More on the scientific computer

Further details of the monitors

By J. H. Adams, M.Sc.

After publication of the scientific computer series (April to September 1979) there have been many requests for more information on the firmware. This article describes in more detail the machine code and BURP monitors in terms of hexadecimal machine code. Readers will need a hex print-out of the three p.r.o.m.s and the mnemonic to hex conversion tables published in the July 1979 issue of *Wireless World*.

Several readers have expressed incredulity at the thought of working directly in machine code rather than using assembly language mnemonics. However, the hex codes for 50 to 60 of the most regularly used operations can soon be learnt and, thanks to the logical distribution of codes to operations, many more follow from these. The once-in-a-megabyte ones such as IN D (C), ED 50 in hex, can be obtained from the conversion table. This does not rule out working in assembly language and using an assembler, or translating yourself, but in my experience the latter soon becomes tiresome and it is easier to write in hexadecimal.

When writing software it is useful to have a supply of the forms shown in Fig. 1. The instruction 18, a relative jump, should be pronounced one eight and not eighteen. Similarly, the second byte is one seven and definitely not seventeen. If you want to jump forward with a relative jump, simply make the jump byte the number of bytes (up to 7F) over which execution must move, in this case 17 — 1 row and 7 bytes, to reach the target byte FF. For a jump back to the same target from the second 18, calculate the jump forward code to the next byte immediately under the target, 02 in this case, and then jump up row by row, decrementing the higher order hex character, i.e. from 02, F2, E2. When using a jump back the byte must be in the range 80 to FD (FE and FF serve no useful purpose).

Fig. 1. Typical software form.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
					18	17									
														FF	
										18	E2				

Work . . . . . No . . . . . .

Machine code monitor

Both monitors follow the same basic sequence as illustrated in Fig. 2. With the machine code monitor the base address of the Z80 stack is set, the address for the top corner of the screen is loaded into the DE register pair which is then used throughout the monitor as the destination pointer or vector for v.d.u. operations, and the message READY is printed by the subroutine at 03CE. This is one of several routines in the computer which draws data from the locations directly following the call of the routine. The program counter, which will have been pushed onto the stack, is exchanged with the contents of the HL register pair and then used as a pointer to that data before being exchanged back onto the stack, at the end of the routine, to cause a return to, in this case, 0010. The start procedure then clears the rest of the top line, resets the teleprinter output flip-flop and, using the subroutine at 0355, reads in and encodes a command from the keyboard. As explained in table 1, only the first and last letters are important to the subroutine. Whilst this limits the number of possible combinations which will produce different codes, a byte by byte comparison with a look-up table comprising all of the commands would use far too much p.r.o.m. space. After this has been achieved (001A), a comparison is made and if the code is not FC (the entry code for RUN) executions jump over 0D bytes for a further comparison and so on until a match is found, whereupon a block of instructions is executed before operation reverts to 0000 again.

Table 1. Low level monitor subroutines.

Address	
0254	Sets tape interface tone to 2400Hz and then calls 255 long time delays — about 4 seconds.
0260	Transmits the byte in register A to the tape interface, preceded by a start bit and followed by two stop bits.
027F	Calls a new line and then prints the contents of HL on the teleprinter.
028E	Formats the hex byte in register A for printing as two characters on the teleprinter.
02EC	Prints a space on the teleprinter.
02F0	Calls a new line on the teleprinter.
0301	Prints the contents of the A register on the teleprinter.
0317	List subroutine. Entered at 0317, the starting address must be loaded in from the keyboard. Entry at 031D assumes the address to be at 1FF0 to 1. Entry at 0320 assumes that the address is already in HL.
0336	A programmable time delay. The computer loops through six E3s, a long exchange instruction which, if used in pairs, does nothing but use up time. The number of loops is set by the byte immediately following the CALL in the original program. Each loop lasts 64μs.
0345	Clears the top line and sets DE at 8000.
034E	Used to format results, as in FIND and COR, this rounds DE up to the next multiple of 8.
0355	The algorithm for encoding input commands. Returns with last letter of the command minus the first letter in register A.
0372	The formatter used in LOAD and LIST in machine code language.
0393	Clears the v.d.u., leaving DE unaffected.
039F	Displays HL and a space. Used in LIST, LOAD, FIND, COR and in BURP lists.
03AA	Displays the contents of A as a two character hex byte.
03C4	Calls a new line on the v.d.u. and clears the remainder of the current one.
03CE	Displays the string of characters following the call in the program block up to byte 1D.
03DE	Loads HL from the keyboard.
03E7	Loads A with a hex byte from the keyboard.
03F6	Reads in a single keyboard character and, if a letter adds 6, then truncates to four bit binary (used as part of 03E7).

