(41)

TRS80 SYSTEM 80 COMPUTER GROUP

CLUB NEWSLETTER

REGISTERED BY AUSTRALIA POST
PUBLICATION NO. QBH 3667

ISSUE: MARCH 1986

COMMITTEE 1985-6:

PRESIDENT / CHAIRMAN : J P GALVIN
TREASURER : L J LAWES
SECRETARY : W J ALLEN
NEWSLETTER EDITOR : A P WEST
LIBRARIAN : D CLARKE
PROGRAMS CO-ORDINATOR: P GOED

ADDRESS ALL MAIL TO:-
HON. SECRETARY,
16 LAVER ST.,
MACGREGOR,
QUEENSLAND 4109

DEAR MEMBER, our last meeting, held at Lindum Progress Hall on March 2nd, saw another very strong turnout -- the hall was again packed -- with 67 people, including 8 visitors. Four apologies were tendered. There were some new members present who were welcomed to the group.

Secretary's news: Mail received -- an invitation from the Hobart Users Group to exchange newsletters; The Federal Publishing Co. inviting original games programs to be published in a new games mag. @ $25 per game; library book returned by mail by conscientious member; Dick Smith sent several copies of their handout newsletter, which includes groups' contact addresses -- including ours; Northern Bytes v6n8 plus TAS software catalogue; Amug, Cog 'n' Spiel, Illawarra newsletters; Wilbroprint advising on their computer supplies lines; 2 Sydtrug newsletters, the latest describes installation of 1 megabyte RAM kit (c. $500) to Model 4P (suits Models I/III/4 also); an offer of cheaper Viatel modem/software packages (Model 3) -- Rank Electronics, Sydney; another letter from Victorian prisoner.

Business as usual reported by other Committee members.

GENERAL BUSINESS

A main topic was a proposal to have a workshop session at the hall on the morning of our regular meetings, where more personal tuition and demonstrations on the machines can be done in small groups. About 20 members indicated intention of attending the first session and it was decided to have the first of these sessions on April 6th. Since the meeting, it has been found that the hall is not available to us in the morning, so Lance Lawes has kindly offered us the facilities downstairs at his house (only a short distance from the hall). The address is 21 Rodney Street, Lindum, and you turn RIGHT just before the railway crossing (in Kianawah Rd) if coming via Wynnum and Kiawanah Rds., drive past the Station and shops, and Rodney is the second street on the right. Via Lytton and Lindum Rds., go past the hall and turn RIGHT, go across the crossing and immediately turn LEFT, pass the Station etc as above. For $2 a head, Lance will accommodate us and provide eats at midday and the session will be from 10 a.m. to noon approx. It would assist Lance greatly to cater if you give him a ring (396 2998) about a week before if you're coming to the workshop. The normal monthly meeting will take place as usual at the Hall at 2 p.m.

Further work on the Helpdisk is proceeding and the plans have been changed a bit to make it a much handier facility. Here is an outline of the planned features:- With great help from Glen MacDiarmid, we hope to soon have this facility accessible at ALMOST ALL times on the computer (running m/l with interrupts enabled or running Basic [excepting during garbos etc] or either Command modes in DOS or Basic)! A small projected change to the operating system (Newdos80) means that you can press 1-2-3 simultaneously and a block of memory and the present screen is saved to disk (to restore the situation prevailing before you pressed 1-2-3 on exit from the Help mode) and a menu of Helpfiles is presented on screen.

If Debug is wanted instead, you hold down <SHIFT> while pressing 1-2-3. If you have a minimum disk system (2x40 or 35-track single density single siders -- this facility is going to be TOO LARGE to be practicable on a SINGLE 40-tr ss sd system), you will have to mount the disk with the chosen Helpfile at this point, then the plan is to have the facility cursor driven. If you have a much larger system, such as double siders/double density/80-trackers etc, you will be able to have more than one Helpfile on a single disk. It seems likely that a full file will be between 60/80K. Each record will be in 256-byte modules, up to 3 modules -- i.e. 1, 2 or 3 sectors per record, so 4 lines on the screen will be available for instructions.

The time taken to get this project into use will depend on a LOT OF INPUT RIGHT NOW from members concerning the actual information stored. I'll take it on paper, written, printed or ASCII files or any suggestions will be carefully considered and I'll do the formatting.

