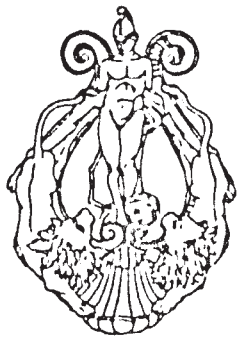
Ultrasoft

Graphics Construction Kit



for the Sinclair QL

GRAPHIC CONSTRUCTION KIT

AUTHORS

M&A PABICH 1986

Introduction:

This program is not only a set of new SuperBASIC keywords to enable you to release the graphic power of your QL. BUT to enable you to write a graphic program to suit your needs. In other words puting power to your fingertips and letting your feel the raw power of your QL. Many people believe that the QL graphic capabilities have not yet been released. This programme will help you to do exactly this.

Graphic Construction Kit lets you construct your own graphic programs to suit your own requirements.

For your own saftey please make at least one backup copy by using LRUN MDV1_GCK_CLONE

To give you an idea of how to write a graphics programme there is a working example in BASIC and as a compiled version.
First of all load the extensions by LRUN MDV1_BOOT Then LRUN MDV1_GCK_DEMC for the BASIC version and EXEC_W MDV1_GCK_EXE for the compiled one.

The two example programs are to give you ideas and to inspire you to write your own graphics program which is much better.

We have not written a handbook for the demostration programs because that is exactly what they are.

The programs have been marked and legal prosecution will be persued in respect of any illegal or pirate copies.

ULTRASOFT reserve all rights, copyrights whether intellectual or physical on this program.

(c) Ultrasoft 1986

Ultrasoft accept no liability for any direct, indirect, incedental or consequential damage or loss including but not limited to loss of use, stored data, profit or contracts which may arise from any error, defect or failure of Graphic Construction Kit.
Due to Ultrasoft's policy of continual development and improvement of its products, Ultrasoft reserve the right to change the manuals and programs at any time and without notice.

QL, SuperBASIC are the trade marks of Sinclair Research Ltd

SCREEN HANDLING

GRAPHIC CONSTRUCTION KIT allows you to handle up to 16 screen banks (memory permitting) numbered from 0 to 15. Many commands are installed to aid your work with complete screens and parts of them.
In addition it enables you to process SPECTRUM screens directly, allows scaling and reducing and makes conversion of a screens mode easy.
All the following commands do mainly work with the whole main screen. For block processing please refer to the appropriate section of the manual.

Keywords:

SCR_IN	SCR_OUT	SWITCH	BLANK
SLOAD	SSTAT	ZOOM *	REDUCE
MODE	CMODE	SPECLOAD	STRETCH
SCR_MODE	SCR_ADR		

SCR_IN

SYNTAX:

1 SCR_IN num
2 SCR_IN num,width,height,x,y
3 SCR_IN num,width,height,x,y,x_new,y_new

1 This command transfers a specified screen display from a bank into the main screens memory.
2 Alternatively a part of the screen bank may be transfered to the main screen.
3 The most complex variation of SCR_IN transfers a specified block from a screen bank to an absolute position in the main screen.

Examples:

i SCR_IN 3 transfers bank 3 to the main screen.
ii SCR_IN 5,256,128,128,64 transfers the middle quarter of bank 5 to the same position in the main screen.
iii SCR_IN 2,100,100,0,0,411,155 gets a part from the upper left corner of bank 2 and puts it to the lower right corner of the main screen.

SCR_OUT

SYNTAX:
the same as SCR_IN

This command works exactly in the opposite way as SCR_IN, i.e. it transfers screens or parts of it to a screen bank.

SWITCH

SYNTAX:
the same as SCR_IN

This command is another alternative to SCR_IN, but swaps a complete screen or a part of it with an existing bank.

Example: SWITCH 4 completely swaps the main screen with bank 4, that means the main screen is put into the bank and the bank will be put into the main screen at one time.

BLANK

SYNTAX:

BLANK num

This command clears the heap of a given bank, i.e. the stored screen doesn't exist any longer.

SLOAD

SYNTAX:

SLOAD 'dev_name'
SLOAD num,"dev_name"

SLOAD loads a screen from a medium to the main screen or to a bank if any bank number is given.

