Appendix K – Token types

These are all of the *token types* which may appear in tokenised SuperBasic programs.

Token	Value	Description
SPC.B	$80	space token
KEY.B	$81	keyword token
BIP.B	$82	built-in-procedure
BIF.B	$83	built-in-function
SYM.B	$84	symbol
OPS.B	$85	operation symbol
MON.B	$86	mono-operation symbol
SYV.B	$87	system variable
NAM.B	$88	name
SHI.B	$89	short integer
LGI.B	$8A	long integer
STR.B	$8B	string
TXT.B	$8C	text
LNO.B	$8D	line number
SEP.B	$8E	separator
FP.B	$F0	floating point number

Appendix L – Arithmetic package op-codes

Arithmetic operations on the QL can be performed by the *arithmetic package*. This is accessed using the utility routines RI.EXEC (vector $11C) and RI.EXECB (vector $11E). The floating point package accepts two types of op codes (operation codes). Codes between $02 and $30 are true arithmetic operations. The other byte codes are extended to a negative word between $FFFF and $FF31 inclusive, and are instructions to load or store the floating point number to the address given by (A6.L + A4.L + ((opcode OR $FF00) AND $FFFE). The operation is a load if bit 0 of the opcode is clear. The operation is a store if it is set. Loading causes A1 to be decremented by 6 bytes and saving causes A1 to be incremented by 6. The old NOS becomes the new TOS.

Operation codes for the arithmetic package

Code	Name	Change to A1	Operation
00	RI.TERM		
02	RI.NINT	+4	find nearest integer to TOS
04	RI.INT	+4	truncate TOS to integer
06	RI.NLINT	+2	nearest long integer to TOS
08	RI.FLOAT	−4	convert TOS integer to floating point
0A	RI.ADD	+6	add TOS to NOS
0C	RI.SUB	+6	subtract NOS from TOS
0E	RI.MULT	+6	multiply TOS by NOS
10	RI.DIV	+6	divide TOS into NOS
12	RI.ABS	0	positive value of TOS
14	RI.NEG	0	negate TOS
16	RI.DUP	−6	duplicate TOS
18	RI.COS	0	take cosine of TOS
1A	RI.SIN	0	take sine of TOS
1C	RI.TAN	0	take tangent of TOS
1E	RI.COT	0	take cotangent of TOS
20	RI.ASIN	0	take arcsine of TOS
22	RI.ACOS	0	take arccosine of TOS
24	RI.ATAN	0	take arctangent of TOS
26	RI.ACOT	0	take arccotangent of TOS
28	RI.SQRT	0	take square root of TOS
2A	RI.LN	0	take natural logarithm of TOS
2C	RI.LOG10	0	take logarithm to the base 10 of TOS
2E	RI.EXP	0	take exponential of TOS
30	RI.POWFP	+6	take NOS to the power of TOS
00	RI.LOAD		load operand key if bit 0 clear
01	RI.STORE		store operand key if bit 0 set