An exception to this is for the code FC, the routine for which 001E jumps immediately to 0042. This avoids one of the subroutines which have to be located at particular points in the memory map. Several subroutines can be called by single byte instructions which are known in mnemonic form as RSTs. These were originally intended for use with the 8080 and the Z80's "8080 mode" interrupt response which, after receiving the interrupt, calls for the interrupting device to place one or more instruction bytes onto the data bus for execution. Although this mode is not used, the single byte calls are a useful space-saver where a subroutine may be short and often needed. The subroutine which is avoided in this case at 0020 is called by byte E7 and produces a space on the v.d.u. At address 0028 is a jump to a subroutine which would require more than eight bytes. It is intended for use during the testing of machine code programs and when its RST byte EF is inserted into the program by using an ALT, it will suspend the execution of the program and display the contents of the HL, DE and AF registers, the point at which the EF was found, and the last entry onto the stack. Note that whilst there is not a specific subroutine at 0000 the one byte call CF for this address is used as an end command to a program. Although it does not do the same as C3 0000, because the stack is immediately re-defined at 0000, it has the same effect and saves two bytes.

Table 2. Machine code routine starting addresses.

002F	FILL	0042	RUN	004D	MOV
0099	LOAD	00F7	PROM	0120	ALT
0113E	ALPH	014D	PRINT	016F	COR
01CD	FIND	0203	TAPE	0226	READ
01A6	LIST				

The two interrupts also use fixed service routines. At 0038 is the maskable interrupt routine which reads in and stores number cruncher data using HL as a pointer. At 0066 the non-maskable interrupt's routine services the keyboard, first checking if the computer is at a HALT byte (76) and reading in the keyboard if it is or reseting the computer if it is not (006B is an example of a long relative jump). This particular software does not make use of the control characters available in ASCII except for the RETURN byte 0D, which it translates to 1F. Instead, it blanks off the top three bits of any codes above 3F (mainly the letters) at 007C and moves lower and upper case codes into the area 00 to 1F. This compression of the ASCII code into six bits produces bytes which are compatible with the v.d.u. character generator and this makes writing to the v.d.u., which occurs at many places in the monitors, a simple operation.

Beyond the service routines, the routines for the various operations in table 2 fit end to end up to 0253, with the exception of some unprogrammed space at 0130-9. This space may be used by overprogramming the jump byte 011F-10 and the ten bytes as required. Note that the LIST (01A4) routine is just a call to a subroutine at 0317 because an identical block of instructions are required as part of the ALT routine. As this is the last command code to be checked, the call is conditional on a match so that if the code is undefined, execution passes to 01A9 and a software reset.

Table 3. BURP subroutines.

