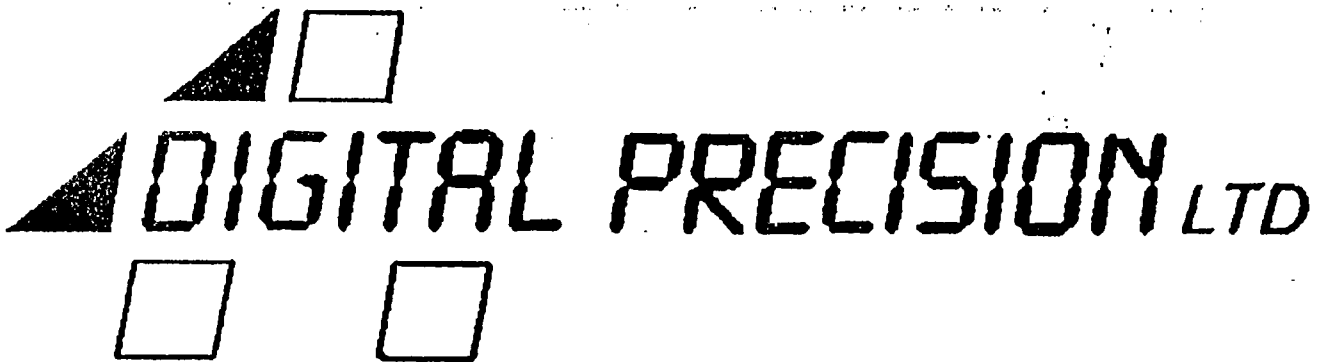


TOOLKIT III



TOOLKIT III

TOOLKIT III for the Sinclair QL computer is designed to enhance the SuperToolkit II (by Tony Tebby) by providing many new facilities and making some of its existing facilities even more powerful.

You will almost certainly already have a copy of SuperToolkit II built into your disk interface. Virtually all the TOOLKIT III extensions, however, do not require you to have SuperToolkit II.

Because TOOLKIT III alters the way in which the QL works a small number of its commands may not be suitable for use on "alien" environments like the ST/QL Emulator, the Amiga/QL emulator and the THOR. Also, its use with "new" device drivers (such as the ones accompanying QL hard disks) is not recommended: make backups of your programs and data before experimenting.

To even further enhance TOOLKIT III buy TURBO, which incorporates the 200 command TURBO toolkit.

If you have bought TOOLKIT III on ROM you get both a ROM and a disk. If you have bought only the disk version and now want the ROM, send us £25.

1. Contents of TOOLKIT III

The following paragraphs contain a comprehensive list of the TOOLKIT III extensions. Default parameters are often included, for ease of use: parameters which may be omitted are enclosed in square brackets []. Commands which require the use of SuperToolkit II are marked with an asterisk *.

Section 2: File Maintenance and Information

The commands in this group all make use of the DATA directory, which may be set or changed by the SuperToolkit II commands DATA_USE, DDOWN, DUP or DNEXT. The current DATA directory may be found by using the DATAD\$ function or executing the DLIST command. Please refer to the SuperToolkit II manual for a description of standard directories.

Commands beginning with a 'W' are wildcard commands. These generate prompts requiring a response from the user, therefore may not always be suitable for incorporation into your SuperBASIC programs.

Commands

* DIR_USE name	sets PROGD\$ and DATAD\$
USER number	selects user number
SETUSER name,number	sets the USER number of a file
SETRO name	set file to READ ONLY
SETRW name	set file to READ/WRITE
SETSYS name	set file to SYSTEM status
SETDIR name	set file to DIRECTORY status
SETDIR_A [dev_]	set all files to DIRECTORY status
SETHOST name	set file to HOST ONLY status
SETNET name	set file to HOST/NET status

Functions

PEND [#channel]	check channel for pending input
CH_BASE [#channel]	find base of channel definition block
WN_BASE [#channel]	find base of window definition block

Section 6: Keyboard Queue Access

These commands enable the user to gain direct access to the keyboard queue. The functions return a value or error code.

Commands

QWAIT	wait for keyboard queue to change
-------	-----------------------------------

Functions

QIN (byte) or QIN ('string')	put byte/string into current queue
QOUT	get a byte from current queue
QTEST	find status of current queue

Section 7: Memory Management and Access

TOOLKIT III contains a group of commands which make a number of forms of accessing memory both easier and faster. By their very nature these routines are 'dangerous': use them with care.

Commands

RESET [value]	reset machine
DOKE\$ address,string	put string at memory address
DOKE_F address,float	put float at memory address
MEMCOPY addr1,addr2,n	copy n bytes from addr1 to addr2
MEMSWAP addr1,addr2,n	swap n bytes from addr1 with addr2

Functions

DEEK\$ (address,length)	get string from memory address
DEEK_F (address)	get float from memory address
BV_BASE	find base of basic variable lists and stacks

Section 8: SuperBASIC Programming and Editing

These commands should assist the user in writing and editing his own SuperBASIC programs.

Commands

BASREF [#channel]	list all SuperBASIC PROCs and FNs
REPLACE [ranges.] oldname,newname	replace all old names with new name
REPLACE\$ [ranges.] old\$,new\$	replace all old\$ with new\$

Section 9: Database Handling

This group of powerful commands enables the SuperBASIC user to write his own database applications using 2 and 3-dimensional string arrays. The functions return a value/position or an error code. The load/save operations and the analytic functions work with any type of array. All arrays handled by the new commands may be sub-arrays of their main dimension.