Appendix M – System variable definitions

```
SV.BASE    EQU $28000 base of system variables

SV.IDENT   EQU $00    identification (word)
SV.CHEAP   EQU $04    base of common heap area (long)
SV.CHPFR   EQU $08    first free space in common heap area
                      (long)
SV.FREE    EQU $0C    base of free area (long)
SV.BASIC   EQU $10    base of basic stack (long)
SV.TRNSP   EQU $14    base of transient program area (long)
SV.TRNFR   EQU $18    first free space in transient prog area
                      (long)
SV.RESPR   EQU $1C    base of resident procedure area
                      (long)
SV.RAMT    EQU $20    top of RAM (+1) (long)

SV.RAND    EQU $2E    random number (word)
SV.POLLM   EQU $30    count of poll interrupts missed
                      (word)
SV.TVMOD   EQU $32    0 if not TV display (byte)
SV.SCRST   EQU $33    screen status (0 = active)
SV.MCSTA   EQU $34    current value of MC status register
                      (byte)
SV.PCINT   EQU $35    current value of PC interrupt register
                      (byte)

SV.NETNR   EQU $37    network station number (byte)

SV.I2LST   EQU $38    pointer to list of interrupt 2 drivers
SV.PLIST   EQU $3C    pointer to list of polled tasks (long)
SV.SHLST   EQU $40    pointer tolistofscheduler tasks (long)
SV.DRLST   EQU $44    pointer to list of device drivers (long)
SV.DDLST   EQU $48    pointer to list of directory device
                      drivers (long)
SV.KEYQ    EQU $4C    pointer to a keyboard queue (long)
SV.TRAPV   EQU $50    pointer to trap redirection table
                      (long)

Pointers to resource management tables

SV.BTPNT   EQU $54    pointer to most recent slave block
                      entry (long)
```



```
SV.BTBAS   EQU $58    pointer to base of slave block table
                      (long)
SV.BTTOP   EQU $5C    pointer to top of slave block table
                      (long)

SV.JBTAG   EQU $60    current value of Job tag (word)
SV.JBMAX   EQU $62    highest current Job number (word)
SV.JBPNT   EQU $64    pointer to current Job table entry
                      (long)
SV.JBBAS   EQU $68    pointer to base of Job table (long)
SV.JBTOP   EQU $6C    pointer to top of Job table (long)

SV.CHTAG   EQU $70    current value of channel tag (word)
SV.CHMAX   EQU $72    highest current channel number
                      (word)
SV.CHPNT   EQU $74    pointer to last channel checked
                      (long)
SV.CHBAS   EQU $78    pointer to base of channel table
                      (long)
SV.CHTOP   EQU $7C    pointer to top of channel table (long)

SV.CAPS    EQU $88    caps lock (word)
SV.ARBUF   EQU $8A    autorepeat buffer (word)
SV.ARDEL   EQU $8C    autorepeat delay (word)
SV.ARFRQ   EQU $8E    autorepeat 1/frequency (word)
SV.ARCNT   EQU $90    autorepeat count (word)
SV.CQCH    EQU $92    keyboard change queue character
                      code (word)
SV.WP      EQU $94    write protect (word)
SV.SOUND   EQU $96    sound status (word)
SV.SER1C   EQU $98    receive channel 1 queue address
                      (long)
SV.SER2C   EQU $9C    receive channel 2 queue address
                      (long)
SV.TMODE   EQU $A0    ZX8032 transmit mode (includes
                      baud rate) (byte)

SV.CSUB    EQU $A2    subroutine to jump to on
                      CAPSLOCK (long)
SV.TIMO    EQU $A6    timeout for switching transmit mode
                      (word)
SV.TIMOV   EQU $A8    value of switching timeout (2
                      characters) (word)
```

```
SV.FSTAT   EQU $AA     flashing cursor status (word)

SV.MDRUN   EQU $EE     which drive is running? (byte)
SV.MDCNT   EQU $EF     microdrive run-up run-down counter
                       (byte)
SV.MDDID   EQU $F0     drive ID*4 of each microdrive (8
                       bytes)
SV.MDSTA   EQU $F8     status 0 = no pending ops (8 bytes)
SV.FSDEF   EQU $100    pointers to file system physical
                       definition (16 long)
SV.FSLST   EQU $140    pointer to list of file channel
                       definitions (long)

SV.STACB   EQU $180    bottom of stack (192 long)
SV.STACT   EQU $480    big stack - NO check (up to here)

SV.TRAPO   EQU 2*($28+2) offset of trap vector table from
           (SV.TRAPV)

SV.IDENT   EQU $D254   green, red, blue and black with flash
                       bits
```

Devicedriver definitions(forthe driver's own pseudo system variables)

```
SV.LXINT   EQU $00     link to next external interrupt service
                       (long)
SV.AXINT   EQU $04     address of external interrupt service
                       (long)
SV.LPOLL   EQU $08     link to next polling interrupt service
                       (long)
SV.APOLL   EQU $0C     address of polling interrupt service
                       (long)
SV.LSCHD   EQU $10     link to next scheduler task (long)
SV.ASCHD   EQU $14     address of next scheduler task (long)
SV.LIO     EQU $18     link to next IO driver (long)
SV.AIO     EQU $1C     address of IO routine (long)
SV.AOPEN   EQU $20     address of channel open routine
                       (long)
SV.ACLOS   EQU $24     address of channel close routine
                       (long)
```


Appendix N – Channel definition block

A channel definition block is set up every time that a new channel is opened. All relevant information about the channel is held within this block, such as the device driver which is to be used, the owner Job, the channel number and channel status.

```
CH.LEN     $00 long    length of definition block
CH.DRIVR   $04 long    address of driver
CH.OWNER   $08 long    owner Job
CH.RFLAG   $0C long    address to be sent when space
                       released
CH.TAG     $10 word    channel ID
CH.STAT    $12 byte    status:

                       0 = OK
                       -1 = A1 absolute
                       $80 = A1 relative to A6
                       negative = waiting

CH.ACTN    $13 byte    stored action for waiting Job
CH.JOBWT   $14 long    ID of Job waiting on IO
*
* extended channel definition for serial queue handlers
*
CH.QIN     $18 long    pointer to input queue (or zero)
CH.QOUT    $1C long    pointer to output queue (or zero)
```

