Sinclair Scene

Timothy Green reports on SAM, Spectrum, QL and ZX-81, plus Sinclair emulators on the ST and Amiga

I summarised *SAM Basic* in the March review, but have found out lots more since. This month I'll discuss some of the new features, neat tricks, and bugs!

The program listing *Change* (on page 155) demonstrates the elegance of the language. *SAM Basic* allows structured design – a method of programming that encourages reliable, readable code. Old Basic is fine for short programs, but gets hard to read and test as programs expand. Long listings become a web of numbers, GO TOs and GO SUBs – and SAM allows programs up to 480K.

GO TOs can be used to write loops or complicated conditional tests, but it's easy to make mistakes that way. *SAM Basic* lets groups of lines be controlled and listed as blocks, with automatic indentation to emphasise the context of each statement.

IF THEN ... END IF makes a single block of lines conditional. IF THEN ELSE END IF selects between two blocks. You can add ELSE IF, with a new test, to select between three or more alternatives.

FOR loops are fine if you know how many iterations you want before you start, but sometimes you need a loop that stops, or restarts, when certain conditions crop up. *SAM Basic* executes lines between DO and LOOP repeatedly. EXIT IF gives a conditional jump past the LOOP, while LOOP IF can send you back to the DO.

LOOP UNTIL goes round and round till the condition after UNTIL is satisfied. DO WHILE loops unless the condition after WHILE is false. You can mix these together, or use LOOP WHILE or DO UNTIL, and write any loop without GO TOs.

SAM Basic will run old programs, but it's better to use structured programming for new routines. Blocks can be written inside other blocks, without limit.

Procedures are self-contained blocks of code. Once you've learnt to use procedures you can write your own library of Basic commands, and use them like standard keywords.

Procedures are often built up from calls to other procedures. You can pass 'parameters' to and from Basic procedures, just like standard keywords. Parameters can take a 'default' value if no explicit value is supplied.

Often procedures use 'local' variables, which are independent of other variables which may have the same name. Local variables vanish when the procedure has finished.

Hardware whizz Bruce Gordon distracts SAM Basic author Dr Andy Wright from precious moments with the first Coupe prototype (on the desk)

Parameters and local variables mean that you can call a procedure at any time, without worrying what variable names it uses inside, or what variables are used by procedures it may call. Communication can be confined to the parameters specified when the procedure is called, so it's easy to see the dependency between routines.

Normally, only the value of a parameter is passed to a procedure. The code inside can use the value by referring to the corresponding name on the DEF PROC line.

Sometimes we need to pass a value out of a procedure. We indicate this by putting REF before the name in the DEF PROC line. If the name has a REF, changes to the variable inside the procedure will affect the corresponding parameter outside. Otherwise, they won't.

Extras

SAM Basic has two undocumented commands. WINDOW without parameters resets the margins to use the full screen. OPEN 2 allocates two extra 16K pages for Basic – but what if you want to set the size to a fixed value? OPEN TO 2 does the trick, allocating 2*16384 bytes to Basic – or try OPEN TO 10, for 163840 bytes. Remember to CLEAR a valid address first.

The secret functions BAND and BOR allow bitwise operations. BIN 11001100 BAND BIN 11110000 gives 192, or 11000000 in binary. In hex &82 BOR &35 gives &B7. The new display settings OVER 2 and OVER 3 combine pixel palette numbers similarly.

Bugs

Like any new program the first SAM Rom has bugs, but the language is flexible so it is easy to avoid trouble if you know what's going on. The VAL function works out arbitrary expressions, which may include variables and functions, like EVAL in *BBC Basic*. It takes PRINT VAL('COS(PI)+2') in its stride. However, VAL only works at the start of an expression, on ROM 10, so you may need temporary assignments to get correct results.

Some string comparisons give the wrong answer if the strings match. Avoid problems by writing B$ > A$ instead of A$ < B$, and B$ <= A$ for A$ >= B$. IF THEN ELSE usually works fine, but can stop unexpectedly in some programs; the cure is to put ELSE IF 1 instead of ELSE. READ and INPUT get confused in long programs. You can avoid trouble by putting DATA at the start, or use INPUT LINE a$ instead of INPUT a$.