Commands

SARRAY name,array	save an array to a file
SARRAY_O name,array	SARRAY with overwrite
LARRAY name,array	load an array from a file
SORT array\$	sort 2-dimensional string array
SORT array\$,field	sort 3-dimensional string array by field
SORT_I array\$ [,field]	inverted SORT

Functions

ADIM (name)	find no. of dimensions of a saved array
ADIMN (name,i)	find dimension i of a saved array
ATYP (name)	find type of a saved array
SEARCH (array\$,search\$,start)	search 2-dimensional string array from start for search\$
SEARCH (array\$,search\$,start,field)	search 3-dimensional string array from start for search\$ using field

Section 10: More Extensions to SuperBASIC

There are various other extensions which will be useful in a variety of circumstances.

Functions

INPT\$ ([#channel,][buffer,] string)	edit a string with a buffer
KEY\$ ([#channel,][keylist\$])	wait for a key of keylists
ISINT (string)	return ERR.XP if string is no integer
ISFLT (string)	return ERR.XP if string is no float
UPPER\$ (string)	convert string to upper case
LOWER\$ (string)	convert string to lower case
SGN (integer)	signum function
FRAC (float)	return fraction of a float
ROUND (float)	round to nearest long integer
CINT (integer)	convert integer to unsigned float
ODD (integer)	return 1 for odd value, 0 for even
PRED (integer)	return predecessor of an integer
PRED (char\$)	return predecessor of a character

Functions (continued)

SUCC (integer)	return successor of an integer
SUCC (char\$)	return successor of a character
DIV_L (integer, integer)	longword integer division
MOD_L (integer, integer)	longword modulo function
AND_L (integer, integer)	binary AND for long integers
OR_L (integer, integer)	binary OR for long integers
XOR_L (integer, integer)	binary XOR for long integers

Section 11: Extras

Commands

TK3_EXT	initialise TOOLKIT-III (and SuperToolkit II, too, if present)
DEVLINK	link all additional devices
EXTRAS [#channel]	list all extras linked to SuperBASIC

Functions

QDOS\$	return QDOS version
ISDEV	checks if named device is linked in (ie; "present") or not

Section 12: Extended Device Drivers

All drivers present at RESET time are enhanced to fully support the SuperToolkit II subdirectories (DUP< DDOWN etc) so that every program is bound to a specific subdirectory. The file access byte is used for additional status information in the file header.

Section 13: MEMory Device

Memory can be accessed like a file, using the unique MEM device. In order to allow the use of commands which do not use the file pointer (eg: SBYTES) a relative base address can be specified.

Section 14: Extensions to QDOS

In order to handle the new file attributes directly via QDOS there are some new traps and system variables.

2. File Maintenance and Information

2.1 User Areas

TOOLKIT III is able to handle 16 different user areas, numbered 0 to 15, in order to make file access by different users accessing the same medium (eg; via the Fileserver) more reliable. The user number has to be set by executing the USER command. This user number is written to the file header automatically whenever a new file is saved to a medium. The user number of a file can be changed by the actual owner using the SETUSER or WSETUSER command. The default user number is zero. User areas are ignored when using microdrives.

There are two rules to bear in mind when considering user areas:

- a) Any user can only access files which are in his user area.
- b) System files can be read by any user.

Please refer to the SuperToolkit II manual (which may be part of your disk interface manual) for a description of wildcard names.

Commands which require the use of SuperToolkit II are marked with an asterisk *.

Commands

USER number	selects user number
SETUSER name,number	sets the USER number of a file
* WSETUSER [#channel,] [name,] number	wildcard SETUSER

Examples

USER 4	sets the actual user number to 4
SETUSER fred,7	sets the USER number of 'fred' to 7
WSETUSER#1,fred_.1	prompts all files starting with 'fred_' in channel #1 and sets them to USER 1, if confirmed
WSETUSER 3	prompts all files in command channel and sets them to USER 3, if confirmed

Hints

PRINT PEEK(163894)	returns the actual user number
POKE 163894,10	is equivalent to USER 10

2.2 System files

Files which are set to SYSTEM status are not shown in the directory listing, but can be read by any user. They are particularly useful where several users need to work with the same program (eg; a word-processing utility), because it is not necessary to have a copy of the program in each user area. SYSTEM files are automatically made READ ONLY for all users other than the owner of the file. The SYSTEM status, similarly, can only be changed by the user who owns the file. It cannot be changed via NETWORK. The SYSTEM status is ignored when using microdrives.

Commands

SETSYS name	set file to SYSTEM status
SETDIR name	set file to DIRECTORY status
SETDIR_A [dev_]	set all files to DIRECTORY status
* WSETSYS [#channel,] [name]	wildcard SETSYS

Examples

```
SETSYS john      sets file 'john' to SYSTEM status
SETDIR test_exe  resets SYSTEM status of file 'test_exe'
```

2.3 READ ONLY files

Because it's often more useful to write-protect single files, as opposed to the entire medium, the READ ONLY flag is introduced. If a file has been set to READ ONLY status, it is not permitted to delete the file and any exclusive OPEN will act as OPEN_IN.

