# Of dongles and toolkits

by Sid Martin and Timothy Green

## *This month a look at Multi-ROM, an automated 'dongle juggler' for the QL owner, and the Z80 Toolkit for all Spectrum programmers*

### Multi-ROM

LURKING round the back of the Sinclair QL is a useful socket – the 'ROM cartridge' connector, which lets you add new code onto the machine's operating system. ROM stands for 'Read Only Memory' – in other words, memory that contains ready-made code and data that cannot be erased, and is instantly available when you turn on your computer.

The QL can handle up to 1024K of memory, but it only lets you plug 16K into the ROM cartridge socket. Other ROMs can be added via an interface for the big connector at the left-hand end of the computer, but that's a tricky business, as we noted in May.

The MCS Multi-ROM is an ingenious alternative to expansion ROM boards. It's easy to use, compatible with all the ROMs currently available for the QL, and adds some very original tricks to the QL's repertoire.

### Home of the ROM

The Multi-ROM plugs into the ROM socket, famous as the home of the 'kludge' on early QLs. When the QL was designed, it was meant to have 32K of built-in ROM software, with space for another 32K of add-on code plugged into the ROM socket. Unfortunately this decision was made in 1983, long before the built-in software was finished.

The built-in code turned out to be more than 32K long, even after it had been pared down to a minimum. The designers tried rewriting the SuperBASIC interpreter, trading speed for space, and chose the familiar terse error messages, but still the code would not fit in the two 16K sockets inside the machine.

Sinclair shipped early QLs with a clumsy add-on board hanging off the back, holding the code it couldn't fit inside the machine. This 'kludge' made it impossible to plug in real ROM expansion, because the socket was already taken up with vital code that should have been inside the machine.

After a while, Sinclair managed to fit the code inside, by the crude but effective technique of soldering the legs of two 16K ROM chips together, piggyback-fashion, and adapting one of the internal sockets to take two chips instead of one. The first reliable release of the QL, version 'AH', used this technique. Later ones took advantage of new and expensive 32K chips, removing the need to stack ROMs.

This left 48K of ROM inside the computer – part of it unused, but inaccessible – and 16K of space for add-ons plugged into the back of the QL. After a while the early machines were upgraded, and by 1985 plug-in ROMs began to appear in ever-increasing numbers.

### Socket to 'em

This led to a new problem for QL enthusiasts. The computer has only one ROM cartridge socket, and several good products are competing for that space.

Every time you want to swap from one ROM to another you have to unplug the computer, change the cartridge, reconnect the power and wait for the memory to be tested. This wastes your time and – after a while – begins to wear out the socket! If you work the socket too hard you can end up with a system that crashes every time you accidentally knock it and rattle the cartridge hanging from the back.

ZX-80 and ZX-81 owners will remember that the 16K RAM on the back of Sinclair's early machines suffered from a similar problem. The QL is not as fault-prone, because ROM cartridges are relatively light and well-supported, but even so it is dangerous as well as irritating to have to keep swapping ROMs as you use your machine.

MCS' Multi-ROM aims to cure this. It's a programmable ROM simulator, which lets you 'change ROMs' very quickly using electronics instead of physical plugging and unplugging.

### Optional extras

In essence there are two different types of product that use the add-on ROM socket. First came operating-system upgrades – routines that add to the facilities available to all QL programs. It makes sense to put these in the ROM socket, so they're instantly available once the machine's turned on, as if Sinclair had built them in from the start.

The best-selling QL ROM is probably QJump's SuperToolkit 2, a comprehensive re-write of QL commands and devices by the original author, Tony Tebby. SuperToolkit facilities will be discussed in a future column, when we plan a comparative review of QL toolkit programs.

Nowadays SuperToolkit is usually bought as part of an expansion system, like Miracle Systems' Trump Card and Sandy's SuperQboard. But there are still lots of things to plug into the ROM cartridge socket, like the CP/Mulator, which enables the QL to run programs designed for old CP/M business systems.

Other operating system upgrades include *ICE*, a plug-in 'Icon Controlled Environment' which lets you delete, copy and view files by pointing to pictures on the screen; *QFlash*, a fast simulated disk that uses spare QL memory; and *SpeedScreen*, the general-purpose display speed-up package we reviewed in March.

Add-on ROMs are popular with users and publishers alike, because they run code very quickly without using up any of the precious RAM on an unexpanded QL.