SSTAT

SYNTAX:

SSTAT(num)

SSTAT is a function which returns the states of a given bank:

0 = bank unused
1 = bank used

Example:
```
10 INPUT"Screen number:";a
20 IF SSTAT(a)=1
30    SCR_IN(a)
40 ELSE PRINT"Screen does not exist!"
50 ENDIF
```

SPECLOAD

SYNTAX:

SPECLOAD 'dev_name'

This command loads an original SPECTRUM screen into the main screen. Of course the screen must be previously transfered via RS 232 to a QL medium.

STRETCH

SYNTAX:
STRETCH

This command stretches a loaded SPECTRUM screen to the full QL size.

ZOOM

SYNTAX:
1. ZOOM
2. ZOOM n

1. ZOOM allows you to enlarge a quarter of a screen to the full size and works fully interactive.
2. ZOOM n enlarges a specified half of the screen, where n must be in the range from 0 to 3:

0 - top half
1 - bottom half
2 - left half
3 - right half

REDUCE

SYNTAX:
1. REDUCE bnum
2. REDUCE num,bnum

1. This command reduces the full main screen to the quarter size and stores it in a specified block (not screen bank!)
2. Alternatively to the main screen a given screen bank might be reduced.

MONO

SYNTAX:
MONO

MONO reduces a coloured display to a monochrome screen.

CMODE

SYNTAX:
CMODE

This command allows you to convert a screen picture from MODE 4 to 8 and reversed.

SCR_MODE

SYNTAX:
SCR_MODE

This function returns the current MODE of the screen, which is 8 for MODE 4 and 0 for MODE 8.

SCR_ADR

SYNTAX:
SCR_ADR(x,y)

This function returns the memory adress of a given absolute position on the screen.
Please refer to a QDOS manual for details about screen organisation.

BLOCK HANDLING

The following section covers the commands neccessary to handle blocks (stored parts of a screen display) in a very confortable way. There are 16 blocks available at a time (memory permitting) numbered from 0 to 15. Many of these commands are interactive commands, i.e. in many cases a block is simply positioned with help of the cursor keys. Please refer to the appendix for a detailed description of interactive commands.

Keywords:

SET_BLK *	BLK_OUT	BLK_IN *	COPY_BLK *
CUT_BLK *	CLR_BLK	BLOAD	BSAVE
BSTAT	INVERT	SIZE*	ROLL *
ROLL_H	ROLL_V	MIRROR *	MIRROR_H
MIRROR_V	ADJUST	RANGE *	

(*-marked commands are interactive)

SET_BLK

SYNTAX:
1 SET_BLK bnum
2 SET_BLK bnum,width,height,x,y

This command is used to set up a block definition, where 1 works interactive and 2 needs the parameters to be given in the command line. The set definition will be used as a size and default position for all subsequent accesses on the specified block.

BLK_OUT

SYNTAX:

```
BLK_OUT bnum
```

This command saves a part of the main screen in a specified block, which size and position was defined by SET_BLK

BLK_IN

SYNTAX:

```
1  BLK_IN bnum
2  BLK_IN bnum,x,y
```

This command gets a specified block and puts it into the main screen if given the block is put to this position, otherwise the command works interactive

CLR_BLK

SYNTAX:

```
CLR_BLK bnum
```

CLR_BLK deletes a specified block from memory

EXAMPLE:

```
10 SET_BLK 4,256,128,128,64
20 BLK_OUT 4
30 BLK_IN 4,0,0
40 BLK_IN 4,256,0
50 BLK_IN 4,0,128
60 BLK_IN 4,256,128
70 CLR_BLK 4
```

BSAVE

SYNTAX:

```
BSAVE bnum,'dev_name'
```

This command saves the specified block from memory to a device

BLOAD

SYNTAX:

```
BLOAD bnum,'dev_name'
```

Loads a saved block from a device back to memory The block will be stored under the given number

RANGE

SYNTAX:

```
RANGE
RANGE width,height,x,y
```

RANGE sets the working area for all interactive block commands
If a block will be set by BLK_IN which is to big for the specified range window, it can't be moved
This command is very useful for example to secure icons at one edge of the screen, because it actually defines a work window.