Commands

SETRO name	set file to READ ONLY
SETRW name	set file to READ/WRITE
* WSETRO [#channel,] [name]	wildcard SETRO
* WSETRW [#channel,] [name]	wildcard SETRW

Examples

```
SETRO flp2_myfile  sets 'flp2_myfile' to READ ONLY
WSETRW test_       prompts all files starting with 'test_' in
                    command channel and sets them to READ/WRITE if
                    confirmed
```

2.4 HOST ONLY status

If a file is set to HOST ONLY it cannot be accessed via NETWORK.

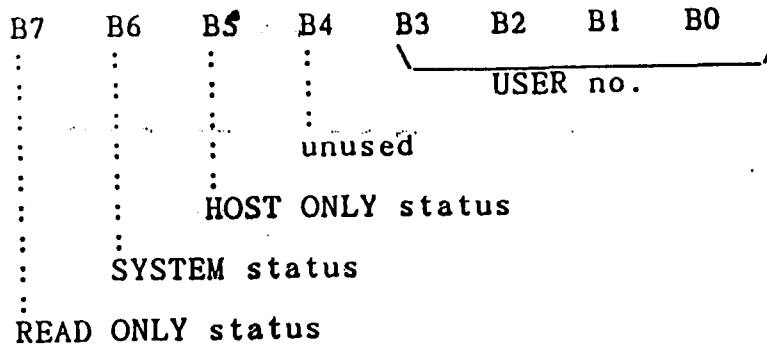
Commands

SETHOST name	set file to HOST ONLY status
SETNET name	set file to HOST/NET status
* WSETHOST [#channel,] [name]	wildcard SETHOST
* WSETNET [#channel,] [name]	wildcard SETNET

2.5 The file access byte

All information about the file status is stored in the file access byte, which is part of the file header. Please refer to a QDOS manual for further information about file headers.

The format of this byte is:



The file access byte is ignored when using microdrives, because it would slow down file access considerably.

Functions

* FACC (#channel) or FACC (\name) find file access byte

Examples

PRINT FACC(\fred) prints file access byte of file 'fred'
PRINT FACC(#ch)&&128 returns READ ONLY status of a file channel
PRINT FACC(#ch)&&64 returns SYSTEM status of a file channel
PRINT FACC(#ch)&&32 returns HOST ONLY status of a file channel
PRINT FACC(#ch)&&15 returns user number of a file channel

2.6 Setting the directory tree

To simplify operations involving PROG and DATA, a new command, DIR_USE, is introduced, which sets both to the same name. The DATA device is used as the subdirectory name.

Please see the SuperToolkit II manual and Section 12 of this manual for further information on subdirectory structures.

Commands

* DIR_USE name sets PROGD\$ and DATAD\$

Commands

RJOB_A remove all jobs except Super-BASIC

Functions

JOBID returns ID of job from which it was invoked

5. Channels

5.1 General channel handling

There are two new commands and one function to access channels which are currently open.

CHANNELS displays a list of all open channels, including the internal channel number, the tag number, the owner job and a detailed description of the channel.

Example

Channel	tag	owner	name	
0	0	0	CON_512x50a0x206	SuperBASIC #0
1	1	0	CON_256x202a256x0	SuperBASIC #1
2	2	0	CON_256x202a0x0	SuperBASIC #2
3	3	0	CON	(Slave Channel)
4	7	1	CON_512x256a0x0	Job 1 console window
5	8	1	RAMI temp	open file of job 1
6	11	0	SCR_100x50a200x100	SuperBASIC screen
7	12	0	PAR	channel to printer
8	13	1	*** ANON ***	what's that?!!!
9	17	2	NSV	Net Server channel

The CLOSE% command enables the user to close a channel using its internal channel number. This is particularly useful when something has gone wrong eg: a channel remaining open after the owner job has been killed. Slave channels should never be closed!

Commands

CHANNELS [#channel] list all open channels
CLOSE% channel close an internal channel

5.2 Pipe handling

Because SuperBASIC is unable to open the passive end of a pipe, a new command is introduced: the CONPIPE command can be used in two different ways.

Type 1: CONPIPE #input TO #output
creates an output pipe and connects it with an input pipe. The output channel must already exist, eg: as a SCR channel.

Example

```
10 OPEN#3,pipe_256      create input pipe
20 OPEN#4,scr           create dummy channel
30 CONPIPE #3 TO #4    connect pipe
40 PRINT#3,'Pipe Test' fill pipe
50 INPUT#3,a$          get string from pipe
60 CLOSE#3,#4          close both channels
```

Type 2: CONPIPE #channel
converts an input pipe to an output pipe

Example

```
10 OPEN#3,pipe_256      create input pipe
20 PRINT#3,'Hello pipe!' fill pipe
30 CONPIPE #3           convert pipe
40 INPUT#3,a$           get string from pipe
50 CLOSE#3              close channel
```

The choice of type will depend on the application. Type 1 is to be preferred when a pipe is needed for more than one operation. Type 2 should be optimal for simple operations. Please refer to a QDOS manual for further information about pipes.

The PEND function checks a channel for pending input, which is normally quite the opposite of an EOF(#ch) call, and is very useful in pipe handling.

Commands

CONPIPE [#INPUT,] #output connect outputpipe with input-
pipe

Functions

PEND [(#channel)] check channel for pending input

The default channel is #1.