Dr Andy Wright is working on a 'final' Rom which will nail these bugs. MGT's Mark Summerfield tells *Shopper* that it will be sent out free to all existing customers at the end of May, after exhaustive testing. Disk users will get an upgraded SAMDos, with extra Spectrum compatibility and speed.

Free demo

The listing *Change* illustrates the power of *SAM Basic*. It is a short routine to be MERGEd with existing programs. It can search through any range of lines and replace any program text you specify. For instance:

```
CHANGE 'GO SUB 2000','TIDY',2000,3000
```

scans all lines from 2000 to 3000 and replaces GO SUBs with calls to PROC TIDY.

You can use *Change* to convert old Basic into structured code. You might replace the Spectrum's 'RANDOMIZE USR' with SAM's 'CALL', or swap line-numbers for LABEL names. You can rename variables, but be careful – if you change a name like 'n', every letter 'n' will be replaced!

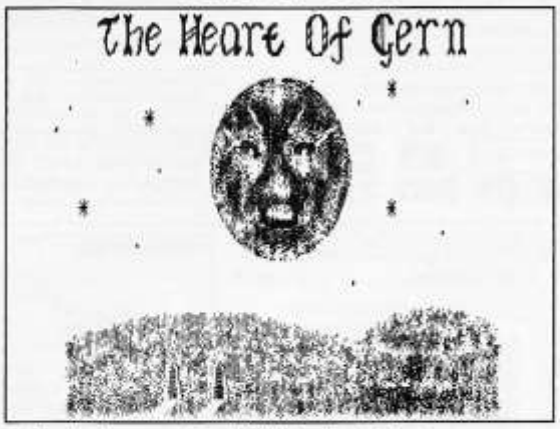
If the replacement line is not valid Basic, *Change* stops with an error report. It could use ON ERROR, but there's little point – it might as well stop at once if your command has ungrammatical consequences.

You don't need to supply all the parameters. CHANGE 'fred' just lists all the lines containing the text 'fred'. By default, *Change* scans lines from 1000 to 64999; if you use it repeatedly it uses the last range explicitly specified.

Change uses RECORD and LLIST to get each line into the string A$, and KEYIN to enter the changed version.

UPDATE searches the line in A$ for OLD$ and replaces it with NEW$. If the new string includes the old one it can only replace one instance on each line; ▶155

HEART OF GERN



The loading screen from PCBS' authentic QL adventure, as stretched and recoloured by TASCOPY and Tim's old Epson!

like error handling, the floating-point calculator and memory allocation are explained from first principles.

Other sections discuss Rom bugs, jargon and typos in the *Disassembly*, but you don't need the old book – the last 90 pages of the *Index* list the Rom code, with labels but no comments. The *Index* presumes familiarity with Z80 code and the Spectrum manual, but nothing else.

The alphabetical index is exhaustive. It catalogues every data-structure, keyword and symbol, and all calls to each routine, with entry and exit conditions. It documents every access to system flags. It even tells you all the places in the Rom that the IX register is set or read! There's some repetition of similar sentences, but the text is not gratuitously long, and everything is there for a reason. Presentation is amateurish, with 90 lines of dot matrix print packed onto each page; even then, photocopied onto thin paper, *The Index* is over an inch thick! I had problems with faint or missing pages in the review copy, but Francis Miles has promised to chastise his local copy shop and reprint some pages.

The Index to the Spectrum Rom is clearly-written and will be invaluable to anyone writing or hacking Spectrum machine code. It's a long, fascinating read for Z80 and ZX enthusiasts. Thanks (and a *Shopper* Guru mug) to reader SM Goodman for bringing it to my attention.

Photocopies cost £25, delivered in a loose-leaf binder, but you can get it cheaper on five 5.25" Disciple disks, bundled with a version of *Word Manager*. If you want the disks you should phone Francis Miles first to let him know what printer interface you use. Blurb and representative sample pages are available if you send an SAE.