Appendix P – File system channel definition block

File system channel definition block

$18	FS.NEXT⋆	long	link to next file system channel
$1C	FS.ACCES⋆	byte	access mode (D3 on open call)
$1D	FS.DRIVE⋆	byte	drive ID
$1E	FS.FILNR+	word	number of file on drive
$20	FS.NBLOK	word	block number containing next byte
$22	FS.NBYTE+	word	next byte from block
$24	FS.EBLOK+	word	block number containing byte after EOF
$26	FS.EBYTE+	word	byte after EOF
$28	FS.CBLOK	long	pointer to slave block table for current slave block which may hold current/next byte
$2C	FS.FNAME⋆	2+36 bytes	filename
$58	FS.SPARE	72 bytes	spare

Physical definition block

$10	FS.DRIVR⋆	long	pointer to access layer link for driver
$14	FS.DRIVN⋆	byte	drive number
$15		byte	reserved
$16	FS.MNAME⋆	word+10 bytes	medium name
$22	FS.FILES	byte	number of files open

Length is determined by the access layer linkage.

Key

⋆ Initially the blocks are full of zeros apart from those marked with an ⋆ which are filled in by the IOSS

\+ The open routine is not called if a shared file is open to more than one channel. In this case, FS.NBYTE is set to $40 and the fixed information is copied from the first channel open to the file.

Appendix Q – Job control block

addr.	length	name	description
$00	long	LEN⋆	total length of Job control + Job area
$04	long	START	start address on activation
$08	long	OWNER?	Job ID of this Job's owner
$0C	long	HOLD?	pointer to a byte which will be cleared when the scheduler releases the Job (see MT.SUSJB D0=8)
$10	word	TAG⋆	tag for Job as allocated by MT.CJOB
$12	byte	PRIOR?	current accumulated priority. This increments when a Job is active but not executing. It is set to zero when the Job is executing. The scheduler allows the Job with the highest accumulated priority to execute.
$13	byte	PRINC?	this priority increment is the initial priority of the Job. Basic activates Jobs at priority $20.
$14	word	STAT⋆	Job status: 0 is not suspended >0 is number of frames before release 1 is suspended (IO or MT.SUSJB) 2 is waiting for a Job to finish
$16	byte	RELA6?	MSB is set if next trap #2 or #3 has relative addressing (as set by trap #4)
$17	byte	WFLAG?	set if a Job is waiting for this one
$18	long	WJOB?	ID of Job waiting for this to finish
$1C	long	TRAPV?	pointer to trap redirection vectors
$20	8 long		saved values of D0 to D7
$40	8 long		saved values of A0 to A7
$60	word		saved value of status register
$62	long		saved value of program counter

⋆ means value should not be changed
? means value may be changed by a trap, or directly (but with care!)

Appendix R – Common heap header

Name	value	description
LEN	$00 long	length of definition block
DRIVR	$04 long	address of driver to free block when the owner is deleted
⋆ NEXT	$04 long	pointer to next free spae
OWNER	$08 long	owner Job (0 if free space)
RFLAG	$0C long	address to be set when space is released
⋆ CLOSE	$0C	offset of close entry point wrt driver address

Appendix S – Window block definition

Name	value	description
⋆ SD.XMIN	$18 word	window top lefthand side
SD.YMIN	$1A word	
SD.XSIZE	$1C word	window size
SD.YSIZE	$1E word	
SD.BORWD	$20 word	border width
⋆ SD.XPOS	$22 word	cursor position
SD.YPOS	$24 word	
SD.XINC	$26 word	cursor increment
SD.YINC	$28 word	
⋆ SD.FONT	$2A 2⋆long	font address
⋆ SD.SCRB	$32 long	base of screen
⋆ SD.PMASK	$36 long	paper colour mask
SD.SMASK	$3A long	strip colour mask
SD.IMASK	$3E long	ink colour mask
⋆ SD.CATTR	$42 byte	character attributes
	bit 0	underline bit
	bit 1	flash bit
	bit 2	transparent background
	bit 3	XOR characters/graphics
	bit 4	double height
	bit 5	extended width
	bit 6	double width
	bit 7	graphics positioned characters
⋆ SD.CURF	$43 byte	cursor flag, 0=suppressed, >0=visible
SD.PCOLR	$44 byte	paper colour byte
SD.SCOLR	$45 byte	strip colour byte
SD.ICOLR	$46 byte	ink colour byte
SD.BCOLR	$47 byte	border colour byte
⋆ SD.NLSTA	$48 byte	new line status (>0 implicit, <0 explicit)
⋆ SD.FMOD	$49 byte	fill mode (0=off, 1=on)
SD.YORG	$4A float	graphics window origin
SD.XORG	$50 float	graphics window origin
SD.SCAL	$56 float	graphics scale factor
SD.FBUF	$5C long	pointer to fill buffer
SD.FUSE	$60 long	pointer to user defined fill vectors
SD.END	$64	