0400	Used in graph plotting. Converts a number stored in 1E00-F to the nearest integral value. Negative values are put to zero.
042E	Executes MM57109 instruction present in register A. The sequence checks that the 57109 is ready, outputs the instruction with the hold off, waits for the ready to go off and then puts the hold on again.
0446	Repeats 042E for the string of 57109 instructions following the call in the main program. The list is terminated by FF.
0452	Jumps over the next word in the program line. Used in FOR statements to miss STEP and UNTIL.
045A	04E6 with BC protected.
0460	Outputs the contents of the 57109 X register to locations 1FF4 - 1FFF and then reformats it into the location specified by the contents of register A at the call, i.e. 1E10 for 01 in A, 1E20 for 02 etc.
04BA	Converts denary digits in the text to binary in register C. HL must be pointing to the first digit which must be in register A.
04D4	Graph plotting routine which scales the variables to be plotted to the screen matrix of 63 × 126. It divides the variable specified by the contents of A, by the declared maximum for that axis, and then multiplies by 126 before outputting to 1E00.
04E6	Jumps any spaces in the text and then analyses the following for (a) operators (04FB) which are converted by algorithm to 57109 code and executed (b) numbers (050F) which are rearranged and then input to the 57109 (c) instructions (054E) which are encoded by algorithm and the result used as part of the address for the location in a look-up table (positioned at the end of the r.o.m.) where the 57109 instruction code can be found, drawn and executed (d) variables (057B) which are found as in 0460 and entered as a series of 57109 instructions. When encoding words the standard algorithm two times first plus last is used but to compress the range of codes produced, those under 20 have 20 added and those above 50 are reduced by 10. This compressed byte is then added to 07B4 which is used as a pointer to the instruction required. Some instructions need two bytes for their execution, the first being 20, e.g. 24 is SIN but 20 then 24 is SIN^{-1}. These are encoded in the table and detected at 0566 by bit 7 of the instruction being set.
058A	Handles the 57109 $\overline{BR}$ (branch) output which pulses low whenever one of the 57109 test instructions proves to be true. The subroutine starts the execution of the instruction in register A and then reads in the 6-bit data word from the 57109. The four digit out lines are blanked off so that only the READY and BR lines, both initially high, get through (0591). By continually re-reading and jumping back on even parity, the Z80 is effectively waiting for one of these two lines to become active. If it is the BR line the Z80 outputs a NOP to the 57109 because, when tests prove true, the 57109 immediately looks for a new instruction and waits for its completion. If READY became active to indicate a failed test, the last procedure is jumped. Finally, the read in and masked byte which caused the exit from the parity checking loop, stored in register B as part of that loop, is read into A and masked for bit 6 so that the state of this line and hence the zero flag in the F register is set on a successful test.
05A9	With the HL register pointing to a variable in the text, and with that variable in register A, this subroutine computes the variable's address, formats it into 6 bit ASCII in the area 1 E00-F and converts results in the range 0.000 1-99 999 999 to floating point. The byte in the text is checked and, if a digit, is used as the new mantissa digit count to be stored at 1FE0 (063A). Whether or not the contents of 1FE0 are then drawn out, the block from 0641 to 0681 rounds off the figures after the decimal point to the extent indicated by this digit. Blanked figures are replaced by ASCII spaces. The mantissa is then sent to the v.d.u. and the text interrogated again, this time for a comma, which has the effect of suppressing the printing of any spaces and close packs the digits in the number (0693-7). Finally, at 06A3 an E for the exponent is looked for and if found the exponent is displayed. The alternative is three spaces or nothing, depending upon the comma, for floating point numbers.
06BB	Prepares the store area specified by the contents of A using 0714 and then reads in a number from the keyboard, converting standard and non-standard scientific and floating point arrangements to the machines standard format.
0714	Prepares a number store by dumping 9 00s, 60Fs and a 3F. This means that 06BB dump the input data into the store without having to worry about leading or trailing zeroes or the non-existance of an exponent (0Fs being NOPs as well). The 3F terminates number entry to the 57109 as well as being a NOP therefore two consecutive variable inputs to the 57109 do not have to be separated by an ENTER as with reverse polish calculators.
0729	Algorithm for entering words from keyboard (two times first letter plus last).
0736	Inputs denary keyboard digits to binary in C.
074A	Converts A to three digit denary and displays on v.d.u.
076D	Converts A to three digit denary and displays on teleprinter.
07A2	Data for MOD command.
07AC	Forms the address for the start of a variable store area in HL from the variable code in A.
07B7	Displays a number formatted by 05A9 in 1E00-F displaying E for 0B, . for 0A, a space for 0F, - for 0C and ASCII digits for 00 to 09.
07D6	The look-up table for the 57109 instructions.

