More on the scientific computer — 2

An improved monitor

By J. H. Adams, M.Sc.

Since publication of the scientific computer, correspondents have suggested several features to improve the performance. This new monitor incorporates many of those features and includes a general expansion of the facilities available in BURP, including the routines for graph plotting. By restructuring the interpreter four extra functions, described in table 7, have been fitted into the three original e.p.r.o.ms. The demonstration programs have been removed, but these could be stored on tape, and the Creed 75 teleprinter interface has been replaced by a standard 110 baud ASR/KSR interface. The KSR machine is now cheaper and is fairly standard whereas the 75 may have different speeds and encoding as I suspect some readers have found to their cost.

Hardware modifications

Connections for the two extra keys are shown in Fig 3. The interface for the teleprinter is essentially a latch as in the original design, but this must be connected to D_0 instead of D_7. Most teleprinters contain an interface card for a 20mA loop or an RS-232 link. For a current loop, the second circuit drives the printer quite satisfactorily.

Firmware modifications

Changes to the firmware are detailed in tables 8 and 9. Primarily, space has been made in the first e.p.r.o.m. for three of the subroutines originally in the second which deal with instruction entry and condition testing of the MM57109. This has been achieved by using a simpler and shorter teleprinter interface, eliminating the subroutine at 034E, and trimming the low level monitor so that it ends at 024E. This has left space in the second e.p.r.o.m. for a new subroutine 051D which extends the old 04E6, now 047C, and together they can recognise and deal with the new facilities. Because these routines are quite complex, a disassembled listing of each is given in table 10.

The third r.o.m. is slightly briefer because checks for ends of lines, present in virtually all of the statement handling routines, are replaced by 051D. The command MOD (08BE) has been changed so that PRINTs buried in multi-statement lines are also changed to WRITEs. CALLs have been readdressed to suit the first two r.o.ms and CALL 042E has been replaced by the single RST byte CF (see 0008). In the original r.o.m., after going through the sequence of recognition checks for encoded commands or, later, first words of statements, the interpreter returns to the command state or ignores the rest of the line respectively, if it cannot find a match or the generated code within the firmware.

This is particularly useful for dealing with REM because, being unrecognised, such lines are ignored as explained last month. A major change in the modified r.o.m. provides jumps to 1C00 (at 0975) for commands, to 1C60 (at 0AD7) for new statements and to ID00 (at 0BDE) for new functions. As a result REM has disappeared but the apostrophe has the same effect and retains the facility for remarks.

0993 is an example of where 051D is used solely to jump spaces between the line number and the first word of the statement. Therefore, it is the point to which 051D transfers execution after coming across an ! in the text being interpreted. 097F pops off the stack, increments and pushes back the C register which is used as the line register store and then looks for and executes that new line. Thus, it is the point to which 051D transfers control after finding a ' or 8DH number in the text. Because the computer scans the text for line numbers whether they exist or not, the lines in a program should be as close together as possible (say every other line) for the fastest program execution. Using multiple statements avoids this problem to some extent and can therefore reduce the execution time of some programs, particularly simple ones, by up to 20%.

Table 7. Additional facilities for the new monitor.