EXAMPLE:

```
10 RANGE 416,256,96,0
20 WINDOW 96,256,0,0
30 BORDER 1,230
40 CLS
50 PRINT 'NOW WE WILL SET'????'THE SIZE OF '????'    BLOCK 4'
60 SET_BLK 4
70 CLS
```

ADJUST

SYNTAX:

```
ADJUST bnum
```

If a block was defined before the last RANGE command, this command will manipulate it to fit inside the actual range.
First the x,y position will be corrected, then if the block still doesn't fit it will be truncated to the actual RANGE size.

BSTAT

SYNTAX:

```
BSTAT bnum
```

This function returns the states of a block: 0 = unused, 1 = used

SIZE$

SYNTAX:

```
SIZE$(bnum)
```

This function returns the size of a block as a string.
The returned string consists of the usual parameters width,height,x,y which are seperated by commas. Any single parameter could be accessed by string slicing

COPY_BLK

SYNTAX:
1 COPY_BLK bnum
2 COPY_BLK width,height,x,y

Use the following example (or use your own!) to show your friends what's a DL graphics are!! Full screen power! A real window is moved by COPY_BLK bnum and is copied to the end position COPY_BLK with full parameters moves a defined block onto a given screen position.

EXAMPLE:
```
10 REMark first load a screen into bank 0
20 SCR_IN 0
30 SET_BLK 0,256,128,128,64
40 COPY_BLK 0
50 COPY_BLK 64,64,0,0
60 GO TO 30
```

CUT_BLK

SYNTAX:
1 CUT_BLK bnum
2 CUT_BLK width,height,x,y

This command is similar to COPY_BLK, but leaves the old position blank. You can change COPY_BLK in the last example to CUT_BLK to see the difference.

INVERT

SYNTAX:
1 INVERT bnum
2 INVERT width,height,x,y

INVERT inverts a block which was defined by SET_BLK or a specified area on the screen.

POLL

SYNTAX:
1 POLL bnum
2 POLL width,height,x,y

The command POLL works interactive. It rolls a block which was defined by SET_BLK or a specified area on the screen.
The difference between ROLL and PAN/SCROLL is that no pixels will be lost but really be rolled around.

POLL_H, POLL_V

SYNTAX:
1. POLL_H step,bnum
2. POLL_H step,width,height,x,y
POLL_V (the syntax is the same for both commands)

These commands will roll a given block or a specified area on the screen horizontally or vertically (_H or _V) by the defined step.

EXAMPLE:
```
10 SET_BLK 16,240,127,96,23
20 POLL_H -10,16
30 POLL_V 2,16
40 GO TO 20
```

MIRROR

SYNTAX:
1. MIRROR bnum
2. MIRROR width,height,x,y

This command works interactive. It will mirror a given block or a specified area on the screen.

MIRROR_H, MIRROR_V

SYNTAX:
1. MIRROR_H bnum
2 MIRROR_H width,height,x,y
MIRROR_V (the syntax is the same for both commands)

These commands will mirror a given block or a specified area on the screen horizontally or vertically (_H or _V)

COMPRESSOR HANDLING

These commands are used to handle the included screen compresser. The commands are sometimes very similar to the BLOCK and SCREEN commands, but because the compressor has got it's own banks they have nothing to do with the ordinary block or screen banks. Only the size and position set by SET_BLK will be respected.
There are 16 different banks available for compressed screens AND 16 banks for compressed blocks, numbered from 0 to 15.

Keywords:

CSCR_OUT	CSCR_IN	CSSAVE	CSLOAD
CSSTAT	CLR_CSCR	CBLK_OUT	CBLK_IN
CBSAVE	CBLOAD	CBSTAT	CLR_CBLK
CFRAME	CREFRAME		

SYNTAX:
CSCR_OUT cnum

CSCR_OUT compresses the main screen and puts it in the given bank

CSCR_IN

SYNTAX:
CSCR_IN cnum

CSCR_IN decompresses the given bank and transfers it to the main scree

CSSAVE

SYNTAX:
CSSAVE 'dev_name'
CSSAVE cnum,'dev_name'

This command saves either the compressed main screen or a specific bank to a medium, where the latter must be previously compressed b CSCR_OUT

CSLOAD

SYNTAX
CSLOAD 'dev_name'
CSLOAD cnum,'dev_name'

This command is used to load back a stored compressed screen, either t the main screen or to a given compressor bank

CSSTAT

SYNTAX:
CSSTAT(cnum)

This function returns the status of a compressor screen bank, where 0 means unused and 1 that the bank is in use