5.3 Accessing channel definition blocks

Two new functions are intended to help the user to access the channel definition blocks. CH_BASE returns the base address of the whole channel definition block and WN_BASE returns the start address of the window definition block. Of course, the latter will only return an address when used with a window channel.

On a standard QL the result of WN_BASE is usually CH_BASE + \$18, but if extended screen drivers are used (for example QJump's Pointer Environment) this might not be the case. Please refer to a QDOS manual for further information.

Functions

CH_BASE [(#channel)] find base of channel definition
block
WN_BASE [(#channel)] find base of window definition
block

The default channel is #1.

Examples

```
PRINT PEEK_L(CH_BASE(#ch)+8)
                                returns the owner job ID of a channel
PRINT PEEK(WN_BASE(#ch)+44)
                                returns the paper colour of a window
```

6. Keyboard Queue Access

In order to access the current keyboard queue directly a command and three new functions are introduced. All three functions return an error code; alternatively one of them returns a byte.

Possible return values/error codes are:

0		O.K.
>0		extracted byte (QOUT only)
-1	ERR.NC	queue empty/full
-10	ERR.EF	end of file

QIN puts a byte or string into the current keyboard queue, depending on the type of the parameter. If a string value is desired it should be enclosed within apostrophes.

QOUT extracts a single byte from the current queue and returns it. If the queue is empty ERR.NC is returned.

QTEST simply checks the status of the current queue, without making any changes.

Sometimes it may be necessary to select or re-activate a queue. This can be done by reading the channel for a short time.

Example

```
dummy=INKEY$(#0)    will select the command channel (#0)
```

The command QWAIT is used to detect a change in the keyboard queue. This is particularly useful if you want to start another job using a command string, which should be typed in after the job has loaded.

Example

```
EX flpl_job_exe : QWAIT : dummy=QIN('Hello job...')
```

This line starts 'flpl_job_exe' as a job and types in 'Hello job...' after loading, assuming that this job has an active cursor (and a queue) after loading.

Why not start Quill and automatically load a document...?

Please refer to a QDOS manual for a detailed description of queues.

Commands

QWAIT

wait for keyboard queue to change

Functions

QIN (byte) or QIN ('string') put byte/string into current queue
QOUT get a byte from current queue
QTEST find status of current queue

7. Memory Management and Access

All commands in this chapter should be used with great care!

7.1 Advanced PEEKs and POKEs

Some new commands have been provided in order to make storage of strings and floats in memory easier to handle and faster in operation. We've used DEEK and DOKE instead of PEEK and POKE in order to prevent collisions with TURBO toolkit.

DOKEs and DEEKs are used to store/retrieve a string in/from memory. Since there is no restriction on the type of string they can also be used to store and restore an amount of memory (maximum 32766 bytes). The memory address must be even.

Examples

TOOLKIT III Syntax	SuperBASIC equivalent
10 a\$=DEEK\$(131072,32766)	10 DIM a\$(32766)
	20 FOR n=1 TO 32766
	30 a\$a\$&CHR\$(PEEK(131071+n))
	40 END FOR n

These programs store the actual screen image (except the last two bytes) in a\$. To restore it simply type 'DOKE\$ 131072,a\$' using the TOOLKIT III syntax. Can you spot the difference?

```
10 PRINT DEEK$(49148,PEEK_W(49146))
10 m=PEEK_W(49146):DIM a$(m)
20 FOR n=1 TO m
30 a$a$&CHR$(PEEK(49147+n))
40 END FOR n
50 PRINT a$
```

or simply: 10 PRINT VERS

DOKE_F and DEEK_F enable you to store/retrieve floating point numbers in/from memory using the internal QDOS format (6 bytes).

Commands

DOKE\$ address,string put string at memory address
DOKE_F address,float put float at memory address

Functions

DEEK\$ (address,length) get string from memory address
DEEK_F (address) get float from memory address

7.2

MOVEing and SWAPping memory

Two new commands are intended to make the handling of large amounts of memory easier and faster.

MEMCOPY copies any amount of memory from one address to another (intelligently, if blocks overlap), and MEMSWAP simply does what one might expect: it swaps two blocks of memory.

All addresses and lengths of blocks must be even.

Commands

```
MEMCOPY addr1,addr2,n          copy n bytes from addr1 to
                                addr2
MEMSWAP addr1,addr2,n          swap n bytes from addr1 with
                                addr2
```

Examples

This program rolls the entire screen image:

```
10  FOR n=1 TO 256              256 lines to roll
20  m$=DEEK$(131072,128)        store first line
30  MEMCOPY 131200,131072,32640  roll up 255 lines
40  DOKE$ 163712,m$            put stored line
50  END FOR n
```

The following program turns the screen image upside-down:

```
10  FOR n=0 TO 127
20  MEMSWAP 131072+n*128,163712-n*128,128
30  END FOR n
```

7.3

Accessing the SuperBASIC variable lists and stacks

In order to access SuperBASIC's variable lists and stacks (BV_VARS) it is necessary to know the actual base address. Please refer to a QDOS manual for further information.