Table 4. Format for storing and printing three variables.

```
005 LET A=23.45
006 LET B=-0.00733
007 LET C=3.456E33
008 PRINT A
009 END
0C45

1E00  A0 32 33 2E 34 35 30 30 20 20 A0 A0 A0 A0 A0 6F
1E10  02 0A 03 04 05 00 00 00 00 0F 0B 0F 00 01 0F 3F
1E20  07 0A 03 03 00 00 00 00 00 0C 0B 0C 00 03 0F 3F
1E30  03 0A 04 05 06 00 00 00 00 0F 0B 0F 03 03 0F 3F
```

Table 5. BURP routine starting addresses.

083F	DEL	08C7	MOD	08F7	ADD
0929	LOAD	092F	LIST	0939	DUMP
0977	RUN				
099A	INPUT	09B2	PRINT	09D8	END
09DF	GOTO	09EB	LET	0A04	IF
0A43	WRITE	0A83	ERASE	0A8F	TOP
0A9E	GOSUB	0AAA	RETURN	0AB2	FOR
0AE2	NEXT	0B13	HALT		

From 0254 to 03FF are the subroutines listed in table 1. When necessary, the subroutines PUSH registers to be used solely within the subroutine and then POP them back before the return so that no interference is caused to data within the main program. Most subroutines are self-contained but some, e.g. 02EC, jump on to others for their completion. As subroutines are sometimes called within subroutines, within subroutines etc., the stack storage area, which extends into the r/w.m. from 1FDF, should be left free to at least 1FC0 for the computer's use. Like C7, other space savers will be found in the subroutines, e.g. AF to clear the register A instead of 3E 00, A7 to set the flags according to the contents of A. To save byte space some apparently unnecessary bytes appear, e.g. E3 at the start of the routine at 03C4 is included so that it and 03CE can share the same ending. Care is needed when writing subroutines because with a lot of PUSHing, POPping and EXchanging taking place it is important to ensure that the bytes called back off the stack by the return command at the end of a subroutine are definitely the correct ones. I have found this to be the Z80's most adept way of erasing painstakingly developed programs. This type of error is common when a conditional return is used as in 034E which prints spaces until the lower three bits of DE are zero. Ideally this should have pushed AF initially as it uses A and F, but to also arrange for them to be restored on return would extend the routine to at least nine bytes. The EF described earlier is a very powerful tool for sorting out these problems.

BURP monitor

For the BURP monitor the first p.r.o.m. is solely for subroutines whilst the second contains the operating system which makes use of them. Details of the subroutines are given in table 3. In BURP, program material is loaded from 0C00 on, the area 1E00 to 1E0F is used for the formatting of results to be printed and 1E10-F stores variable A and so on up to 1FB0-F which holds the FOR loop step. Table 4 shows the formatting used for the storage and printing of three different variables. Note that all results are stored scientifically to maintain eight digit accuracy. Although the MM57109 can operate in either mode, it is quicker to stay in the scientific mode and let the Z80 convert the results within the range 0.0001-99999999 to floating point for display.

At 0800 the stack pointer is set and DE is assigned as the screen pointer again. BURP is then displayed and the rest of the top line cleared. The mantissa digit count is set at 04 (0817) and the screen position reset to 8008 ready for the input of a command. 081E to 0823 is harmless nonsense and 0824 to 0837 resets the number cruncher by sending the operation 3F (NO OP) with the hold to the 57109 off, pausing for 8ms and then reapplying the hold. During this sequence the interrupt mode is set but as it is the masked one that is driven by the number cruncher, the somewhat capricious behaviour of the i.c. before it has been reset has no effect upon the rest of the system. The i.c. is then given a master clear (2F) and switched to the scientific mode (22) by a multiple executive subroutine at 0446 (0832).