Legoland

Danish correspondent Leif Mortensen has sent us the new issue of his fanzine *Sinclair Freakeren*. It has 36 A5 pages, including a detailed review of the SAM Coupe, plus news, advice and programs aimed mainly at Danish Spectrum fans, but mentioning the QL and ZX-81 too.

Leif is busy porting 8Mb of ST clip art, reading Atari disks with a homegrown CS disk system on his Spectrum. Apparently a band of Danes has started to design a Spectrum hard disk interface.

Emulators

Leif also enclosed a ZX-81 emulator for the Atari ST. He enthused about an American Pascal compiler for the ZX-81, and enquired hopefully after a Spectrum emulator for the ST and a SAM version of PCG's *WordMaster*.

PCG expects to have converted Spectrum *WordMaster* to use SAM's 85-column MODE 3 display by the time you read this. The price will be the same as the Spectrum versions – £11.90 for the word processor, or £37.80 for the full DTP system.

Overseas characters will still be a problem, as on the Spectrum. PCG plans to ignore SAM's standard accents and use those character codes for screen layout graphics. Eventually it intends to swap to *Ventura* codes so that pages can be output on PC laser printers.

I hope to test and report on the ZX-81 emulator in a future issue. In the meantime I have passed the disk on to Softville for distribution.

I have not heard of a Spectrum emulator for the ST but I expect the main problems would be cassette loading and emulation of attribute graphics. Leif is keen to hear from anyone with a ZX-81 and a serial interface, as he has lots of ZX programs he'd like to try on the ST. Write to: Sinclair Freakeren, Bryggervangen 29, 7120 Vejle 0st, Denmark.

Existing CP/M emulators prove that you can run Z80 machine code on the ST at a reasonable speed, but many Spectrum and ZX-81 programs rely on hardware idiosyncracies like the display; this runs in real-time and it's very hard to emulate time-critical operations on another processor. That said, it should be easy enough to downgrade the ZX-81 emulator to impersonate a ZX-80.

MGT's SAM Coupe seems a better bet for ZX-80 and 81 emulation. It should be possible to run images of Sinclair's Roms in the bottom 16K area, as the Spectrum emulator works at the moment. Display output would need tweaks but could probably be handled using SAM's screen interrupts to time each line, writing to the palette to control TV or monitor output.

If anyone shows enough interest, SAM might be persuaded to emulate other Z80 machines, like the Memotech, Einstein or MSX models. SAM's MODE 1 and MODE 2 are a reasonable match for MSX's SCREEN 1 and 2. Sprites and font animation would have to be emulated in software, but SAM's Z80B has power to spare.

Rainer Kowalik's Public Domain QDos emulator for the Amiga is picking up a keen following among QL enthusiasts. At the time of writing, the current version dates from late 1989, but an upgrade is promised in mid March so I have decided to postpone a full review until Softville can supply the latest version.

The 1989 emulator does an excellent job, despite a few rough edges in disk and keyboard handling. The blitter copies the QL screen in real-time, without slowing down the 68000. As it does this it separates the QL MODE 4 data into distinct red and green 'playfields' to suit the Amiga.

Timing differences mean you may see brief coloured flashes as white displays are scrolled, and the red and green information is updated separately, but you can minimise these – at the expense of 68000 speed – by boosting the priority of the blitter. Alternatively, you can select red or green ink on black paper, and scroll away at full speed without flicker.

The emulator leaves 203K for *SuperBasic* on a standard Amiga 500, but takes advantage of extra memory or interfaces if you have them. It supports the Amiga 2000 coprocessor 'bridge board', real-time clocks and expansion memory, but not hard disks – as yet.

You must run the utility program TAS_REPLACER to replace TAS instructions in QL code, as the Amiga hardware can't cope with a fast 'read/modify/write' memory cycle; thereafter the emulator runs literally every QL program I have tried, including *Turbo*, *Flightdeck*, *The Editor* and the Psion packages.

At the time of writing, Amiga QDos can read QL floppy disks, but it is slow because the code reads a complete track for each sector required. The disk code is hacked from CST source files, originally written by Tony Tebby, and needs revision. Formatting is slow and unreliable and the QL cannot read disks once the Amiga has written to them.