Functions

```
BV_BASE          find base of basic variable
                  lists and stacks
```

Examples

```
123 PRINT PEEK_W(BV_BASE+104)    prints the actual line number
100 PRINT BV_BASE+PEEK_L(BV_BASE+24) prints the start address
                                   of the name table
```

7.4

Intelligent RESET

The new RESET performs a standard system reset, but closes all open channels first. Optionally a memory reduction can be done, using multiples of 32K, eg; RESET 128 resets to 128KB.

Commands

```
RESET [value]      resets machine
```

8. SuperBASIC Programming and Editing

8.1 Replacement of names and strings

To assist the SuperBASIC user in editing programs two powerful REPLACE commands have been designed, to replace either names (variable names, device names) or strings. Every comment after a 'REMARK' statement is handled as a string, too. Alternatively a range of line numbers could be specified, using the same syntax as 'RENUM'.

When replacing strings the search string and the replacement string must have the same length. Both must be within quotation marks or apostrophes. The name and string search is case independent.

Commands

```
REPLACE [ranges,] oldname,newname  replace all old names with
                                     newname
REPLACE$ [ranges,] old$,new$       replace all old$ with new$
```

Examples

Before	After REPLACE 20 To 40,a,byte and REPLACE 10,flpl_file,Test
10 OPEN#3,flpl_file	10 OPEN#3,Test
20 BGET#3,a	20 BGET#3,byte
30 a=256-a	30 byte=256-byte
40 BPUT#3\0,a	40 BPUT#3\0,byte
50 CLOSE#3	50 CLOSE#3

Before	After REPLACE\$ 'test','DEMO' and REPLACE Test,Number
10 REMARK This is a Testprog	10 REMARK This is a DEMProg
20 Test=0	20 Number=0
30 REPEAT loop	30 REPEAT loop
40 IF Test=10:EXIT loop	40 IF Number=10:EXIT loop
50 PRINT 'Test Nr. ';Test	50 PRINT 'DEMO No. ';Number
60 Test=Test+1	60 Number=Number+1
70 END REPEAT loop	70 END REPEAT loop
80 REMARK Test End	80 REMARK DEMO End

8.2 Listing PROCedures and FuNctions

BASREF lists all existing SuperBASIC PROCedures and FuNctions.

Commands

```
BASREF [#channel]  list all SuperBASIC PROCs and
                   FNs
```

Example

```
Proc Test           Line 100
Proc Long_Name_Procedure Line 320
FN SuperFunc       Line 1340
```


9. Database Handling

9.1 SAVEing and LOADING arrays

These powerful commands and functions enable the SuperBASIC user to save and load any type of array or subarray (main dimension only). The speed of these operations is comparable with SBYTES and LBYTES, so that several minutes of sequential record loading is a thing of the past. When loading back an array it has to be pre-dimensioned using the same type and dimensions. The main dimension can be changed by loading the array to a subarray.

Three analytical functions have been provided which can be used to find certain information relating to a saved array. These functions return either a value or an error code. If a non-array file is accessed ERR.OR will be returned. Please refer to Appendix A for the format of a saved array.

Commands

SARRAY name,array	save an array to a file
SARRAY_O name,array	SARRAY with overwrite
LARRAY name,array	load an array from a file

Functions

ADIM (name)	find no. of dimensions of a saved array
ADIMN (name,i)	find dimension i of a saved array
ATYP (name)	find type of a saved array

Examples

This program fills an integer array saves it to a file and analyses its structure:

10	DIM i%(100)	dimension array
20	FOR n=0 TO 100	loop to fill array
30	i%(n)=RND(32767)	fill with random integers
40	END FOR n	end of loop
50	SARRAY flpl_test_ary,i%	save array to 'flpl_test_ary'
60	CLEAR	clear memory
70	PRINT ADIM(flpl_test_ary)	print No. of dimensions
80	PRINT ADIMN(flpl_test_ary,1)	print first dimension
	(100)	
90	PRINT ATYP(flpl_test_ary)	print array type
	(3 = integer)	

The following part of a program saves a floating point array, which has been dimensioned using 'DIM f(10,10)' and loads it back into a bigger array. This process could be used to enlarge a dimensioned array.

1000	SARRAY ram1_temp,f	save array to 'ram1_temp'
1010	DIM f(20,10)	enlarge array
1020	LARRAY rem1_temp,f(0 TO 10)	load array as a subarray
1030	DELETE ram1_temp	delete temporary file

The third example is a program which analyses the structure of any type of a saved array:

```

100 CLS:f$=INPT$(36,DATAD$) get filename
110 t=ATYP(f$):SElect ON t analyse type
120 =1:PRINT'Floating Point'
130 =2:PRINT'String'
140 =3:PRINT'Integer'
150 =REMAINDER:PRINT'No array':STOP
160 END SElect t
170 PRINT('';d=ADIM(f$) get dimensions
180 FOR n=1 TO d
190 PRINT ADIMN(f$,n);
200 IF n=d:PRINT')':ELSE PRINT',';
210 END FOR n

```

9.2 Sorting database arrays

A database array has to be a 2 or 3-dimensional string array, the latter having the usual field/record structure. These arrays can be sorted in ascending (SORT) or descending (SORT_I) order.

A type 2 comparison is used to do the sorting, but empty records are always sorted to the end of the array. Please refer to your QL User Guide for further information about comparison types.

Parts of an array can be sorted using a subarray of the main dimension. Nevertheless only string arrays can be sorted: these can be filled with integer or floating point numbers in ASCII format. So there is no need for any other array type.