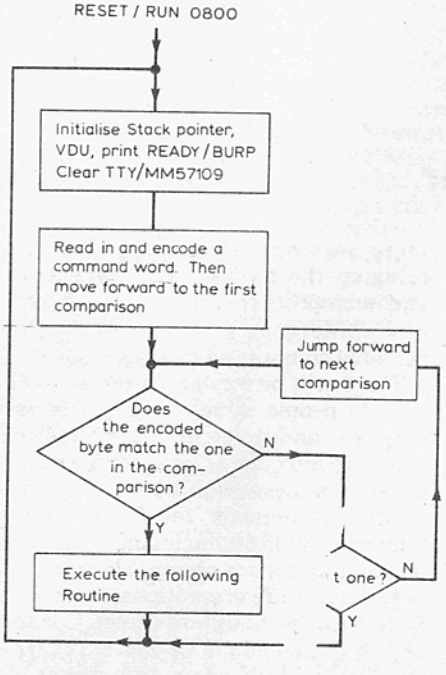

Fig. 2. Basic operating sequence for both monitors.

At 0838 another command encoder is called to read in a command from the keyboard. The algorithm used here is two times first plus last, so once again only two letters are required. However, this algorithm is capable of producing a far greater list of codes and therefore reduces the chance of two words deriving an identical one. As with the low level monitor, routines entered by recognition of this code ensue, see table 5. The start of the last of these, for the RUN command, reads in and encodes the line number input in the command and stores it in register C. The v.d.u. pointer is then set to 8040, the start of the second line, and C is decremented, pushed, popped, incremented and then pushed again. Four of these operations might seem to do nothing to C and on this occasion they do not. The total effect is to store the current line number on the stack. When the execution of a line is completed however, the next line number can be computed and saved by returning to 097F. After GOTOs, when A will hold the next line number, a pop to remove the old number followed by a jump to 0981 will load this as the next line to be executed. As all lines will terminate by jumping back to one of these locations (except for END which returns to 0800), to avoid absolute jumps (i.e. jumps to specific addresses), relative jumps to these two critical points are string out through the third p.r.o.m.

A line of BURP is stored as the hex byte ED, the line number in hex, the actual data in modified ASCII and then the byte 1F to signify its end. The end of the memory block in use is signalled by the byte C0. With the commands ADD, DEL, DUMP, LIST and RUN involving line numbers, the interpreter scans the program block up to C0 and looks for ED followed by the line number in question. During a RUN the next word in the line is encoded using the two times first plus last algorithm (0993) and again, the routines for all of the commands are strung end to end and each is preceded by an immediate compare and a jump-on-not-zero (20 hex). The last command, HALT, compares at 0B0F and if a match is not made the computer jumps over the single byte 76 of the HALT routine and goes on to the next line by executing several relative jumps back to 097F. This explains why there is no routine for REM as it and any unrecognised first word on a line is just

Table 6. New features of the improved firmware.

General
V.d.u. cursor on all modes.
DEL delete last character on all modes.
RETURN available in graphics mode.
Interface for ASR or KSR teleprinter (as printer and/or punch).

BURP
Extended IF statements. Any statement may be conditioned by IF.
Mathematical capability available in IF, FOR PRINT, WRITE, GRAPH and AXIS statements.
Printed strings in INPUT as well as PRINT statements.
Multiple statements — virtually unlimited numbers of statements may be written against a single line number. This speeds execution and expands the effective statement capacity well beyond the 254 lines.

Extra maths functions:

ABS	makes current result positive
INT	blanks digits following decimal point
FRAC	blanks digits preceding decimal point
RND	places pseudo-random number into the MM57109

No need for LET at the start of LET-type lines.
'in a line, causes the computer to ignore the data following, up to the end of that line (alternative to REM).

Hardware changes required
The wiring of several spare keys.
The teleprinter interface shifted from D_7 to D_0, and 55V reduced to 5V.

P.r.o.m. required
Complete with the graph plotting firmware, this will still fit into three 2708 e.p.r.o.ms.

ignored (the very requirement for REM). Throughout the monitor the register pair HL address the program block contained from 0C00 onwards, whilst B and C are available for general use within the execution routines.

Subroutine p.r.o.m.

As far as possible, subroutines have been written which can be called in many different places within the interpreter. This particularly applies to 04E6 which can be thought of as a basic text handler which recognises and deals with words, variables, numbers and operators.

In the next part a new monitor will be described, the features of which are given in table 6. I would like to thank all of the readers who have contacted me with suggestions for extra facilities and I hope that the new system will meet their requirements. Lists of the new firmware will be available from *Wireless World* (editorial department) upon receipt of a large s.a.e. and these will be a useful accompaniment to the details in part two.