I think it's an ideal project for combined effort from which we will learn much and it will thus knit us into an even stronger group. We will be able to upgrade it from time to time by issuing substitute Helpfiles. I believe we can give our machines a tremendous upgrade with this facility, which greatly increases their usefulness and I believe it will offer a great deal more than is currently available on much more expensive machines. Won't it be lovely to be able to have practically all the info you'll ever need at the touch of a few keys instead of scratching around in books etc? Why not let the computer do all the hard work? Isn't it why we bought them? The idea can be easily extended later to helpfiles for specific applications programs -- so you can access the program crib on the run without destroying the program.

MEMBERS' PROBLEMS: Sorry, but I've misplaced my notes (that's MY problem) -- nevertheless, we managed to organise cures for most problems tackled at the meeting.

MEETING TOPIC: At the close of the regular meeting, Alf West gave a lecture covering the main points of the series of articles on Newdos80 file handling. I think we are learning a lot from all Alf's efforts. We're much obliged to Alf for showing how Newdos80 runs rings around all other DOS's in this area. It's a subject they (the other DOS proponents) keep very mum about. Many thanks, Alf.

The meeting closed at 4.10 pm.

The NEXT MEETING (see General Business re Workshop) will be at the LINDUM PROGRESS HALL, Lindum Rd., Lindum, on Sunday, APRIL 6TH, starting at 2 P.M. (Come via Wynnum Rd. and turn into Kianawah Rd., go over rail crossing and turn left or via Lytton Rd. and turn into Lindum Rd. The hall is < 100 yds Lindum Railway Station). After the main meeting, Dave Clarke will give a talk on the way information is stored on tape and disk, to fill in for those who may have got lost in the Superzap talk. Hoping to see you there -- in the meantime,

BETTER COMPUTING, Bill Allen

==========

FOR SALE

System 80 Business Computer 48K, RS232c interface, Green Screen Monitor, 2 40t SS, SD disk drives, GP-80 printer, all power supplies, cables etc, some software w/ manuals and instructions. $800 the lot. Steve Brandt 359 1329 after hours.

1 Model I expansion unit (new) w/full memory fitted, $200. George Cornwell, 209 5238.

Bob Holmes wants to sell the following surplus items: 2 Tandy 40-tr ss drives, drives cable, printer cable, numerous computer books. Phone 349 7666.

System 80 Blue Label with hi-res graphics unit (with suitable programs) and expansion unit with double density board -- will separate. Contact Brian Boettcher 848 3877 (a.h.) or 222 5597 (b.h.).

TRS-80 Model I w/Level 1 AND 2 (switchable), 48K on board, sound built in, high speed mod, and lowercase fitted. $200. G. Dehayr, 204 1589.

Model 4 complete with two standard disk drives. $1200 o.n.o. Contact Bill Allen 343 5771.

T R S - 8 Ø S Y S T E M - 8 Ø C O M P U T E R G R O U P

BITS & BYTES

ISSUE NO. 41

WHEN IS EASTER ?

As this issue of BITS & BYTES will be published around about Easter time, the following three Basic programs which calculate the date when Easter Sunday falls in any given year are topical.

The first program is from Jack Bognuda, the second was published in the last isssue we received from "NORTHERN BYTES" Issue 6.8, and the third program was one that our Editor had lying around in his bottom drawer. This third one has a few more frills in that it calculates the date of Easter Sunday for each year between any two given years and it will also send the results to the printer if desired.

The editor of "Northern Bytes" had the request that if anyone could extract the basic mathematical algorithm used in the program that he had published, to send it to him as it could prove more useful in that format, particularly to readers coding in languages other than Basic. Perhaps our mathematical buffs may be able to set out the conditions involved, and the comments in the third program below might give some clues. We will publish any comments that we receive in our next issue.

Note: Except to say that all three programs gave the same date for the Easter Sunday of the Public Easter Holiday this year, 1986, we accept no responsibilty for the validity of the dates that any of the programs give over a wide range of years, (nor for the difference in some religous faiths, for example using a modified Gregorian calander, and the actual date of the Equinox).