INT (0B64)	Outputs the number in the 57109 to 1EOO — F and tests the exponent sign. If negative, the whole number is written to zero, if positive, the lower mantissa exponent is drawn and used to calculate (OB72-8) where blanking should start. If the exponent is not less the 09 (OB80-B), blanking is carried out. The number stack in the 57109 is then collapsed by one to remove the old value (OB97) and the new value is entered into the 57109 by a jump to 050F at OB9A.
FRAC (0BA1)	Outputs the number and tests as in INT. If the exponent sign is negative, execution jumps to OB96 (OBA5) and effectively does nothing. For positive exponents a similar sum involving the lower mantissa exponent digit is performed and a jump is made back to OB79 in the INT routine (OBAE).
RND (0BB4)	029F is called which loads the refresh register into A, converts it to a three digit decimal integer and enters it into the 57109 (this subroutine runs straight into 02AD). A pseudo-random delay (OBB8-A) based on the current v.d.u. printing position is then called so that a second call of 029F will generate a second number from the Z80 refresh register which is only tenuously linked to the first. These numbers, now in the Y and X registers of the 57109, are combined through the sequence of instructions at OBBE to give X = 128X + Y/16383, i.e. a reasonably random number between 0 and 1. Note that as this uses two of the 57109 stack registers, no more than two other variables must be present in the 57109 when RND is used.
ABS (0BD3)	This simply uses the number cruncher test instruction 12 to test for a negative number in the X register. The result of this test governs whether the instruction to change sign, OC, is executed.

Table 8. Alterations to the first r.o.m.

024F was 03CE	0263 was 0260	0282 was 058A
02AD was 024E	02C7 was 0446	0326 was 0317
0345 was 0336	0367 was 0729	0374 was 0372
0395 was 0393	03A1 was 039F	03AB was 03A9
03C6 was 03C4	03D1 was 0260	

029F Generates a 7-bit pseudo-random number and inputs it to the 57109.
02D1 Converts the computer 6-bit ASCII to true ASCII and prints it.
02D9 Prints a space.
02DE Prints carriage return and line feed.
02E8 Prints the contents of register A.
02F0 Prints (A) as a two character hexadecimal byte.
0317 Prints CR, LF, the contents of HL in hexadecimal and a space

Using the new facilities

In low level the first feature to be noted is that READY does not disappear when a command is typed in nor does the first letter appear at the beginning of the second v.d.u. line. This is because the same algorithm is now used for both high and low level word recognition. Clashes produced in the changeover explain the changes of COR to MOD and PROM to PROG. To leave LOAD, the space key is now used instead of @. The main change which affects both levels is that the interrupt-and-reset, which occurred whenever any key was depressed, has been omitted because control can be regained by using RESET. The "arrow" keys now revert to standard keys, RESET enters the low level and Control A (depressing A and the control key simultaneously) enters the high level. The delete key to the right of] can be used to delete complete bytes by one depression per byte. Although this will cause the formatting to go out of true during the LOAD, the grouping by four is maintained and on pressing the space bar at the end of the load the format will be restored.

When loading programs in high level language, another character Control E is used to signify the end of LOADing or ADDing. This allows the colon, which was previously used for this purpose, to be included in printed messages etc. without terminating the current operation. Ensuring correct format of the input has been eased by a cursor, although with the original monitors few problems will be encountered if a space is typed when in doubt. The DEL key backsteps and clears the last v.d.u. character and also backsteps HL. Corrections are, therefore, easily typed in, but mistaken returns and line numbers cannot be corrected in this way because

Fig. 3. Modifications to the keyboard and teleprinter interface.

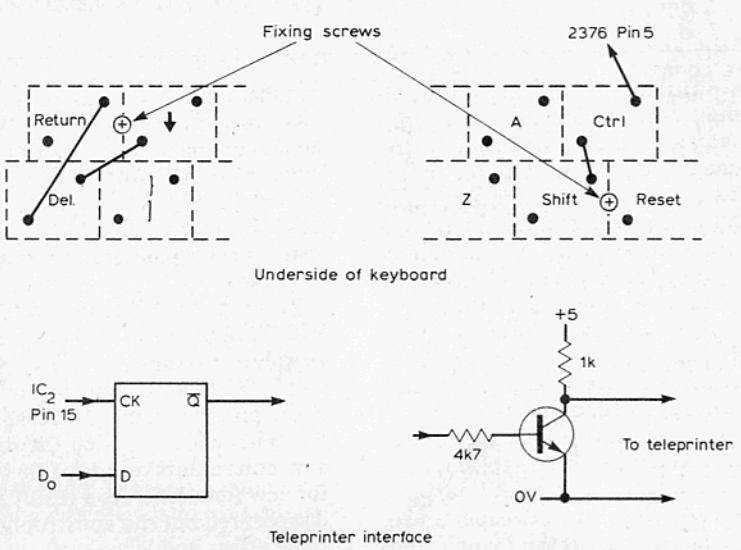

Table 9. Firmware changes.