The snag is that it's hard to use more than one ROM at a time. To make things worse, manufacturers have come up with another use for the ROM socket – as a way of discouraging software theft.

The argument is that programs which come with some of their code in ROM are difficult for the average pirate to steal. High-priced products like MetaComCo C and Pascal, and Prospero's Pro Pascal

## CST Thor XVI specifications

**FOLLOWING on from the review last month, here's the full technical specification of CST's new 'Super-QL' computer, the Thor XVI. For further details call CST on (0438) 352150.**

Main box dimensions:
400 mm wide, 330 mm deep, 85 mm tall.
Main box weight:
6.6 Kilos (about 15 pounds)
Power supply:
220-250v / 110-120v, 50W, 50-60 Hz AC.

Processors:
16 bit 68000, 8 MHz clock – usable
RAM bus bandwidth 3177 KHz.
8 bit 68B02, 2 MHz clock.

Display modes: 512 x 256 pixels, 4 colours & 48 stipples.
256 x 256 pixels, 8 colours & 224 stipples,
plus hardware flashing.
256 x 256 pixels, 16 colours, 896 stipples.

Free memory for tasks, programs and variables:

430K (520K system)
930K (1,032K system)

Maximum system RAM: 6,664K (current limit 2,056K)

Standard devices:

720K 3.5 inch floppy disk, 18K/second max.
2 serial ports, 2K/second max. (but see text)
Multiple window display, 2K c.p.s. & graphics
IBM-AT standard keyboard port.
9 pin mouse/joystick port.
Centronics parallel port.
Local area network, 1.5K/second maximum 63 stations, 8 file servers.
Fast 8 bit Digital to Analogue sound port.

Add-on devices available now:

GPIB IEEE-488 instrument control interface.
Speedscreen display driver, max. 12K c.p.s.
INTROM re-entrant editor and command toolkit
QEP Intelligent EPROM programmer.
SCSI winchester disk interface for 1-16 20
or 40 megabyte drives, top speed 140K/second.
Extra 3.5", 3" or 5.25" floppy disk drives.
Most QL expansion cards (but not QL RAM).

Speeds are in characters per second (c.p.s.); 1K = 1024 c.p.s.

Built-in ROM software:

ARGOS multi-tasking operating system.
Extended Sinclair SuperBASIC
9 digit floating-point maths pack
Network file server.

Bundled software (on disk):

Psion Xchange integrated business package:
Quill word processor
Archive database
Abacus spreadsheet
Easel business graphics

and Pro Fortran compilers, refuse to work unless special ROM routines are plugged into the back of the computer. This sort of protective device is known as a 'dongle'.

**Automated dongle juggling**

At this point keen QL users might find it useful to learn to juggle. Armed with Toolkit ROMs, speedup ROMs, dongles and miscellaneous plug-in widgets, you find the more ROMs you buy the more difficult it is to use them.

Enter Micro Control Systems (MCS) and Mark Snape, inventor of the Multi-ROM. His gadget is a small un-labelled black plastic box, a little over twice the size of a standard ROM cartridge. The box comes with a six page A5 manual, and a mysterious bit of black wire with spring-loaded connectors at either end.

Inside the box there's a ROM control program, 16K of RAM – 'Random Access Memory' which can be written, as well as read – and sundry clever bits. The unit is well-made, and guaranteed for two years after purchase.

The 16K RAM pretends to be a ROM. You can load any normal ROM code or data into it, without plugging or un-plugging anything. The clever bits fool the computer into believing that you're using a standard ROM.

Normally the memory at the ROM connector can only be read, but never written-to. Multi-ROM removes this limitation.

Programs can turn the ability to write on and off, so that normally-harmless writes to the cartridge port don't mess up the RAM contents once they've been set. This protection is vital, because many cartridges deliberately try to overwrite their own code, to stop people running stolen code in RAM.

You control whether the 16K RAM can be altered by reading two memory addresses that are monitored by Multi-ROM but not normally used by the QL. After PEEK(49086) you can write to the 16K RAM, until you use PEEK(49084) which makes it appear to be ROM, protecting it against corruption until you turn off the power.

**Hardware hacking**

You can't just plug in Multi-ROM and start work, because the QL circuit board doesn't supply a 'write' control signal to the cartridge port. MCS has made it fairly easy for you to add this wire; you don't need a soldering iron, as it supplies a pair of spring-loaded clips which hook inside the machine to make the required connection.

