dev,blknr,x,y,x0,y0

This command marks a block with the chosen menublocknumber.

#dev: devicenumber of a screen
blknr: blocknumber
x,y: size of the block
x0,y0: position relative to the selected window

MENUPR #dev,blknr,text$

Prints a text on the screen similar to the print command and activates it as a menublock.

#dev: devicenumber of a screen
blknr: blocknumber
text$: text

The separator ';' is allowed.

ICON #dev,blknr,sprdat,x,y

Similar to the INVMASK command it displays a spriteblock on the screen. The difference is that ICON marks it as menublock. With this command it is possible to access symbols in a similar way as the MENUPR command. You can define ICONs and use them for defining MACINTOSH (R) style programs.

dev: devicenumber
blknr: menublocknumber
sprdat: spritedatablock
x,y: pixel coordinates relative to the window

INVBLOCK blknr

Inverts a block.

blknr: menublocknumber

CLRBLOCK blknr

Clears a block.

blknr: menublocknumber

nr=MOUSE(x,y)

Displays an arrow which can be moved over the screen by using the mouse. With the arrow you can select an item.

nr: if no menupoint was chosen -1 will be returned,

otherwise the menublocknumber will be returned.

x,y: startcoordinates of the arrow

x=MXPOS,y=MYPOS

These functions return the position of the arrow after pressing the SPACE-key.

Pull-Down-Menus

This a new type of menutechnique. On top of the screen you can see a headline holding the menupoints. If you move the arrow to one of the points, a window will be opened with a submenu. Now you can choose the point you want in the submenu. With the Pull- Down- Menus you can handle a great number of menupoints on a very small room.

Example :

```
100 SPRDIM :REMark Reserves space for the arrow
110 MENUDIM :REMark Allocates space for the pull-down-menu
120 :
130 MENU 0,0,1,"Addresses"
140 MENU 1,0,1,"Clear"
150 MENU 2,0,1,"Input"
160 MENU 3,0,1,"Edit"
170 :
180 MENU 0,1,1,"File"
190 MENU 1,1,1,"Load"
200 MENU 2,1,1,"Save"
210 :
220 MENU 0,2,1,"Exit"
230 MENU 1,2,1,"Reset"
240 MENU 2,2,1,"Basic"
250 :
260  SETMENU :REMark Clears the screen and shows the
menuheadline
270 :
280 GETMENU :REMark Shows the arrow and gets the menupoint
290 x=HMENU
300 y=VMENU
310 :
```

MENU vnr,hnr,active,string$

Command to define a pull-down-menus.

vnr: Vertical coordinate. The headline has the coordinate zero.

Note: The menupoints within the headline (vnr=0) must be defined in ascending order. Every headlinepoint must have a submenu. A maximum of 10 items can be

defined in the vertical direction.

hnr: Horizontal coordinate. The number of horizontal items is restricted to a maxiaum of 8. The total length of the items in the headline must be selected to fit according to the selected screenmode. This is important for compatibility between mode 256 and mode 512.

active: Flag which selects whether you can access the menupoint or not.

string$: Text of the menupoint. The length is restricted to 14 characters.

SETMENU paper1,paper2,actcol1,pascol

Clears the whole screen. Displays the headline.

paper1: Screencolour
paper2: Bordercolour of the headline
actcol: Colour of the active menupoints
pascol: Colour of the passive menupoints

GETMENU x,y

Displays the arrow and allows the user to select menupoints.

x,y: startposition of the arrow

Default: GETMENU 256,100

ACTIVE vnr,hnr,active

Activates and deactivates menupoints.

vnr: vertical Position of menupoint
hnr: horizontal Position of menupoint
active: flag, 1-active, 0-inactive

x=HMENU
y=VMENU

With these functions you can get the position of the chosen menupoint.

possible range :
HMENU (0-7)
VMENU (1-9)

Programmable function keys

Directly after starting the Basic extension, the functionkeys

are programmed. Information about the assignments can be gained by pressing "F1". This assignment can easily be changed by the user. Furthermore the functionkeys can be switched off if they would disturb the function of other programs.

KEYS #dev

Lists all functionkey assignments to the specified device.

dev: devicenumber (default is 1)

KEY keynr,string$

Allows the user to change the functionkey assignment.

keynr: Number of the functionkey (1 to 10, numbers greater than 5 are aktivated by pressing the shiftkey simultaniously.

string$: String containing the command (max. 32 characters). Example: KEY 1,'LIST' & chr$(10)

KEYSON

Turns funktionkeys on.

KEYSOFF

Turns functionkeys off.

CLOCKCOMMANDS

It is possible to display either a digital or an analogue-clock on the screen. There is also the possibility of changing colour and size to adapt the clocks to own programs.