0400	Old 04D4 running straight into 040D
040D	Old 0460
0467	Old 04BA
047C	Old 04E6, 04FA-E is added to this so that when a code of less than 0B is drawn from the look-up table at the end of the r.o.m., execution jumps to 0B60. These new codes are for ABS, FRAC, INT, RND and any others which are not simple MM57109 operations and will thus require some Z80 software.
051D	Jumps spaces and then returns on bytes less than 1B and greater or equal to 2A (except for 8D). Thus, for letters, operators and spaces, this routine will just jump spaces and return with HL pointing to the first non space, i.e. 051D is a supplement to 047C. If the byte found lies between 1A and 2A it will, after; (a) ''(052D) transfer text up to the next'' onto the v.d.u. and then jump back to the start of the subroutine to deal with whatever follows. (b)) (053B) collapse the stack and return. (c) ((0542) call 051D to jump spaces and then 047C to execute the text within the parentheses until the call of 051D finds a). As this) will have been found during the calling of 051D at 0546 and as) indicates that the original call of 051D is no longer required, i.e. the bracketed term has been computed, detection of) drops the stack pointer past the return address the call at 0546 so that a return is made to the original point in the interpreter from where 051D was called. After dealing with an expression in parentheses, the computed result is left in the X register of the 57109 and the SCII for), 29, is left in register A. If the interpreter has not yet recognised the byte it must now be at the end of the statement. Before looking for a ! ' or 8DH, two types of statement need special attention. 1FE1 is used in the third r.o.m. (0999) to store the code generated from the first word of the line. If it is 33 (i.e. a WRITE statement), execution shifts from 0554 to 056B. WRITE lines are similar to print types except that the material to be displayed is fed to locations from 1D80 rather than to the v.d.u. 056B sets an FF at the end of the block used and then resets DE to 1D80 and outputs the characters up to FF on the teleprinter. After restoring AF and DE it returns to 0563. If the line is a LET (code 2C) the variable to which the computed value is to be assigned is drawn from its store (1FE2) and the contents of the 57109 X register are fed to it. After dealing with these two special cases, checking of the original byte continues (0560). The remaining possibilities will transfer control rather than return from the subroutine and so the pointer is moved down the stack, losing the previously stored return address and then, after; (d) ! (0563) execution passes to 0993. (e) 8DH ' or anything else, passes execution to 097F. 8D is the code for return and indicates the end of a line. ' signifies that the rest of the line is a remark which the interpreter will also want to treat as the end of a line.
0582-4	Jumps text and then calls 051D and, when required (i.e. letters, operators or digits), 047C as well.
0589	Calls 051D as above.
0594	Old 0714.
05A9	Unchanged.
0736	Unchanged.
074A	Modified 074A.
075A	Old 076D.
0773	Used in the above two to cover common parts and thus save space.
0789	Used in INT and FRAC.
07A2	Unchanged.
07AC	Unchanged.
07B7	Unchanged.
07D6	Look-up table which now includes codes for new functions (07DA/DC/E3/E9).

they involve internal operations by the interpreter rather than the byte by byte storage which takes place during lines. The critical formatting points are LET lines where the variable following let must be followed immediately by the equal sign, and IF lines where, when a variable precedes the comparison sign, there must be a space in between.