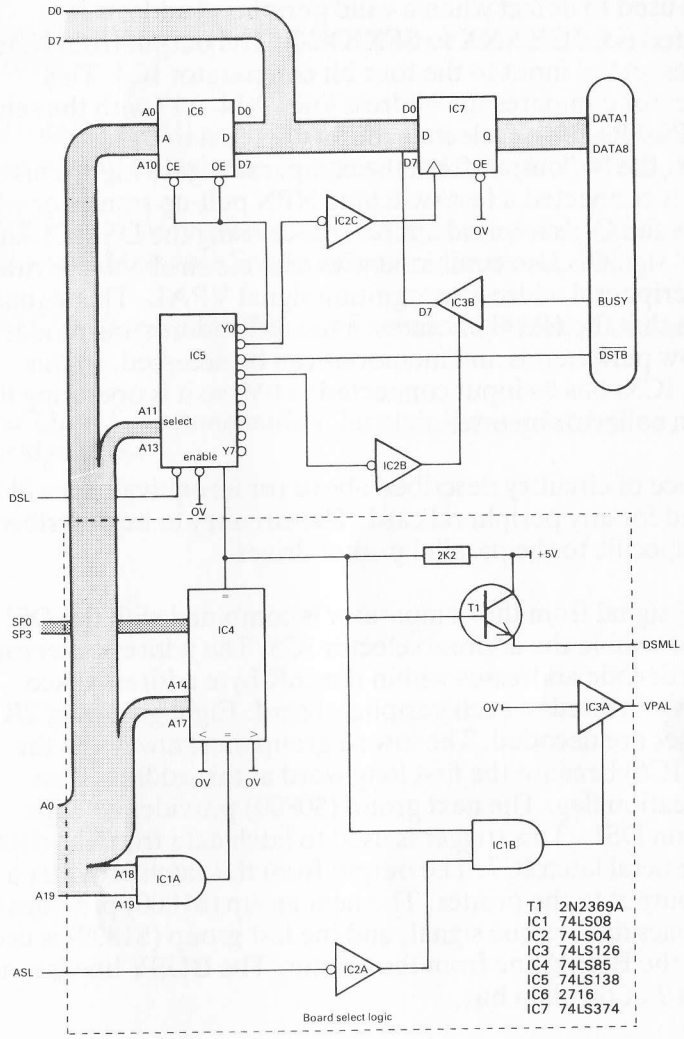
Appendix T – Microdrive backup utility

This short BASIC program allows backup copies of a microdrive cartridge to be made. Unlike most other copy programs, it does not require the names of all the files on the cartridge to be specified in data statements.

The catalogue of entries on mdv2_ is copied into a file called mdv1_dir. Entries are then read from this new file one at a time, and copied across. To keep the operator informed of the current copying status, the program will print out the name of each new file before the transfer starts.

```
100 REM Microdrive backup utility
110 OPEN_NEW #3,mdv1_dir
120 DIR #3,mdv2_
130 OPEN_IN #3,mdv1_dir
140 INPUT #3,a$;a$
150 REPeat cop
160    IF EOF (#3):EXIT cop
170    INPUT #3,a$
180    PRINT "Copying mdv2_";a$
190    COPY 'mdv2_'&a$ TO 'mdv1_'&a$
200 END REPeat cop
210 CLOSE #3
220 DELETE mdv1_dir
```


Appendix U Parallel printer interface circuit



The circuit illustrated above is for a Centronics parallel printer interface. The software which allows this to be operated from QDOS is described in section 9.11.

The lower half of the circuit (or its equivalent) will be required on any QL expansion card. For a description of the standard interface requirements, you should refer to Appendix F.