DCLOCK on,x,y,paper,ink1,ink2

Displays a digital clock.
Default : DCLOCK 1,340,0,2,7,4

on: flag, 0-removes the clock, others-activates the clock
x,y: right upper coordinate of the clock in pixel coordinates
ink1: inkcolour
ink2: bordercolour

ACLOCK on,x,y,size,paper,ink1,ink2,ink3,ink4

Displays an analogue- clock.
Default : ACLOCK 1,0,0,40,0,2,2,4,4

on: flag, 0-removes the clock, others-activates the clock
x,y: right upper coordinate of the clock in pixel coordinates
size: vertical ize of the clock
paper: papercolour.
ink 1-4: colour for the hands of the clock an the circle around it

OTHER COMMANDS

CAT #mdvnr

Displays the directory of the specified drive in a formatted form. Furthermore it displays the number of blocks (512 bytes) each program uses.

#mdvnr: number of drive (default : 1)

DUMP #dev

Displays all variables with contents, procedures and functions with linenumbers.

#dev: outputdevice (default is 1)

COMMANDS #dev

Lists all new Basiccommands with their startaddress on the outputdevice.

#dev: outputdevice (default is 1)

HRDCOPY inv

Prints hardcopy on EPSON-compatible printers. Through technical restrictions, it is only possible to print a maximum of 480 horizontal points.

inv: flag, 1-inverted print, 0-normal print

SYSTEM #dev

Displays the systemvariables on screen.

#dev: outputdevice (default is 1)

a=FREE

Returns the amount of free Basicmemory.

SCREEN #dev,linenr,tab

default : SCREEN #1,1,3

This command enters the screeneditor. It allows the user to edit Basicprograms in a way similar to QUILL. Unlike a normal ASCII-Editor all entered lines are syntactically checked by the interpreter.

Note : The interpreter will not accept lines after a programbreak if the functions and procedures are not reinitialized. This is possible by using the CLEAR command, which will produce the message 'PROC/DEF CLEARED'. After this message the work with the screeneditor can go on.

dev: windownumber to edit in
linenr: linenumber which will be displayed first
tab: stepsize of the inbuilt tabulator

The editor accepts the following keysequences:

cursup	
cursdown	
cursright	
cursleft	
ESC	leaves the editor
TABULATE	tabulator
SHIFT&ALT&UP	jumps to start of program
SHIFT&ALT&DOWN	jumps to end of program
ALT&UP	page up
ALT&DOWN	page down
CTRL&RIGHT	deletes character under cursor
CTRL&LEFT	deletes character at the left of the cursor
CTRL&ALT&LEFT	clears Basicline
CTRL&ALT&RIGHT	deletes all characters at the right of the cursor
SHIFT&UP	jumps to the first line of the screen
SHIFT&DOWN	jumps to the last line of the screen
ALT&LEFT	jumps to start of line
ALT&RIGHT	jumps to end of line

SETFONT #dev,fount1,fount2

Gives the user the possibility of using a selfdefined Characterset. It is possible to define up to two charactersets at one time, in which case a character is displayed from the first characterset, if defined there, if not defined, it is taken from the second and, if it also is not defined there, the first defined character of the

second set is displayed. To select the inbuilt fonts of the QL, just enter zero for the startaddress of the font.

fount1 : startaddress of the first font
fount2 : startaddress of the second font

Example: (Using the supported characterset 'BIG_CST'.)

```
100 a=RESPR (1024)              :REMark Reserve space for font.
110 LBYTES 'mdv1_BIG_CST',a     :REMark Load new font.
120 FOR channel = 0 TO 2        :REMark Loop
130   SETFONT #channel,a,0        :REMark Activate new font for
140   CLS #channel                :REMark every window.
150 END FOR channel             :REMARK End loop
```

MONSCR mode

Activates the switch-on-status of the windows for the monitormode.

mode: Selects 4 or 8 colourmode.

TVSCR mode

Activates the switch-on-status of the windows for the televisionmode.

mode: Selects 4 or 8 colourmode.

SETMON #dev, xsize, ysize, x0, y0, paper, strip, ink, borderwidth, bordercolour

Changes a defaultwindow in the monitormode.

SETTV #dev, xsize, ysize, x0, y0, paper, strip, ink, borderwidth, bordercolour

Changes a defaultwindow in the televisionmode.

mode=GETMODE

Returns the screenmode.
4= fourcolour, 8= eightcolour.

Windowcommands

The windowcommands allow the user to work with 'real' windows. With these commands it is possible to save the background of a window before writing to it and to restore this background after closing the window. This technique is known as 'refreshing'.

SCALE #nr,xs,ys,x,y

Saves an area of the screen.

Number from 0 to 15. Tis number represents the label for the saved screen. It has to be specified in the other commands refering to the saved screen area.

ys: size of the window
s: left upper position of the window

RLOAD nr

Redisplays an saved area of the screen.

Labelnumber (0 to 15)

RCLEAR nr

Clears the part of the memory containing the saved screen.

Labelnumber (0 to 15)