When sorting 3-dimensional arrays the order field has to be specified using its index number.

Commands

SORT array\$	sort 2-dimensional string array
SORT array\$,field	sort 3-dimensional string array by field
SORT_I array\$ [,field]	inverted SORT

Example

This program sorts a 2-dimensional random character array in ascending order. Only a subarray is sorted because record 0 is used as a header.

```

100 x=19:y=40
110 CLS#0:PRINT#0,'Setting up array...'
120 DIM a$(x,y)
130 a$(0)='SORT Test:'
140 FOR n=1 TO x
150 x$='':FOR m=1 TO y
160 x$=x$&CHR$(RND(65 TO 90))
170 END FOR m
180 a$(n)=x$
190 END FOR n
200 CLS:PRINT a$
210 PRINT#0,'Sorting...'
220 SORT a$(1 TOx):CLS#2:PRINT#2,a$

```

9.3

Searching database arrays

In order to make it possible to write advanced database applications using SuperBASIC, only one key operation is still missing: the SEARCH function enables the user to search a database array for a specified string. If a 3-dimensional string array is used, the search field must be specified. The number of the record from which the search is to start has, in any case, to be specified in order to allow a repeated search, eg: to find the next occurrence of a string.

The search function returns either the number of the matching record or -1 if the string cannot be found. The search is case independent. Subarrays of the main dimension can be used.

Functions

SEARCH (array\$,search\$,start)	search 2-dimensional string array from start for search\$
SEARCH (array\$,search\$,start,field)	search 3-dimensional string array from start for search\$ using field

Examples

a=SEARCH(adr\$,'Miller',0,0)	returns first occurrence of 'Miller' in field 0 of a 3-dimensional string array adr\$
a=SEARCH(adr\$,'Miller',a,0)	returns next occurrence
PRINT SEARCH(num\$,PI,0)	returns first occurrence of PI in a 2-dimensional string array filled with floating point numbers

The following program creates a 2-dimensional string array and fills it with random integers. Then the numbers of all records containing '10' are listed:

10	DIM num\$(100,2)	create array
20	FOR n=0 TO 100:num\$(n)=RND(10)	fill array
30	m=-1:REPEAT loop	loop to search array
40	m=SEARCH(num\$,10,m+1)	search
50	if m<0:EXIT loop	no more matches?
60	PRINT '10 found in record';m	print record number
70	END REPEAT loop	end of loop

10. More Extensions to SuperBASIC

These extensions may or may not be of value to users because most can be written in SuperBASIC, but they are easier to use if present as resident extensions.

10.1 String operations

Four new functions are included which read strings or characters from a console and/or manipulate them.

INPT\$ can be regarded as an advanced INPUT command, allowing a default string and a maximum buffer length to be specified. If no buffer length is specified the length of the default string is used.

KEY\$ scans the keyboard using a given list of characters and returns a character if it is included in the list. When used without any given character list this function is equivalent to INKEY\$(#ch,-1).

ISINT and ISFLT check whether an entire string is convertible to integer/float and return ERR.XP if not. These functions are particularly useful in avoiding the appearance of 'error in expression' when reading numbers using the INPUT command.

UPPER\$ and LOWER\$ convert a string to upper/lower case, including all foreign characters.

The default channel for INPT\$ and KEY\$ is #1.

Functions

INPT\$ ([#channel,][buffer,] string)	edit a string with a buffer
KEY\$ ([#channel,][keylist\$])	wait for a key of keylists
ISINT (string)	return ERR.XP if string is no integer
ISFLT (string)	return ERR.XP if string is no float
UPPER\$ (string)	convert string to upper case
LOWER\$ (string)	convert string to lower case

Examples

a\$=INPT\$(10,'Test') allows editing of the default string 'Test' using the default channel and a buffer length of 10 characters

confirm\$=KEY\$('YyNn'&CHR\$(27))
waits for a character of 'YyNn' or ESC to be pressed

a\$=UPPER\$(INPT\$(#2,36,DATA\$))
reads a filename in channel #2 using a buffer of 36 characters and the DATA device as the default string and returns it converted to upper case

The following program demonstrates the testing of parameters:

```
100 WMON 4:n$=''
110 PRINT'Test FLOAT or INTEGER Input (F/I)?';
120 CURSEN:s$=UPPER$(KEY$('FfIi')):CURDIS:PRINT.s$
130 AT 2,0:IF s$='F'
140 PRINT'Float:':n$=INPT$(12,n$)
150 IF ISFLT(n$):BEEP 3000,1000:GO TO 130
160 ELSE
170 PRINT'Integer:':n$=INPT$(12,n$)
180 IF ISINT(n$):BEEP 3000,1000:GO TO 130
190 END IF
200 PRINT\\'O.K.'
```

Several new functions are built in to enhance or replace the standard SuperBASIC set of mathematical and binary functions.

SUCC and PRED return the successor/predecessor of an integer number or character. They are particularly useful in REPEAT loops.

SGN returns 1 for a positive number, 0 for zero and -1 for negative numbers. The argument has to be an integer.

CINT converts a signed integer to an unsigned float, which can be useful when reading integers from a file.

ODD simply tests whether an integer is odd or not. Please note that 'i=ODD(x)' is equivalent to 'i=x&&1'.