Even so, it's an impressive piece of work and amazing value at £3, or £9 for the full set of disks. Maybe these niggles will be fixed in the new release. ■

Club Contact will be back next month.

CONTACTS

Amiga QL emulator
ST ZX-81 emulator
Softville, Unit 5, Stratfield Park, Electra Avenue, Waterlooville, Hants PO7 7XN.
(0705) 266509

QL Heart of Gern
PCBS, 60 Highburgh Road, Dowanhill, Glasgow G12 9JN.

Spectrum Rom Index
FG Miles, Windrush, Rabley Heath, Welwyn, Herts AL6 9UF.
(0438) 813406

SAM WordMaster
PCG, 61 School Street, Barrow in Furness, Cumbria LA14 1EW.
(0229) 36957

CHANGE

```
100 REM CHANGE by Andy Wright & Timothy Green March '90

110 DEF PROC CHANGE old$,new$,first,last
120   LOCAL matches,changes,ok,changed,lnum
130   DEFAULT new$=old$,first=1000,last=65299
140   LET matches=0,changes=0,ok=0,changed=0,lnum=first
150   DIVERT
160   DO
170     LLIST lnum TO lnum
180   EXIT IF a$=""
190     LET lnum=VAL (a$( TO 5))+1,a$(6)=" "
200   EXIT IF lnum>last+1
210     PRINT PEN 2;a$;
220     UPDATE changed
230     IF changed
240       CHECK ok
250       IF ok
260         LET a$=a$( TO LEN (a$)-1),changes=changes+1
270         PRINT a$
280         KEYIN a$
290       END IF
300     END IF
310     LET a$="",changed=0
320   LOOP
330   REVERT
340   PRINT "Scan complete, "matches" matches, "changes" lines changed."
350 END PROC

370 DEF PROC UPDATE REF found
380   LOCAL f
390   DO
400     LET f=INSTR(a$,old$)
410     IF f
420       LET found=1, matches=matches+1
430       LET a$=a$( TO f-1)+new$+a$(f+LEN (old$) TO )
440     END IF
450   LOOP UNTIL f=0 OR INSTR(new$,old$)
460 END PROC

470 DEF PROC CHECK REF status
480   LOCAL n,n$
490   LET status=1
500   IF FN ROM_VERSION<=11
510     RESTORE names
520     FOR n=1 TO 8
530       READ n$
540       IF INSTR(a$,n$) THEN LET status=0
550     NEXT n
560     IF status=0 THEN PRINT "Unable to KEYIN "!a$
570   END IF
580   LABEL names
590   DATA "READ","INPUT","DEF FN","DEF KEY"
600   DATA "GET","KEYIN","DELETE","RENUM"
610 END PROC

620 DEF PROC DIVERT
630   LET lf status=DPEEK SVAR 14
640   LET vector=DPEEK FN CHANS(25)
650   POKE SVAR 14,255,0
660   DPOKE FN CHANS(25),DPEEK FN CHANS(20)
670   RECORD TO a$
680   RECORD STOP
690 END PROC

700 DEF PROC REVERT
710   DPOKE SVAR 14,lf status
720   DPOKE FN CHANS(25),vector
730 END PROC

740 DEF FN ROM_VERSION=PEEK 15

750 DEF FN CHANS(n)=DPEEK SVAR 59+n
```

This routine can change 'water' into 'wine', or vice versa! MERGE it at the front of your SAM Basic, then use the CHANGE command to search and replace inside your program

otherwise it changes every occurence of OLD$.

The original KEYIN cannot enter a few commands, so the procedure CHECK excludes them. ROM 10 gets confused if you KEYIN a line before the line being executed, invalidating compiled line addresses, so you should MERGE *Change* before the lines you want to alter.

DIVERT redirects LLIST output to stream #16. RECORD TO A$: LIST #16 sends the listing to A$, but tokens are not expanded so it's hard to edit or KEYIN the result. Andy Wright's DPOKE in the CHANS area links RECORD to LLIST, which expands keywords into ASCII.