The first step is to undo eight screws, to expose the circuit board of the computer. The wire runs between the big connector on the left and the cartridge socket at the back. At first we got it in the wrong place; nothing worked, but the QL survived till we found the right position. There's a useful diagram in the manual, but you still have to read the instructions carefully.

The clips are easy to use, and hook on securely, but when you come to re-assemble your QL you may find it a tight fit if you've got a Schoen replacement keyboard or internal RAM expansion.

When all is well you plug Multi-ROM into the cartridge port. Your ROM cartridges still work as normal with the wire fitted, but Multi-ROM can't work without it. At first it behaves like a small Toolkit ROM, announcing its name when you reset the machine and adding a few extra commands.

**Frills**

The WARMSTART command resets the machine but by-passes the normal QL RAM test, getting you straight to the F1/F2 startup display without delay.

Multi-ROM also lets you 'warm-start' the machine from the keyboard, by pressing the keys CTRL-ALT-7 – a combination which otherwise crashes the machine. There are rare occasions when these keys don't have the desired effect, due to quirks of the communications mechanism between the QL's two processors, but generally it's a useful time-saver. Unfortunately you can't use these keys to interrupt and speed up a normal reset.

WARM128 has a similar effect, but the machine resets as if it was a standard 128K QL, fooling badly-written programs like *Psion Chess*, which normally won't work on an expanded system. The system forgets about expansion memory until you type RESET or press the button behind the microdrives, triggering a normal, slow reset.

The EXIST function tells you if a file or device exists or is busy. Other toolkits include such a function under names like FOPEN or DEVICE-STATUS.

The rest of the commands are concerned with the 'Defaults Device Driver' – a patch that stops the computer saying 'not found' when you forget to supply a device name, by supplying a 'default' device when no device is explicitly indicated.

For instance you might enter DEFAULT MDV2-. After that, SAVE WALES creates the file MDV2-WALES, rather than giving an error report. The default name can be up to 16 characters long.

SuperToolkit can do similar tricks, but the MCS version has the advantage that it works with machine-code programs, like MetaComCo's editor and compilers, as well as most BASIC commands and functions. That's because it patches the operating system device list rather than the BASIC command definitions.

The bad news is that DEFAULT does not affect DELETE or FORMAT operations, because they don't use the OPEN vector which DEFAULT intercepts.

You must teach the Default Device Driver the names of any odd devices on your system, so it knows not to tack the default in front of them. Standard devices, like CON, SER, RAM, MDV, FLP, and PAR are pre-defined. Commands and functions let you set and check both the default and standard names.

**ROM images**

The toolkit commands are useful, but they're incidental to the main purpose of the Multi-ROM, which is to liberate your ROM library. Multi-ROM can load any one ROM into its own 16K RAM, for perfect compatibility without hardware hassle.

Multi-ROM can also convert some ROMs to run – at reduced speed – in normal expansion RAM, rather than the standard socket. Not all ROMs can be converted – QFlash and most of SuperToolkit run OK in RAM, but ICE, CP/M and the dongles only work if loaded at the correct address. *Speedscreen* ROMs are supplied with slower RAM versions, for times when your socket is clogged.

You must take an 'image' of each ROM on disk or microdrive before you can use it with Multi-ROM. This is the most laborious part of setting up Multi-ROM, but you only have to do it once. You plug in each ROM in turn, switching the computer on and off between times, and save each ROM image with this command:

```
SBYTES "filename",49152,-
16384
```

Finally you plug in Multi-ROM, and use the LOAD-ROM command to select one of the ROM image files. LOAD-ROM takes a ROM image and puts it into Multi-ROM as if you'd plugged it into the back of the computer.

The machine resets if there was a different image in Multi-ROM earlier. If it did not do this the system would crash when it tried to use the original, now over-written code. There's no reliable way to tell the QL that an add-on has been replaced, unless the machine resets.

The good news is that the reset is almost instant. You don't have to wait for memory to be tested and the screen to clear – the F1/F2 menu appears at once.

You can still only run one ROM image at a time in Multi-ROM. But swapping is easy because of the fast reset and the way you select the next ROM by loading its image from tape or disk, rather than by shuffling hardware.

Oddly enough, Multi-ROM is particularly useful if you make your own ROMs with an EPROM programmer. It means you can test code intended for the cartridge port almost as soon as you've written it.