Program # 1 - From Jack Bognuda

```
100 REM DATE OF EASTER SUNDAY
110 REM *** J BOGNUDA ***
120 CLS
130 INPUT "YEAR EG.1986 ";Y
140 N=Y: D=100: GOSUB 280: B=Q: C=R
150 N=5*B+C: D=19: GOSUB 280: A=R
160 N=3*B+75: D=4: GOSUB 280: E=Q: F=R
170 N=8*B+88: D=25: GOSUB 280: G=Q
180 N=19*A+E-G: D=30: GOSUB 280: H=R
190 N=11*H+A: D=319: GOSUB 280: M=Q
200 N=300-60*F+C: D=4: GOSUB 280: J=Q: K=R
210 N=2*J-K-H+M: D=7: GOSUB 280: L=R
220 N=H-M+L+110: D=30: GOSUB 280: S=Q: T=R
230 N=T+5-S: D=32: GOSUB 280: P=R
240 Z$="APRIL": IF S=3 THEN Z$="MARCH"
250 PRINT: PRINT "  EASTER SUNDAY" Y" :  "Z$P
260 PRINT: PRINT: GOTO 130
270 REM Subroutine for Quotient & Remainder
280 Q=INT(N/D): R=N-D*Q
290 RETURN
```

Program # 2 - From Northern Bytes

```
100 REM DATE OF EASTER SUNDAY
110 REM *** FROM NORTHERN BYTES - ISSUE 6.8 ***
120 CLS
130 INPUT "YEAR EG.1986 ";Y
140 N=Y-1900
150 Z=N/19
160 V=INT(Z)
170 A=(Z-V)*19: A=A+.05: A=INT(A*10)/10
180 B=((7*A)+1)/19: B=INT(B)
190 T=((11*A)+4-B)/29
200 M=(T-INT(T))*29
210 Q=N/4: Q=INT(Q)
220 D=(N+Q+31-M)/7
230 W=(D-INT(D))*7: W=W+.05: W=INT(W*10)/10
240 E=25-M-W: E=E+.05: E=INT(E*10)/10
250 PRINT: PRINT "  EASTER SUNDAY" Y" :  ";: IF E>0 THEN PRINT"APRIL"E ELSE
    PRINT"MARCH"E+31
260 PRINT: PRINT: GOTO 130
```

Program # 3 - From Editor's bottom drawer

```
  5 REM *** FROM SOURCE UNKNOWN ***
 10 GOSUB 20: GOTO 30
 20 CLS: PRINT@15,"**  E A S T E R   T A B L E  **": RETURN
 30 PRINT: PRINT"THIS PROGRAM WILL GIVE THE DATE OF EASTER SUNDAY FOR ANY YEARS"
    :PRINT
 40 INPUT"ENTER <L> IF PRINTER CONNECTED";L$
 50 PRINT: INPUT"ENTER THE FIRST YEAR REQ'D. ";Z1
 60 PRINT: INPUT"ENTER THE LAST YEAR REQ'D.  ";Z2
 70 IF Z2<Z1 THEN PRINT"THAT'S LESS THAN YOUR FIRST YEAR - TRY AGAIN !": FOR T=1
    TO 1200: NEXT: GOSUB 20: GOTO 50
 80 CLS: PRINT"* THE DATES ON WHICH EASTER SUNDAY FALLS FROM"Z1"TO"Z2"*": PRINT
 90 IF L$="L"THEN LPRINT"** THE DATES ON WHICH EASTER SUNDAY FALLS FROM "Z1" TO
    "Z2" **": LPRINT
100 Y=Z1: GOSUB 400
110 REM     Divide YEAR by 4,       A=Remainder
120 A=Y-INT(Y/4)*4
130 REM     Divide YEAR by 7,       B=Remainder
140 B=Y-INT(Y/7)*7
150 REM     Divide YEAR by 19,      C=Remainder
160 C=Y-INT(Y/19)*19
170 REM     Divide 19*C+P BY 30 ,   D=Remainder
180 D=(19*C+P)-INT((19*C+P)/30)*30
190 REM     Divide 2A+4B+6D+Q by 7, E=Remainder
200 E=(2*A+4*B+6*D+Q) -(INT((2*A+4*B+6*D+Q)/7)*7)
210 REM   F=Number of DAYS to EASTER SUNDAY after MARCH 22
220 F=D+E
230 G=22+F
240 IF G>31 GOTO 290
250 PRINT Y"     EASTER SUNDAY IS    MARCH "G
260 IF L$="L" THEN LPRINT Y"  MARCH "G
270 Y=Y+1: IF Y>Z2 THEN PRINT: END
280 GOTO 110
290 G=G-31
300 REM (EXCEPTION) If F=35 EASTER is APRIL 19, NOT APRIL 26
310 IF F=35 THEN G=G-7
320 REM (EXCEPTION)  IF F=34 and D=28 and C>10 EASTER is APRIL 18, NOT APRIL 25
330 IF F=34 AND D=28 AND C<10 THEN G=G-7
340 PRINT Y"     EASTER SUNDAY IS    APRIL "G
350 IF L$="L" THEN LPRINT Y"  APRIL "G
```

```
360 Y=Y+1: IF Y>Z2 THEN PRINT: END
370 GOTO 110
390 REM   *** Subroutine to calculate P and Q
400 K=INT(Y/100)
410 L=INT(K/4)
420 M=INT((K-17)/25)
430 N=INT((K-M)/3)
440 X=15+K-L-N
450 P=X-INT(X/30)*30
460 Z=4+K-L
470 Q=Z-INT(Z/7)*7
480 RETURN
```