IC1a is used to detect when a valid peripheral address is generated (ie. $CXXXX to $FXXXX). The output from IC1a provides an '=' input to the four bit comparator IC4. This comparator compares the address lines A14-A17 with the select lines SP0-SP3. If the selected address is equal to the actual address, the '=' output from the comparator goes high. This output is connected a fast switching NPN pull-up transistor which disables the QL's internal address space using the DSMCL signal. The '=' signal is also combined with address strobe to provide valid peripheral address recognition signal VPAL. This signal ensures that the 68008 operates in the 6800 addressing mode, so that slow peripherals and memories can be accessed. In this circuit, IC3a has its input connected to 0V, so it is operating like an open collector inverter.

The piece of circuitry described above (or its equivalent) will be required for any peripheral card. The circuitry to be described now is specific to the parallel printer driver.

The '=' signal from the comparator is combined with the DSL signal to enable the address selector IC5. The address selector is used to decode addresses within the 16K byte address space which is allocated to each peripheral card. Eight groups of 2K addresses are decoded. The lowest group must always be the ROM (IC6) because the first long word at this address is an identification flag. The next group ($0800) provides an edge trigger on DSL. This trigger is used to latch data from the databus into the octal latch IC7. The output from this latch provides a stable output to the printer. The next group ($1000) provides the Centronics data strobe signal, and the last group ($1800) is used to read the BUSY line from the printer. The BUSY line is read onto bit 7 of the data bus.

Note that the VPAL signal will have to be asserted by most peripherals. This puts the 68008 into a 6800 peripheral mode, so that all bus timing is similar to that which would be generated by a 6800. Only by selecting this mode can slow memories or peripheral chips be accessed. It also ensures that certain signals (like the Centronics data strobe) last for more than 500ns.

Bibliography

M68000 Cross Macro Assembler Refernce Manual, Motorola Inc., 1979

M68000 16/32-bit Microprocessor Programmer's Reference Manual, Prentice-Hall Inc., Englewood Cliffs, N.J. 07632, U.S.A

MC68008 16-bit Microprocessor with 8-bit Data Bus, Advance Information, Motorola Semiconductors, Scotland, 1983

Sinclair QL User Guide, Sinclair Research Limited, Cambridge, 1984

Sinclair QDOS Documentation, Sinclair Research Limited, Cambridge, 1984

Glossary

Address Bus – a set of 20 connections, each one of which can be set to logic 0 or logic 1. This allows the CPU to address &FFFFF (1048576) different locations.

Active low – signals which are *active low* are said to be valid when they are at logic level 0.

Asynchronous – two devices which are operating independently of one another are said to be operating asynchronously.

Atomic – all actions which are executed in one *time slice* are said to be atomic. Atomic ations are normally carried out in supervisor mode.

Baud Rate – used to define the speed at which a serial data link transfers data. One baud is equal to 1 bit of data transferred per second.

Bidirectional – a communication line is bidirectional if data can be sent and received over it. The data bus lines are bidirectional.

Bit of memory – this is the fundamental unit of a computer's memory. It may only be in one of two possible states, usually represented by a 0 or 1.

Byte of memory – 8 bits of memory. Data is normally transferred between devices one byte at a time over the data bus.

Channels – are routes through which input and output can be directed. Each channel is defined by a *channel definition* block.

Chip – derived from the small piece of silicon wafer or chip which has all of the computer logic circuits etched into it. A chip is normally packaged in a black plastic case with small metal leads to connect it to the outside world.

Common heap – an area of memory maintained by the operating system. User programs can ask the operating system to give them some space in this area for a variety of purposes.

Data bus – a set of eight connections over which all data transactions between devices in the QL take place.

Device driver – all of the hardware items connected to the QL require some form of interfaceing to the operating system. Device drivers are pieces of software which perform this interface.

Font – a set of alphanumeric characters as displayed on the QL screen.

Handshaking – this type of communications protocol is used when data is being transferred between two asynchronous devices (like the RS232 serial ports). Two handshaking lines are normally required. One of these is a *data ready* signal from the originating device to the receiving device. When the receiving device has accepted the data, it sends a *data taken* signal back to the originating device, which then knows that it can send the second lot of data and so on.

High – sometimes used to designate logic '1'

Interrupt – interrupts are generated by pieces of hardware which wants the 68008 to look at it immediately. External interrupts come from expansion hardware and internal ones from devices inside the QL. One of the most important is the 50/60 Hz video interrupt which operates the Job scheduler.

IOSS (Input/output sub-system) – the software routines associated with basic input and output to devices like the serial ports and microdrive.

IPC (Intelligent peripheral controller) – the *second processor* in the QL which controls the keyboard, serial ports and sound generation.