There's no need to blank a chip by cooking it in an ultra-violet lightbox, and no delay waiting for the EPROM programmer to do its work. You just load the new code into Multi-ROM. This is really useful if you're prototyping a 16K cartridge, although not much use if you're developing a bigger ROM program with a ROM code generator like Q-Liberator.

**Clever but costly**

Multi-ROM is a good solution to a technical problem which might not have existed if Sinclair and QL software suppliers had liased more closely. It's not the ideal solution – you should be able to run any number of ROMs at once – but that's impossible, because some ROMs will only work at standard ROM socket addresses. Multi-ROM is the next best thing.

If you make your own ROM cartridges, or own several expensive utilities, Multi-ROM may be worth your while, especially if you've given up using some packages because of the hassle of swapping ROMs. The snag, for most potential users, is the price – at £49 plus VAT, you must have a substantial investment in QL ROMs before Multi-ROM is worth buying.

**Contact**

Micro Control Systems: Electron House, Bridge Street, Sandiacre, Notts NG10 5BA. (0602) 391204.

**Spectrum Z80 toolkit**

One of the worrying things about the Spectrum market in recent years has been the shortage of cheap, easy-to-use software tools for budding programmers.

Almost all commercial Spectrum software is written in Z80 machine code, the language of the Z80A processor at the heart of the computer. Yet Z80 programming tools are harder to find than they have ever been. This must be limiting the availability of new Spectrum software, by discouraging people from learning the Spectrum's native language.

Lerm Software, long-time specialists in tape-to-tape and tape-to-disk transfer programs, has seen a gap in the market and produced *Z80 Toolkit* – a complete Z80 programming system.

It takes a while to become proficient in machine code, but the results are worth waiting for. In general it's best to learn ZX BASIC before you start on machine code – many of the principles are the same, but BASIC is more tolerant of mistakes.

Machine code is famously easy to crash – many mistakes stop the machine accepting further commands – but *Z80 Toolkit* contains an impressive 'stepper' which reduces the risk of the machine

*Continued on page 82*

*Continued from page 81*

running out of control when you test your programs.

*Z80 Toolkit* costs just £7.99, which makes it worth consideration by any Spectrum owner who'd like to know more about how the machine works, or wants to get the best possible performance from their computer.

**The package**

*Z80 Toolkit* comes on cassette, with a 24 page A5 manual. It is divided into two main parts, each of which loads from cassette in about a minute.

The 'Assembler' part lets you enter and edit machine code programs in the form of standard mnemonics – code words – which look like gibberish for a while but soon become familiar. This mnemonic language is called 'assembly language', and forms a bridge between all computers with Z80 processors – Spectrums, PCWs, CPCs, Einsteins, Memotechs, MSX machines and many other 8-bit stalwarts.

The 'Toolkit' part can be loaded with the Assembler, or separately at the top or bottom of memory. It can convert machine-code back into mnemonics – a process known as disassembly. It also lets you edit memory and processor contents, and can run machine code in slow motion – about 1:100,000 of full speed – so you can see the effect of a program, step by step, as you test it.

Both parts, and the data they generate, are handled as standard CODE files, so it's easy to store them on faster devices if you've got the hardware.

This is the first serious program we've seen which runs happily on any kind of Spectrum, from an old 48K machine – perhaps with some obscure add-on storage system – to a new 128K Plus Three. The disadvantage of this generality is that you must manipulate files using BASIC SAVE and LOAD commands, rather than commands built-in to the Assembler.

In practice this is not much of a chore. People with microdrives or non-standard disks will be glad to find a program that takes full advantage of their fast storage, even if it does make them do a bit of typing every so often.

*Z80 Toolkit* works with all printers that recognise the ZX BASIC command LPRINT – in other words, just about any printer that plugs into the Spectrum. It will run in 48K or 128K mode, though you don't get any extra space for code or text on a Spectrum 128.

Numbers can be entered in decimal, as normal, or base 16, known as hexadecimal or just 'hex'. Sometimes it's useful to write numbers in hex, because each digit represents exactly half a byte. Bytes are the fundamental unit of storage in most computers – one byte usually represents a single character.

*Z80 Toolkit* is fluent in both hex and decimal – you mark hex values with a # sign at the start. All numbers are treated as 16 bit unsigned values – in other words, you can only enter whole numbers between 0 and 65535. Output can use hex or decimal, as you choose. *Toolkit* displays are instantly re-written in the new base whenever you change it.

**The Assembler**

The Assembler lets you enter and edit programs, then converts them into machine code in memory. It lacks professional features like macros and conditional assembly, but that doesn't matter much unless you're writing large, complicated programs.