----------oOo----------

ASSEMBLY LANGUAGE ROUTINES
by Glen McDiarmid

EXCHANGE TWO BLOCKS OF MEMORY

The following is a subroutine to mutually exchange two blocks of memory. It is 9 bytes long. Time taken is 59 T states per byte exchanged. HL contains the "FROM" address, DE contains the "TO" address, and BC contains the number of bytes to be exchanged.

```
100 SWAP    LD      A,(DE)
110         LDI
120         DEC     HL
130         LD      (HL),A
140         INC     HL
150         RET     PO
160         JR      SWAP
```

If very particular about timing, the following subroutine is slightly faster (52 T states per byte), but is one byte longer.

```
100 SWAP    LD      A,(DE)
110         LDI
120         DEC     HL
130         LD      (HL),A
140         INC     HL
150         JP      PE,SWAP
160         RET
```

COMPARE TWO BLOCKS OF MEMORY

The following subroutine will compare two blocks of memory, byte for byte, and RETurn with the Zero flag set if they are identical. HL points to one block, DE points to the other, BC contains the size of the blocks. Size is 9 bytes, timing is 66 T-states per byte compared. Registers HL,DE,BC and AF are used. On RETurn, if the blocks are not identical, HL and DE will be pointing to the offending bytes. Otherwise, they will be pointing to one byte past the end of the blocks.

```
100 COMPARE LD      A,(DE)
110         CP      (HL)
120         RET     NZ
130         XOR     A
140         LDI
150         RET     PO
160         JR      COMPARE
```

LOAD HL FROM HL INDIRECTLY

When using tables of 16-bit information (such as addresses), it is often desirable to load HL from HL indirectly. The following is a subroutine which will achieve this without affecting any other registers.

```
100             LD          (HLINDR),HL
110             LD          HL,(0000H)
120 HLINDR      EQU         $-2
130             RET
```

A ONE BYTE "SKIP" INSTRUCTION

It is possible to implement a one-byte "skip" instruction from some of the unused ED opcodes. When the Z80 comes across an ED code followed by one of the codes in the list below, it simply adds 2 to the Program Counter. None of the other registers are affected at all, not even the flags.

The Unused ED opcodes are:
0 to 3FH, 77H, 7FH, 80H to 9FH, 0A4H to 0A7H, 0ACH to 0AFH, 0B4H to 0B7H, 0BCH to 0FFH.

The following code:

```
100             BIT     4,(HL)
110             JR      Z,CLEAR
120             INC     D
130             JR      JOIN
140 CLEAR       XOR     A
150 JOIN        (rest of code)
```

may be coded as:

```
100             BIT     4,(HL)
110             JR      Z,CLEAR
120             INC     D
130             DEFB      0EDH
140 CLEAR        XOR      A
150 JOIN        (rest of code)
```

In the second block of code, if bit 4 of (HL) is non-zero, line 120 (INC D) will be executed, after which the CPU will execute an ED AF instruction (lines 130 and 140 combined), which will have the same effect as two NOP's, one after the other. The timing is identical, too. (8 T states) The second block of code is one byte shorter than the first, due to the one-byte skip instruction used.

----------oOo----------

SCROLL SCREEN ROUTINE

By Ronald J. Sully

This routine which will scroll the video screen either left or right is another example of how we may extend the limit of BASIC into the machine language world.

There is very little I need to say about the program as the source code of the machine language program includes a remark with every line and the BASIC program is simple enough not to need explaning. I need to mention though that the machine code is relocatable so 16K'ers can easily modify the BASIC program to suit their machines. Line 40 of the BASIC program will need to be deleted as it is appropriate to DISK BASIC only. Line 50 serves the same purpose for LEVEL

2 but the memory address will need to be changed. I suggest changing the &HFF to &H7F. The memory addresses in line 3000 should then be changed so that the line reads:

```
3000 FOR X = 32704 TO 32760: READ Y: POKE X,Y: NEXT
```

The BASIC program will allow you to load and execute the machine language program for demonstration purposes.