The author is offering a set of three p.r.o.ms programmed with the new monitor hardware for £30. Alternatively, existing p.r.o.ms can be reprogrammed for £6.50 (both plus 35p post and packing).

Micro show is bigger

Personal computers are prominent among the systems to be displayed and discussed at the Microsystems '80 conference and exhibition, January 30 to February 1. Sponsored by *Wireless World* and associated electronics and computer journals, this annual event has grown in size to such an extent that it has had to be moved from its hotel venue to the Wembley Conference Centre (opening hours, 0930 to 1800 hours each day).

The 1980 conference has a four-part programme ranging from an introduction to microprocessors to an overview of the latest developments in microelectronics. Topics include: technology update, micro processor software, controlling microprocessor projects, microprocessor applications, bridging the hardware/software gap, and microprocessors in process control. The conference will concentrate on personal computers on its third day.

There will be buyers' forum sessions to help people in selecting goods and services, and a one-day appreciation course to introduce managers to the use of microprocessors in business and industry. Delegates' fee for the conference is £145.50, including v.a.t. and booking forms are obtainable from the organizers, Iliffe Promotions, Room 821, Dorset House, Stamford Street, London SE1 9LU (telephone 01-261 8113). The exhibition, with some 110 stands, is open to all at no charge, whether or not the visitor is a conference delegate.

Literature Received

Reference sheets on the world's **electronics industry** produced by Mullard, showing exports, consumption, production of a variety of products. Sheets can be obtained from Mullard, Ltd, Mullard House, Torrington Place, London WC1E 7HD. WW401

Leaflet on the ZIP KDP **computer terminal,** comprising 30ch/s dot-matrix printer, keyboard and v.d.u., can be had from Data Dynamics, Data House, Springfield Road, Hayes, Middx. WW402

Fourteenth edition of **Intel News** contains descriptions of an 8086 single-board computer, 1Mbit bubble memory and other items of interest in the computing field. Intel Corp (UK) Ltd, 4 Between Towns Road, Cowley, Oxford OX4 3NB. WW403

Solid-state relay applications manual on specification, preotection circuits, loading and failure modes, with typical circuits, is available from Hamlin Electronics Europe Ltd, Diss, Norfolk IP22 3AY. WW404

Full ordering information on the component parts of the Elma **collet knob** range is available in broadsheet or wall-chart form from Radiatron Components Ltd, 76 Crown Road, Twickenham, Middx. WW405

Signal-conditioning amplifiers in the SE 990 series are described in a leaflet now available from Spur Road, Feltham, Middx, TW14 0TD. WW406

Data for meteorologists, oceanographers, and ecologists can be collected by sensors on ships, without attention from the crew, collated by a **data collection platform** and transmitted to a satellite for retrieval. The McMichael platform is briefly described in a new leaflet from McMichael Ltd, Wexham Road, Slough, Berks SL2 5EL. WW407

Leaflet from Astralux gives full details of 8000 series of opto-coupled, **solid-state relays** in 10, 20, 30 and 40A versions. Sales department, Astralux Dynamics Ltd, Brightlingsea, Colchester, Essex CO7 0SW. WW408

Selection of test equipment for **logic-testing** is presented by Electroplan in a four-page leaflet, obtainable from Electroplan Ltd, PO Box 19, Orchard Road, Royston, Herts. SG8 5HH. WW409

Various types of **panel meter,** counters, printers, etc., are described in a 48-page catalogue, produced by Techmation, Ltd, 58 Edgware Way, Edgware, Middx. HA8 8JP. WW410

Brochures on the American Crown (Amcron) range of **audio equipment** can be had from the sole UK distributors, HHB PA Hire and Sales, Unit F, New Crescent Works, Nicoll Road, London NW10 9AX. WW411

A collection of tools for **bending and cutting** component leads and for handling i.c. packages is detailed in a Wybar catalogue from Eraser International Ltd, Unit M, Portway Industrial Estate, Andover SP10 3LU. WW412