ROUND rounds a float to the nearest long integer, and FRAC returns the fraction of a floating point number.

DIV_L and MOD_L are intended to replace the QDOS operators 'DIV' and 'MOD', but work correctly with negative values and are able to handle long integers.

AND_L, OR_L and EOR_L should replace the QDOS operators '&&', '||' and '^', but work with long integers, too.

Functions

SGN (integer)	signum function
FRAC (float)	return fraction of a float
ROUND (float)	round to nearest long integer
CINT (integer)	convert integer to unsigned float
ODD (integer)	return 1 for odd value, 0 for even
PRED (integer)	return predecessor of an integer
PRED (char\$)	return predecessor of a character
SUCC (integer)	return successor of an integer
SUCC (char\$)	return successor of a character
DIV_L (integer, integer)	longword integer division
MOD_L (integer, integer)	longword modulo function
AND_L (integer, integer)	binary AND for long integers
OR_L (integer, integer)	binary OR for long integers
XOR_L (integer, integer)	binary XOR for long integers

Examples

```

PRINT PRED(100)          99
PRINT SUCC('a')        b
PRINT SGN(-4)          -1
PRINT CINT(-100)       65436
PRINT FRAC(PI)         .141593
PRINT ROUND(10003.5)   10004
PRINT ODD(37)          1
PRINT DIV_L(131072,65000) 2
PRINT MOD_L(131072,65000) 1072
PRINT EOR_L(100000,12345) 112281

```

The following (slow) program inverts the screen image:

```
10 FOR n=131072 TO 163839 STEP 4
20 POKE L n,EOR_L(PEEK_L(n),-1)
30 END FOR n
```

11. Extras

11.1 Linking the extensions

To link all SuperBASIC extensions and/or device driver extensions to the system two commands are provided:

TK3_EXT initialises all new SuperBASIC commands and functions including those from SuperToolkit II, if present, so TK2_EXT is no longer needed.

DEVLINK links all additional directory devices to the extended device driver system. Normally this is done at RESET time, but if any new devices (eg; RAMPRT from QJump) are loaded this command must be used to make them fully compatible with the new features. DEVLINK is executed automatically whenever TK3_EXT is used.

During the linkage several commands are redefined for compatibility: FLP_USE, FLP_TRACK, FLP_START, FLP_SEC, RAM_USE and NFS_USE.

Anyone currently using a FLP_OPT command must now use the new FLP commands.

The 'WIN' device is not supported because hard disks have their own advanced device drivers.

Commands

TK3_EXT	initialise TOOLKIT III (and SuperToolkit II, too, if present)
DEVLINK	link all additional devices

11.2 System information

EXTRAS lists all resident extensions to SuperBASIC, including their type (PROCedure or FuNction) and their decimal start address.

The QDOS\$ function returns the internal code number of the QDOS version in use, eg; 1.13 for MG or 1.10 for JS.

Commands

EXTRAS [#channel]	list all extras linked to SuperBASIC
-------------------	--------------------------------------

Functions

QDOS\$	return QDOS version
ISDEV	checks if named device is linked in (ie; "present") or not

12. Extended Device Drivers

All device drivers, which were present at RESET time or have been linked by DEVLINK or TK3_EXT, are extended to make full support of the SuperToolkit II subdirectory structures. Originally these are only usable from SuperBASIC, but TOOLKIT III extends them to work properly at every access level, eg; machine code, TRAPS etc.

Please refer to the SuperToolkit II manual (which may be part of your disk interface manual) for a detailed description of subdirectory structures and directory control.

The DATA device is used as the subdirectory name. It can be changed using one of the commands DATA USE, DUP, DDOWN, DNEXT or DIR_USE. The latter is introduced by TOOLKIT III: DIR_USE sets the DATA and PROG device, so it's an easy way to change them both.

The subdirectory structures work via NETWORK using the NFS device, which can be specified by executing the NFS_USE command. Please refer to the SuperToolkit II manual for a detailed description.

A file from the root directory can be accessed from any directory level by separating the device name from the file name using a backslash, eg;: FLP1_\FULL_NAME. In SuperBASIC this name has to be enclosed in quotes or apostrophes.

Please note that subordinate directories must not have the same name, eg; TEST_TEST_CODE is not allowed. This is necessary because the operating system must be able to distinguish the directory levels.

It is a good idea to give the start program of each subdirectory the name 'BOOT', because it is then possible to achieve a list of all subdirectories using a line similar to 'DIR flp1_boot'.

Example

1. Suppose you require to copy all files from FLP_2 to a subdirectory named 'GAME' on FLP1_, you can use the following line:
WCOPY flp2_,flp1_GAME_ and confirm with 'A' for ALL
2. To start the copied game using the BOOT program, the following line can be used:
DDOWN game : LRUN flp1_boot

13. The MEMORY Device

It is possible now to access memory as a file using the 'MEM' device. The file pointer position is used as an address. This address is relative to a base address to allow operation with commands which do not use the actual file pointer (eg; SBYTES). If not specified this base address is 0.

Syntax: MEM[_K[_B]] where K*1024+B gives the base address of the MEM device.