The figure 'Assembler screen display' shows a screen display produced by the Assembler. The display is initially black on white, the Sinclair default, but a command lets you change it to white on blue, or back again.

Normal-sized characters are used, giving 24 scrolling lines of up to 32 characters each. You can move the cursor anywhere on the screen with the arrow keys, typing over the top of existing text, or adding or deleting characters, moving the rest of the line back and forth.

Letters appear in capitals unless you press Shift – the reverse of normal. The annoying pauses of the 128 BASIC editor are absent, thankfully.

Every time you press Enter the current display line is entered as if you'd just typed it. This makes it easy to change lines or to copy them with minor variations. Beware of making changes and forgetting to enter them, because you can't tell whether or not a line has been entered by looking at the screen.

Every line has a five digit number, used to keep it in order. Text can be re-numbered in even steps, from any line to the end, and the editor will write the line numbers for you if you're entering a large group of lines.

You can put several instructions on one line, separated by colons, but lines are limited to 32 columns so it's neater to keep each instruction to a line of its own. SHIFT 2 works like a Tab key, jumping across the screen to fixed columns.

Other operations use one-letter commands, followed by optional line-numbers or just Enter. Unrecognised letters cause a terse 'Error' report. The commands are listed in Table 2 – notice that some are in a 'monitor' section, so you have to type M before using them.

The editor can handle files created by 'Zeus', the official Sinclair Assembler. A few changes are needed before Zeus files will assemble correctly.

Program text can be merged by overwriting the end-mark of one file (the last two bytes) with the next file. You do this by loading the CODE for one file at the end of another, and renumbering if necessary.

You can have as many program texts and assembled code blocks in memory as you can be bothered to keep track of. There's no check for overlaps, and about 23K of memory unused when the complete system is loaded.

The manual tells you the addresses of each part of the package, and explains how you can separate the Assembler from the Toolkit, allowing up to 32K of text to be entered and edited in one file. This is useful, as mnemonics invariably need more memory space than machine code.

## Lerm Z80 Toolkit commands

*The Z80 Toolkit is controlled with single-letter commands, listed rather chaotically in the 24-page A5 manual. Here's a full list, in alphabetical order.*

**Main menu command keys:**

A Call up Assembler
B Return to ZX BASIC
N NEW and return to ZX BASIC
T Call up Toolkit

**Assembler command keys:**

SHIFT 0 Overwrite delete
SHIFT 1 Clear display line
SHIFT 2 Tab to next column
SHIFT 3 Delete & close up
SHIFT 4 Insert space
SHIFT 5-8 Cursor movement
SHIFT 9 Clear screen

A Assemble text to memory
D Delete block of lines
F Find specific text in block
I Insert auto-numbered lines
L List text lines
M Enter Monitor mode
N Set new address for text
O Retrieve old text
P Turn printer on or off
Q Quit to main menu
R Renumber from a line to end
S Show symbol table labels
T Show text address & length
X Execute assembled code

**Assembler monitor keys:**

M A 16 bit dec/hex conversion
M A # 16 bit hex/dec conversion
M B 2 byte dec/hex conversion
M B # 2 byte hex/dec conversion
M C Toggle display colours
M F Search symbol table
M L Show last text line Number
M O Set new default text origin
M P Turn printer on or off
M S Move symbols to/from screen
M T Tabulate memory values/text
M Z Return to Assembler
M SYM X Remove Toolkit from memory

**Machine code Toolkit, control keys:**

A Set new address
B Exit to ZX BASIC (asks Y/N)
C Auto-step till space typed
D Dump text contents of memory
E Swap to alternate register set
H Select hex or decimal display
L Remove logo from top 2 lines
M Return to main menu
N Retrieve old address
O Old address, clear registers
P Printer on/off
Q Change register/stack values
R Clear registers (except IY/SP)
T Trap all store instructions
U Upper/lower case disassembly
V Reset screen colour attributes
Z Enter disassembler:
Z A Alter value of a single byte
Z C Automatic disassembly till space
Z Q Quit from disassembler
Z Z Modify bytes in hex or decimal

*Continued on page 84*

*Continued from page 82*

The *Toolkit* – 8,400 bytes of code – was written this way, and Lerm explains the tricks that were used in order to assemble such a large program on a standard Spectrum. It would be difficult to assemble a file much larger than that in one go with *Z80 Toolkit* – but you can do an awful lot in 8K of machine-code, as each instruction only occupies 1-5 bytes of memory after assembly.