A program in table II demonstrates the uses of the new facilities. Lines 3 and 4 show the new REM and in this case they are complete lines on their own. Remarks may be appended to any "active" line just preceded by an apostrophe. Line 5 shows printed text in an INPUT line. The input variable X is against the " to save r/w.m. space but again, spacing is not critical. In line 7, two spaces are left between step and 1 without any effect on the interpreting of the line. Note that the expression in parenthesis is spaced exactly as in a LET statement. Line 9 demonstrates the compounding of two LET type statements (with the LET omitted) by the use of an exclamation mark. The statement following ! is typed immediately after the !, again to conserve r/w.m. space. Line 11 is "If K is a whole number and if Z is also a whole number, then print half of K plus A to two decimal figures and then half of the positive difference between K and A". This line illustrates the need for a space between the variable and the greater than, equals or less than sign. A space is required because, under the original interpreter, this had to be a variable but it can now be a variable, number or function in parenthesis and therefore has to be distinguishable. A closing parenthesis has no other meaning and does not need the space, i.e. IF (X SIN I —)=Q print....

The text following an IF comparison can be any other permitted statement including another IF as shown in the example program. Therefore, the old form IF X=0 THEN I25 will be IF X =0 GO 125. It might seem that the freedom to place statements end to end on the same line will reduce all programs to one line in length (note that a line is not determined by the length of a v.d.u. line and may consist of any number of characters). However, this is not so because whenever a statement has to be entered as the result of a jump, or it initiates a specific jump, the statement must either start or end a program line respectively. This means that the first instruction in a FOR loop must be at the beginning of a line because further through the execution a NEXT will try to jump back to it. Similarly, the statement after the complete IF term must be on a new line because IF is basically "perform the operation specified after the conditional test if the latter is true or jump to the next line".

By similar reasoning, GOSUB and GO should be at the end of lines, as should RETURN and END. The lines to which GOSUB and GO refer should start with the statement to which the jump was directed.

While encoding the new functions by algorithm, several clashes occurred with already assigned codes and this provided an opportunity to re-encode the two log. functions into a more standard format, i.e. CLG for a common log and LOG for log. to the base e. The radian to degree conversions have also been changed by dropping the first letter, i.e. TD for a conversion to degrees and TR for one to radians.

The author is offering a set of three p.r.o.ms programmed with the new monitor firmware for £30. Alternatively, existing p.r.o.ms can be reprogrammed for £6.50 (both plus 35p post and packing). 5 The Close, Radlett, Hertfordshire.