Examples

The following program gets the current MODE of network station 2:

```
10 OPEN#3,N2_MEM      open MEM device with base 0
20 BGET#3\163892,m    get MC status register
30 PRINT m&&8         extract MODE and print it
40 CLOSE#3           close channel
```

This line transfers the current screen image to network station 1:

```
SBYTES N1_MEM-128,131072,32768
                                base address is 128K = 131072
```

The following program sets the user number of network station 3 to 7:

```
10 OPEN#3,N3_MEM_160  open MEM device with base 160K
20 BPUT#3\54,7        set user number to 7
30 CLOSE#3           close channel
```

14. Extensions to QDOS

In order to handle the new file attributes and the file access byte directly via QDOS, some new TRAP #3 functions are introduced:

```
FS.SETRO DO = $4C      set READ ONLY status
FS.SETSS DO = $4D      set SYSTEM status
FS.SETUS DO = $4E      set USER number
FS.SETHO DO = $4F      set HOST ONLY status
```

Call parameters

Return parameters

D1		D1	undefined
D2.W	=0 reset or <>0 set or user number	D2	preserved
D3.W	timeout	D3	preserved
A0	channel ID	A0	preserved
A1		A1	undefined
A2		A2	preserved
A3		A3	preserved

There are also a few system variables:

```
SV.USER EQU $36      user number (byte)
*
SV.PROG EQU $AC      pointer to PROGDS (long)
SV.DATA EQU $B0      pointer to DATADS (long)
SV.DEST EQU $B4      pointer to DESTDS (long)
*
SV.QTK3 EQU $DC      internal use only (long)
```


APPENDIX A

Saved arrays have a special header at the beginning of the file. This header has the following structure:

Fileposition	0	--- ATK3	(string)
	4	--- Array Type	(word)
	6	--- Offset to data	(word)
	8	--- No. of dimensions	(word)
	10	--- Index 1	(word)
	12	--- Multiplier 1	(word)
	...		
	XX	--- Index x	(word)
	XX	--- Multiplier x	(word)
		Offset --- Start of data	

The following example program analyses the structure of a saved array. It could also have been written using direct access to the special header at the beginning of the file.

```

100  WMON 4
110  UNDER 1:PRINT'ARRAY ANALYSER':UNDER 0
120  PRINT'Enter dev name: ';
130  f$=INPT$(36,DATA'D$&'TEST ARY')
140  dd=ADIM(f$):IF dd<0:PRINT'Error: '::REPORT#1,dd:STOP
150  st=FACC(\f$)&&128
160  PRINT'File is: ';
170  IF st=0:PRINT'READ/WRITE'\:ELSE PRINT'READ ONLY'\
180  PRINT'Arraytype is: ';
190  tt=ATYP(f$):SElect ON tt
200  =1:PRINT'String'\:a$='array$'
210  =2:PRINT'Float'\:a$='array'
220  =3:PRINT'Integer'\:a$='array%'
230  =REMAINDER :PRINT'(Unknown)\
240  END SElect
250  PRINT'No of Dimensions: '!dd\
260  DIM dms%(3)
270  FOR n=1 TO dd
280  dms%(n)=ADIMN(f$,n)
290  PRINT'Dimension '!n; ': '!dms%(n)
300  END FOR n
310  com$='DIM'&a$&' ('
320  FOR n=1 TO dd
330  com$=com$&dms%(n)
340  IF n<>dd:com$=com$&', '
350  END FOR n
360  com$=com$&'):LARRAY f$,'&a$&':PRINT#2,'&a$&':CLS#0:
    CLEAR'&CHR$(10)
370  a$=INKEY$(#0):er=QIN(com$):REPORT er

```

APPENDIX B

This little example program prints a complete directory listing including all necessary data on the current printer device:

```
100 WMON 4:UNDER 1:PRINT'FULL DIRECTORY LIST':UNDER 0
110 PRINT\'Which device:':;dd$=INPT$(5,DATAD$)
120 OPEN#3,DESTD$:BPUT#3,27,78,4,15
130 WIDTH#3,120:STAT#3,dd$:PRINT#3
140 OPEN_DIR#4,dd$:count=0
150 REPEAT loop
160 IF EOF(#4):EXIT loop
170 GET#4\count*64+14,a$:IF a$='':GO TO 280
180 PRINT#3,a$;
190 GET#4\count*64,high%,low%:length=high%*65536+CINT(low%)-
    64
200 PRINT#3,TO 40:length;
210 BGET#4\count*64+4,facb,type
220 IF facb&&128:a$='READ ONLY':ELSE a$='READ/WRITE':
    END IF:PRINT#3,TO 50;a$;
230 IF facb&&64:a$='SYSTEM':ELSE a$='DIRECTORY':
    END IF:PRINT#3,TO 64;a$;
240 PRINT#3,TO 78;'USER'!facb&&15;TO 88;'TYPE'!type;
250 IF facb&&32:a$='HOST ONLY':ELSE a$='HOST/NET':
    END IF:PRINT#3,TO 98;a$;
260 GET#4\count*64+52,high%,low%:a$=DATE$(high%*65536+
    CINT(low%))
270 PRINT#3,TO 112;a$
280 BGET#4\count*64+64:count=count+1
290 END REPEAT loop
300 CLOSE
```

Copyright Notice

Copyright 1988 Martin Berndt & Ultrasoft

Copyright 1990 Digital Precision Limited

All trademarks are acknowledged. References within this manual to the products of other companies neither constitute any endorsement of such products or companies or imply any approval by such companies of this product.