**Operations**

*Z80 Toolkit* recognises all the standard Z80 instructions, plus 'directives' like DEFB, DEFW and DEFM, for storing numbers, memory addresses and messages in assembled programs.

Explanatory comments can be written alongside instructions, or on lines of their own. They must start with a semi-colon, so that the assembler knows to ignore them.

EQU lets you give names, or 'labels' of up to six characters to numeric values. This makes programs easier to read and edit. Capitals and small letters are distinguished, so Newt2, NEWT2 and newt2 are three different labels.

Most assemblers let you use label values in calculations, worked out when the text is assembled. *Z80 Toolkit* can add or subtract a numeric value from a label, but that's all.

Labels in the left hand column of a program automatically take the value of the address of the instruction alongside. The symbol $ stands for the address of the current instruction, so LABEL EQU $ is just the same as LABEL at the start of a line of its own.

Labels, and matching values, are filed away in a 'symbol table' which you can display or print out. The table normally runs backwards through memory, from the start of the Assembler towards your text. You can move it temporarily into screen memory when assembling very large programs.

You tell the computer where you want the code to work with an ORG (short for ORiGin) directive. Sometimes that might get in the way of your program text, so you can specify an alternative address where the code is to be stored after generation, with DISP (short for DISPlacement).

ENT marks an 'ENTry point' into the code. After assembly the X command makes the processor call the code of the line after ENT.

You assemble the program by typing A. The code is scanned twice, and errors are reported by line with 10 rather general error numbers. Assembly is fast, as no file access is needed.

**Toolkit**

The Toolkit is designed for testing programs produced with the Assembler, although you can also use it to examine Sinclair's ROM or other people's programs.

## Assembler screen display

```
L
00100 ; LERM Z80 Toolkit review
00110 ; Computer Shopper 9/88
00112 ; Manual example program
00115          ENT
00120 START    LD        B,255
00130          LD        HL,ATTRS
00140          XOR       A
00150 LOOP     LD        (HL),A
00160          INC       HL
00170          INC       A
00180          DJNZ      LOOP
00190          RET
00200 ;
00210 END      EQU       $
00220 ATTRS    EQU       22528
00230 LENGTH   EQU       END-START
S
LENGTH   = 000C    = 12
END      = 6A76    = 27254
LOOP     = 6A70    = 27248
ATTRS    = 5800    = 22528
START    = 6A6A    = 27242
A -
```

*This shows a typical display while using the Z80 Toolkit assembler. The L command lists the program text, while S shows the 'symbol table' from the last assembly. This printout was produced using Romantic Robot's MultiPrint gadget, which lets you print or save any screen at the press of a button.*

The Toolkit lets you examine and alter memory values, a byte at a time. You can't enter text directly, but must type the appropriate numeric character codes.

The two main parts of the Toolkit are the single-stepper and disassembler. The stepper lets you test code in slow motion – it obviously can't handle interrupt routines, which rely on the computer running at full speed, but it's fine for testing most programs.

The Stepper display is shown in the figure 'Front Panel'. It's the best such display we've seen on the Spectrum, although it's a shame it only shows the current instruction and doesn't use label names from the symbol table. All the commands in the second part of Table 1 are available as you step through code.

You can tell the Stepper to start running code from any address. Instructions flash by, with the rest of the screen changing as values change inside the computer. The Toolkit uses its own fast display printing code, so it can re-draw the screen about five times a second, but even so it takes over 30 seconds to step right through Sinclair's own character-output routine.

You can slow things down by pressing the space key to execute each instruction. Alternatively the Toolkit can 'trap' all instructions that modify memory contents, and wait for you to type Y or N before executing or ignoring them. This simple feature is very useful.

You have to return to BASIC if you want to call routines at full speed. It's a shame that you can't set 'breakpoints' so that code runs at full speed until a certain point is reached, and then returns to the toolkit. It would also be useful to be able to search memory – as opposed to the text or symbol table – for particular values.

The disassembler produces mnemonics from code in memory. It lets you choose between capital or small letters, hex or decimal, but can't tell data from code and does not produce labels.

Lerm says the next version will include a new disassembler program called *C-TO-S*, which converts blocks of code – up to about 6K long – into mnemonics in memory, with arbitrary labels on lines that are referenced by other lines. *C-TO-S* can skip any data-areas in the code, as long as you tell it where they are.