CLR_CSCR

SYNTAX
CLR_CSCR cnum

CLR_CSCR deletes a compressed screen from memory

EXAMPLE 10 REMark First load a screen
20 CSCR_OUT 12
30 CLS:CSSAVE 12,'MOV1_TEST_cpr'
40 CSLOAD 4,'MOV1_TEST_cpr'
50 CSCR_IN 4
60 CLR_CSCR 12:CLR_CSCR 4

CBLR_OUT

SYNTAX:
CBLR_OUT cnum
CBLR_OUT cnum,width,height,x,y

This command compresses a block and stores it in the specified compressor block bank. If only the bank is given, the command uses the parameter defined by SET_BLK.

CBLR_IN

SYNTAX:
CBLR_IN cnum

This command decompresses a given block and puts it to the old position on the main screen

CBSAVE

SYNTAX:
CBSAVE cnum,'dev_name'

CBSAVE saves a compressed block with was stored by CBLR_OUT to a device

CBLOAD

SYNTAX:
CBLOAD cnum,'dev_name'

This command loads a compressed block from a device and stores it in the specified bank

CBSTAT

SYNTAX:
CBSTAT(cnum)

This function returns the states of a compressed block bank where 0 means unused and 1 means used

CLR_CBLK

SYNTAX:
CLR_CBLK cnum

This command deletes a compressed block from memory.

CFRAME, CREFRAME

SYNTAX:

```
CFRAME#ch,width,height,x,y
CREFRAME
```

Alternatively to SUPER Basic 'WINDOW' you can use the commands CFRAME and CREFRAME. The syntax is the same as for WINDOW, but the old area is stored in memory in compressed form. CREFRAME brings back the last stored area to the correct place. The old window definition is restored, too.

OTHER PROCEDURES AND FUNCTIONS

Keywords:

PLOT	FRAME	REFRAME	COLOUR
SETCUR	SET_VAR	VAR	CLEAR
XPOS	YPOS	MESSAGE	BV_BASE
BREAK_ON	BREAK_OFF		

PLOT

SYNTAX:

```
PLOT x,y,col
```

PLOT sets a point at a real position with a defined colour. It works on MC and MCG machines correctly, too.

FRAME, REFRAME

For information refer to the BLOCK HANDLING section (CFRAME, CREFRAME). The stored block is not compressed.

COLOUR

SYNTAX:

```
COLOUR#ch,borderwidth,bordercolour,inkcolour,papercolour
```

COLOUR sets the colours and the border but doesn't clear the window.

SETCUR, XPOS and YPOS

SYNTAX:

```
SETCUR
XPOS
YPOS
```

This interactive command sets a graphic cursor and returns the absolute position in the functions XPOS and YPOS.

EXAMPLE:

```
10 SETCUR
20 PRINT 'Work use cursor keys to position the graphics cursor
and press ESC or SPACE to abort.
30 PRINT 'X-Position = '!XPOS?'Y-Position = '!YPOS
40 GO TO 10
```

SET_VAR, VAR

SYNTAX:

```
SET_VAR
SET_VAR num,var
VAR(num)
```

The command SET_VAR num,var stores a numeric variable in safe memory. Now you can NEW, CLEAR or LOAD a new program and get back the previously stored variable with VAR(num). Only a reset or SET_VAR without parameters will clear it.

BV_BASE

SYNTAX:

```
BV_BASE
```

BV_BASE returns the current base of BASIC_VARIABLES.

CLEAR

SYNTAX:

```
CLEAR
```

This command cleans the memory from fragmentations and does a garbage collection.

BREAK_ON, BREAK_OFF

SYNTAX:

```
BREAK_ON
BREAK_OFF
```

These commands switch the BREAK key (CTRL-SPACE) on or off.

MESSAGE

SYNTAX:

MESSAGE (row,'string$')

MESSAGE is a function similary to INKEY$. It returns the code of pressed key. While the function is waiting for a keypress, the text strings will be rolled on screen at the defined row.

EXAMPLE:

```
10 a=MESSAGE(21,'D I R E C T O R Y   Enter drive number!')
20 SELect ON a
30  =49:DIR adv1_
40  =50:DIR adv2_
50  =RENAINDER:BEEP 2000,20000:GO TO 10
60  END SELect
```