REVERT resets the printer channel, line-width and end of line code, so you can LPRINT normally after *Change*. LPRINT and LLIST use channel "p", which strips out control codes, so multi-coloured flashing listings don't confuse the printer! You must open a special channel to print codes less than 32. OPEN #4,"b" lets you print any code with PRINT #4. Channel "b" passes control and keyword codes unchanged.

Heart of Gern

This QL adventure was first published in 1987. It received good reviews but few sales, costing £17. The original publisher, PCBS, still has loads lying around in a Glasgow garret, so it has cut the price to £7 and sent *Shopper* a review copy.

Heart of Gern is a large text adventure written with the QL version of *The Quill* adventure-writing tool (unrelated to Psion's *Quill*). The eponymous Heart is a magic jewel that traps a nasty 'spirit wolf'. The game consists of about 60K of text, packed with hidden halls, enchanted discs, ghastly undead, swords, armour, statues, secret doors, passages and trick stairs.

It's very much in the Dungeons and Dragons mould, with consistent, claustrophobic settings and many deadly traps. You start out at night, among wolves sent by the evil Brotherhood. When day dawns you spy two pillars, and a great cliff. The right magic word opens up a fissure in the rock, and like a true enthusiast you rush straight in...

Heart of Gern is presented like a semi-professional boardgame. It comes in a small vinyl wallet with a map, glossary and a few pages of instructions in small type, backed by a heavily-inked cartoon strip which sets the scene and provides vital clues.

The text is typical of the genre – descriptive and consistent, but occasionally prolix or ungrammatical. The main window fits on a TV screen but uses the smallest QL characters, in red on green.

Short commands are encouraged – only the first four or five letters of words are checked. You can edit entries as normal, with arrows and the CTRL key, but insertion and deletions stop working if you pass column 30.

With a compressed loading screen the game just fits one microdrive cartridge and a 128K QL. The main code block is about 80K long. You can copy the files to another device but you need the master cartridge in MDV2 as you load. This is annoying for disk users, but it is quite easy to circumvent this 'copy protection' with standard QL commands.

Saved games normally go to MDV1 but you can redirect them to floppy disk with the *SuperBasic* command FLP_USE MDV. Unfortunately, the game is loaded with LBYTES and CALL, rather than EXEC, so it won't multitask. The text is encoded so you can't cheat by copying the file to the screen – unless you XOR each byte with 255 first.

Heart of Gern is a difficult game, aimed at experienced adventurers. At the start, movement in any direction prompts the response "Stumbling around in the dark is not a good idea," and you're stuck unless you scour the documentation for clues. The HELP command is unhelpful and there is no SCORE to indicate how you're getting on – it's all or nothing. If you found *Starplod* hard, avoid this. It's more tricky than *Valagon*.

Issue 4 of the excellent *QL Adventurer's Forum* contains hints, and you can get more help from the designers, by letter or phone. If you're looking for a challenging QL adventure, *Heart of Gern* should fit the bill. Hooray for Scotland and its QL strategists!

Index to the Spectrum Rom

Every Spectrum – from the first 16K models to the latest Plus Three – contains about 15K of densely-packed machine code. This is the Rom program that interprets *ZX Basic*, and it has stayed virtually unchanged since 1982.

The Spectrum Rom was developed by Steve Vickers, and derived from the 4K and 8K interpreters written for the ZX-80 and ZX-81 by John Grant of Nine Tiles. It's very clever, and very complicated. The Rom is packed with neat routines to perform calculations, control interfaces and do other useful things.

Programmers can save much time and trouble by calling Rom routines, instead of filling memory with reinvented wheels. The problem is finding out where the routines are, what they do, and how to call them reliably.

In 1983, Melbourne House published *The Complete Spectrum Rom Disassembly* – a book listing each line of code in the Rom, with comments. Many people found the *Disassembly* useful, but it was dense, sometimes poorly explained, and full of typing mistakes. It is now unobtainable.

Enthusiast Francis Miles has spent six years compiling his own guide to the Spectrum Rom, while developing the *Word Manager* package. The resultant *Index to the Spectrum Rom* runs to 460 densely-packed A4 pages, and goes far beyond the scope of the *Disassembly*.

Every keyword, system variable and label from the original *Disassembly* is explained in the *Index*. Important concepts ▶159