**Further reading**

The 24-page *Z80 Toolkit User Manual* is not particularly well-organised, but it does the job and most information can be found in a minute or so. After the contents, introduction and loading details there's a nine page tutorial section explaining how programs are entered and edited with the assembler and monitor.

The next five pages deal with the Toolkit, used when examining and testing assembled programs. Then come three pages summarising the assembler and monitor control keys, and a list of the 10 assembler error-numbers, with a single-line explanation of each code. There's no summary of the Toolkit or Disassembler commands, apart from lists mixed in with other observations in the previous section.

About 100 instructions work on all Z80 processors, but are not officially documented. *Z80 Toolkit* can disassemble most of these into consistent mnemonics, listed in the manual. The assembler only recognises standard mnemonics, so if you want to use these instructions in a program you must enter them as byte values; the appropriate numbers are in the manual.

The last two pages introduce 'some useful ROM routines' – Sinclair routines which you can use in your own programs. The list contains useful code snippets to print characters, messages and numbers, make beeps, pause for a short time and wait for a key-press.

## Front Panel

```
Z80 TOOLKIT © LERM SOFTWARE 1987

03201 : res 4,(iy+002)    23612

   OPERATE INSTRUCTION Y or N

A'   8   BC'      0   (BC)'243   I    0
B    0   DE'17408   (DE)'  16   X    0
C    0   HL'15700   (HL)'  62   I   92
D   68   IX        0   (IX)  243  Y   58
E    0   IY    23610   (IY)  255
H   61   BC        0   (BC)  243   STACK
L   84   DE        0   (DE)  243        0
A    4   HL        0   (HL)  243        0
                                        0
B 00000000   C 00000000                 0
D 01000100   E 00000000                 0
H 00111101   L 01010100                 0
A 00001000   F 00001000            5630
  76543210     SZ-H-PNC            2780
                                   6177
STACK POINTER   51221  >>>>>  16384
ITEMS ON STACK     5
 TRAP ON                 PRINTER ON
```

*The 'front panel' display of Z80 Toolkit – it looks a muddle, but in use it's clear and comprehensive: underneath the title the next instruction is dis-assembled – the number 23612 is the effective address of the indexed RES instruction. It has been 'trapped', because it modifies memory – hence the question on the next line. The next block shows the contents of registers, individually and in pairs, plus the values they point at. Stack contents are shown on the right, and move up and down as values are pushed and popped. Alongside, the main registers and flags are shown in binary. On the TV some lines are emphasised by shading, but this doesn't show in the screen printout*

The manual contains only two examples, but they're well-thought-out and presented in tutorial style.

'Assembler screen display' shows the display while using the assembler, with the first example program listed. There are a couple of small deliberate errors in the listing in the manual, which goes on to explain how you can find and correct them.

The first example fills the top part of the screen with flashing colours. The second tells you how to trace through the ROM routine that prints a character on the screen, one step at a time. Even at top speed, the stepper takes over 30 seconds to trace the code needed to print one character.

The detail is not explained, but it's still a good example. You can see the character appear, line by line, as the routine runs in slow motion. Instructions flash by and values move up and down the stack.

The manual is a tutorial about *Z80 Toolkit*; it's rather chaotic as a reference guide, although it doesn't take long to learn the commands if you work through the tutorial. There's no on-screen 'help', as that would reduce the memory available for text, symbols and assembled code.

**Read before write**

The manual definitely won't teach you Z80 assembly language – it doesn't even list the processor's instruction-set. You'll need at least one extra book if you're new to the subject.

There are two things you must know beside how to use the Toolkit – how the Spectrum memory is organised, and how the Z80 programming instructions work.

Once there were quite a few books teaching Z80 code specifically for the Spectrum, so you could learn both from one book. Some of the classic titles are listed in the panel. There were dozens of other titles in a similar vein – some good, most poor, a few dreadful.

The market was flooded with books in 1983-4, and has reacted by abandoning the entire genre, so these specialist titles are becoming rare now. You may have to make do with a general book on the Z80 – available from most technical bookshops – in conjunction with a Spectrum programming manual. You'll need the old orange manual, or the +2 or +3 versions – but not the glossy Spectrum Plus booklet, which contains hardly any technical information.

**A good tool**

Extra features would be nice, but *Z80 Toolkit* is a useful and practical package, as long as you're familiar with the first principles of machine-code programming and don't want to write a program longer than, say, 2-3,000 lines of code.