Job – a program which has been set up to run independently. For example, a Job could be set up to produce a display of the current time on the screen. This Job would work independently from BASIC.

LED (Light emitting diode) – acts like a diode by only allowing current to pass in one direction. Light is emitted whilst current is passed. There are three of these on the front of the QL. One indicates that power is ON, the other two indicate which one (if any) of the microdrives is in use.

Linked list – a data structure used in a variaety of applications. For example, a load of routines which all want ot be called in sequence could be connected togaether with a linked list.

Low – sometimes used to designate logic '0'.

Local variables – variables which are specific to a particular BASIC procedure, but which do not exist outside that procedure.

LSW – abbreviation for 'least significant word'.

Machine code – the programs produced by a 68000 Assembler are machine code. A machine code program consists of a series of words in memory which the 68008 can execute directly.

MSW – abbreviation for 'most significant word'.

Multitasking – the operation of running several programs at the same time. Each program is allowed to run for a short time, then another program is allowed to run for a bit. This ensures that all programs operate for some of the available time.

Open Collector – this is a characteristic of a transistor output line. It simply means that the collector pin of the transistor is not driving a resistor load, ie. it is *open*.

OS – abbreviation for 'operating system'.

Parallel – parallel data transfers occur when data is sent along two or more lines at once. The system data bus for example has eight lines operating in parallel.

Peripheral – any device connected to the 68008 central processor unit, such as the serial port, network interface, microdrives etc., but not including the memory.

Position independent code – 68000 code which will operate properly no matter where it is located within the memory map.

QDOS – the software package on the QL which runs all of the necessary system tasks like getting input from the keyboard, or sending output to the screen. This is the *QL Operating System*.

RAM (Random access memory) – the main memory in the QL microcomputer is RAM. This type of memory will store data and programs whilst power is applied to the machine. This data can be modified at will. All data in RAM is lost when the power is switched off.

Re-entrant – routines which can be executed from within themselves without crashing. Interrupt service routines should normally be re-entrant. To make a routine re-entrant, no fixed memory locations can be used. All variables which are used in the routine must therefore be stored on a stack.

Refresh – all of the RAMs in the QL are dynamic memories. This means that they have to be refreshed every few milliseconds so that their data is not lost. The refreshing function is performed by accesses memory regularly for video output.

Resident procedures – programs which are loaded into the Resident procedures area of memory when the machine is booted. Useful utilities like a BASIC command to convert decimal numbers to hexadecimal might be placed in the resisdent procedure area.

Rollover – this is a function provided on the keyboard to cope with fast typists. Two keys can be pressed at once. The previous key with a finger being removed, and the next key with the finger hitting the key. The software in the operating system ensures that rollover normally operates correctly.

ROM (Read only memory) – as the name implies, ROM can only be read from and cannot be modified by being written to. QDOS and BASIC are held in ROM in the QL.

Scheduler – the scehduler decides which one of the Jobs in the QL should be run in each 20ms time slice.

Self-modifying code – a very bad programming technique in which a program may actually change itself.

Serial – data transmitted along only one line is transmitted serially. Serial data transmission is normally slower than parallel data transmission, because only one bit instead of several bits are transferred at a time.

Extended – operations in 68008 code are used to allow for conversion between bytes, words and long words.

Stack – a block of *temporary storage* memory. Data can be added by *pushing* and removed by *popping*. It is used as a temporary data store when it would be inappropriate to use specific memory locations. There are two system stacks used by the 68008 in the QL. One is used exclusively in *supervisor* mode and the other is used exclusively in *user mode*. The stacks are called the 'supervisor' and 'user' stacks respectively.

Supervisor mode – the 68008 can operate in two modes, user or supervisor. In supervisor mode, several instructions can be performed which cannot be executed from user mode.

Trap – is a 68008 instruction which allows a specific routine to be entered. There are 16 trap calls of which the first 5 are used by QDOS.

ULA (Uncommitted logic array) – these are special chips which contain a large number of logic gates. The connection between the gates is defined when the chip is manufactured. Sinclair has produced two special ULAs for the QL. One of them controls video output and the memory. The other controls the microdrives and network.

User mode – the 68008 can operate in one of two modes, supervisor or user. User mode is the normal mode, but doesn't allow certain destructive instructions to be executed (like RESET).

Vectors – memory locations can be set to contain routine addresses. These are often called *vectors* because the address can be used to redirect (or vector) the program to some specific part of memory.

Index

N

O

P

Q

R

S

T

U

V

W