Table 10. Disassembled subroutines.

```
047C  LD A,(HL)
047D  INC HL
047E  LD C,0F
0480  CP  20
0482  JRZ  F8 047C
0484  CP  1B
0486  JRC  51 04D9
0488  CP  3C
048A  JRNC 17 04A3
048C  CP  2D
048E  JRNZ 03 0493
0490  CP  (HL)
0491  JRC  0C 049F
0493  ADD 0D
0495  AND A
0496  JP PE049D
0499  OR  09
049B  AND FB
049D  RST 1
049E  RET
049F  LD C,0C
04A1  LD A,(HL)
04A2  INC HL
04A3  EX AF AF'
04A4  PUSH DE
04A5  EX DE,HL
04A6  LD HL,1E00
04A9  LD B,0F
04AB  CALL 05AC
04AE  LD L,09
04B0  LD (HL),C
04B1  LD L,00
04B3  EX AF AF'
04B4  AND 0F
04B6  CP  0E
04B8  JRNZ 02 04BC
04BA  LD A,0A
04BC  LD (HL),A
04BD  LD A,(DE)
04BE  INC HL
04BF  INC DE
04C0  CP  28
04C2  JRNC F0 04B4
04C4  CP  20
04C6  JRZ  44 050C
04C8  LD L,0A
04CA  LD (HL),0B
04CC  INC HL
04CD  LD A,(DE)
04CE  INC DE
04CF  CP  2D
04D1  JRNZ E1 04B4
04D3  LD (HL),0C
04D5  INC HL
04D6  LD A,(HL)
04D7  JR  DB 04B4
04D9  EX AF AF'
04DA  LD A,20
04DC  CP  (HL)
04DD  JRNZ 03 04E2
04DF  EX AF AF'
04E0  JR  2E 0510
04E2  DEC HL
04E3  CALL 0715
04E6  CP  20
04E8  JRNC 02 04EC
04EA  ADD 20
04EC  CP  50
04EE  JRC  02 04F2
04F0  SUB 10
04F2  ADD B4
04F4  PUSH BC
04F5  LD C,A
04F6  LD B,07
04F8  LD A,(BC)
04F9  POP BC
04FA  CP  0B
04FC  JP C 0B60
04FF  CP  80
0501  JRC  05 0503
0503  EX AF AF'
0504  LD A,20
0506  RST 1
0507  EX AF AF'
0508  AND 3F
050A  RST 1
050B  RET
050C  DEC DE
050D  EX DE,HL
050E  POP DE
050F  XOR A
0510  PUSH HL
0511  CALL 07AC
0514  LD B,10
0516  LD A,(HL)
0517  INC HL
0518  RST 1
0519  DJNZ  FB 0516
051B  POP HL
051C  RET
051D  INC HL
051E  LD A,(HL)
051F  CP  20
0521  JRZ  FA 051D
0523  CP  1B
0525  RET C
0526  CP  8D
0528  JRZ  03 052D
052A  CP  2A
052C  RET NC
052D  CP  22
052F  JRNZ 0A 053B
0531  INC HL
0532  LD A,(HL)
0533  CP  22
0535  JRZ  E6 051D
0537  LD (DE),A
0538  INC DE
0539  JR  F6 0531
053B  CP  29
053D  JRNZ 03 0542
053F  INC SP
0540  INC SP
0541  RET
0542  CP  28
0544  JRNZ 08 054E
0546  CALL 051D
0549  CALL 047C
054C  JR  F8 0546
054E  PUSH AF
054F  LD A,(1FE1)
0552  CP  33
0554  JRZ  15 056B
0556  CP  2C
0558  JRNZ 06 0560
055A  LD A,(1FE2)
055D  CALL 040D
0560  POP AF
0561  INC SP
0562  INC SP
0563  CP  21
0565  JP Z 0993
0568  JP 097F
056B  LD A,FF
056D  LD (DE),A
056E  LD E,80
0570  LD A,(DE)
0571  CP  FF
0573  JRZ  03 057D
0575  AND 3F
0577  CALL 02D1
057A  INC DE
057B  JR  F3 0570
057D  POP AF
057E  POP DE
057F  POP DE
0580  JR  E1 0563
```

Table 11. Demonstration programs.

```
003 'THIS PROGRAM, PUBLISHED IN PART 4, TOOK 19 LINES BEFORE. NOW...
005 PRINT "THIS PROGRAM USES NEWTONS METHOD FOR SOLVING"
007 INPUT "F = F(X). ENTER AN INITIAL VALUE NOW "Q !ERASE
009 X=Q !GOSUB 25
011 G=F !X=X 1.00001 * !GOSUB 25
013 TOP !IF (G ABS )<0.000001 PRINT "SOLUTION ="Q6 !END
015 Q=1 F G / 1 - REC 0.00001 * - Q * !PRINT Q8 !GO 9
025 F=X LOG X 3 * + 10.3074 -
027 RETURN

0D78

003 'THIS PROGRAM COMPUTES PAIRS OF NUMBERS WHICH, WHEN
004 'SQUARED AND SUBTRACTED, GIVE THE INPUT NUMBER
005 INPUT "INPUT NUMBER IN QUESTION "X
007 FOR A=1 STEP  1 UNTIL (X ROOT 1 + )
009 X=X A / !Z=X A - 2 / ABS
011 IF X =(X INT ) IF Z =(Z INT ) PRINT (X A + 2 / )2 (X A - 2 / ABS )
013 NEXT A !GO 5

0D26
```