*Z80 Toolkit* works fast, and has the feel of a program that has been cleanly designed and modified in the light of experience. At £7.99 it should encourage many new Spectrum owners to dabble in Spectrum machine code – and in the long run that should be good news for all Spectrum users.

**End of file**

That's all for this month – but we'll be back next issue with more information and reviews. We welcome readers' letters about any aspect of the Sinclair Scene. We can't offer personal replies, but we'll answer some letters in the column, and offer a free subscription for the best letter featured each month.

**Contact**

Lerm Software: 11 Beaconsfield Close, Whitley Bay, Tyne and Wear NE25 9UW. (091) 253 3615.

*Sid Martin and Timothy Green are freelance journalists and Sinclair aficioniados.*

## Spectrum machine code book list

**A list of some recommended books on Spectrum machine code programming.**

*Understanding your Spectrum* by Ian Logan (Melbourne House)
*The Working Spectrum* by David Lawrence (Sunshine)
*Mastering Machine Code on your ZX Spectrum* by Toni Baker (Interface)
*40 [illegible] machine code routines for the ZX Spectrum* by John Hardman and [illegible] Hewson (Hewson)
*Spectrum +2 Machine Language for the Absolute Beginner* by Joe Pritchard (Melbourne House)
*The Complete Spectrum ROM Disassembly* by Ian Logan and Frank O'Hara (Melbourne House – a good reference but not for beginners)
*Z80 and 8080 Assembly Language Programming* by Kathe Spracklen (Hayden – not Spectrum specific)
*Programming the Z80* by Rodnay Zaks (Sybex – not Spectrum specific, but extremely comprehensive)

## Club Contact

**THIS part of Sinclair Scene lists the better Spectrum, QL and Z88 user gropus, with contact addresses. Send a stamped addressed envelope to the address shown to find out more about groups that interest you.**

**If you are in a group that you think we should be listing, please ask the organisers to put *Computer Shopper* on the group's mailing list, so that we can assess the group and mention it in future issues.**

BetaBASIC Newsletter
24 Wyche Avenue, Kings Heath, Birmingham B14 6LQ.
Newsletter for users of Spectrum BetaBASIC – an extension to Sinclair's ZX BASIC. Editor: Andy Wright.

RAMM! Music Machine Owners Club
1 Hillcrest Court, Shoot-up Hill, London NW2 3PG
Independent club for owners of the Spectrum Music Machine, made by RAM Electronics. Bi-monthly newsletter, software, sound library. Boss: Al Straker.

INDUG – Independent Plus D/Disciple User Group
34 Bourton Road, Gloucester GL4 0LE
Group supporting users of Miles Gordon Technology 'Plus D' and 'Disciple' Spectrum disk systems. Monthly newsletter, GDOS bug-fixes, software library, telephone help and program conversion advice. Over 800 members. Boss Rob Brenchley is now running this group full-time.

QLAF – The QL Adventurer's Forum
Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA
Specialist user-group for QL owners interested in adventures, simulations, strategy and wargames. Publishes a newsletter and specialist programs. Boss: Richard Alexander.

QUANTA – the Independent QL User Group
24 Oxford Street, Stony Stratford, Milton Keynes MK11 1JU
Long-established QL support group, with about 2,000 members. Runs local and national meetings with a massive library of software, mostly free to members. Boss: Alex Tegg.
Quanta also offers telephone help and a newsletter, nationally published monthly, edited by Roy Barber. The newsletter is several issues behind, following a management shakeup and legal threats from a large software house concerned about poor reviews, but it's catching up.

Q-LAVE – the Spanish QL user group
P.O. Box 403, Zaragoza 50080, Spain.
Group for Spanish-speaking QL users – about 400 members. Magazine (in Spanish) and software library.

QL Super User Bureau
PO Box 3, Shildon DL4 2LW
QL Super User Bureau is a commercially-run support group for users of the Sinclair QL, CST Thor, ICL One Per Desk and BT Tonto. Newsletter, reviews, discounts and telephone help service, Sunday to Thursday 1pm-8pm.

Z88 User's Club
68 Wellington Street, Long Eaton, Nottingham NG10 4NG
Independent club for Cambridge Computer Z88 users. Offers a bi-monthly club magazine, software library, reviews, discounts and fascinating information about undocumented features of the machine. About 700 members. Boss: Roy Woodward.