If you have any queries regarding either program please don't hesitate to see me about them.

Source Listing

```
00010 ;SCROLL/CMD V1.0  Aug. 85
00030 ;By Ronald J. Sully
00040 ;
00050 VIDEO   EQU   3C00H          ;Start of VIDEO RAM
00060 ;
00070         ORG   0FFC0H         ;ORIGIN
00080 START   CALL  0A7FH          ;Get Value in USR(N)
00090         BIT   1,L            ;Test if 1 or 2
00100         JR    NZ,RIGHT       ;If 2 goto RIGHT
00110 ;
00120 LEFT    LD    HL,VIDEO+1     ;VIDEO RAM
00130         LD    DE,VIDEO       ;Ditto
00140         LD    B,16           ;No. of Lines
00150 LOOPL   PUSH  BC             ;Save Line count
00160         LD    A,(DE)         ;Get Character
00170         LD    BC,63          ;No. of Characters in Line
00180         LDIR                 ;Move 'em
00190         LD    (DE),A         ;Restore Char. on other End
00200         INC   HL             ;Adjust to next Line
00210         INC   DE             ;Ditto
00220         POP   BC             ;Retrieve Line Count
00230         DJNZ  LOOPL          ;Dec & continue if not 0
00240         JR    FINI           ;No more Lines
00250 ;
00260 RIGHT   LD    HL,VIDEO+62    ;VIDEO RAM
00270         LD    DE,VIDEO+63    ;Ditto
00280         LD    B,16           ;Line Count
00290 LOOPR   PUSH  BC             ;Save it
00300         LD    A,(DE)         ;Get Character
00310         LD    BC,63          ;No. of Characters to move
00320         LDDR                 ;Move 'em
00330         LD    (DE),A         ;Restore Char. to other end
00340         LD    DE,127         ;Line Pointer in HL and
00350         ADD   HL,DE          ; DE is decremented during
00360         PUSH  HL             ; LDDR so adjustment is
00370         POP   DE             ; made by adding 127 to HL
00380         INC   DE             ; and 128 to DE.
00390         POP   BC             ;Retrieve Line Count
00400         DJNZ  LOOPR          ;Dec & Continue if not 0
00410 ;
00420 FINI    RET                  ;Back to BASIC
00430         END                  ;THAT'S ALL FOLKS
```

The Basic program to load the machine language program for a demonstration is on the following page.

```
 5 GOTO 10
 7 SAVE"SCROLL/BAS": STOP
10 POKE 16561,190: POKE 16562,255
20 CLEAR 1000
30 GOSUB 3000
40 DEFUSR=&HFFC0:                          'FOR DISK BASIC
50 POKE 16526,&HC0: POKE 16527,&HFF:    'FOR LEVEL 2
60 PRINT"
   PRESS LEFT ARROW TO SCROLL LEFT
   OR RIGHT ARROW TO SCROLL RIGHT"
70 I$=INKEY$: IF I$="" THEN 70 ELSE S=ASC(I$)-7
75 IF S>2 THEN 70
80 X=USR(S)
90 GOTO 70

2999 '        DATA STATEMENTS FOR MACHINE CODE

3000 FOR X=-64TO-8: READ Y: POKE X,Y: NEXT
3010 RETURN
3020 DATA 205,127,10,203,77,32,23,33,1,60,17,0,60,6,16
3030 DATA 197,26,1,63,0,237,176,18,35,19,193,16,243,24,26
3040 DATA 33,62,60,17,63,60,6,16,197,26,1,63,0,237,184
3050 DATA 18,17,127,0,25,229,209,19,193,16,238,201
```

----------oOo----------

PRINTER PROBLEMS SOLVED ON A MODEL 4P

by Bill Allen (07) 343 5771

Most likely, this article also applies to the Model 4 as well as the 4P, but I haven't had the opportunity to check it out on the Model 4.

On hooking up my Copal SC-1000a printer , an Epson workalike, with a plain printer cable to my 4P, a lot of problems immediately became obvious -- I couldn't get it to print! On my friend, Peter Goed's advice, I cut lines 28 to 33 inclusive on the printer cable just next to the plug on the computer end. [Standing behind the computer and facing it (and with the plug in position), the printer interface cable lines number 1 on the right (the same side as the power switch on the front) to 34 on the left.] The printer came to life when asked to do hard copy and operated satisfactorily after these lines were cut. But later, when I tried to do underlining with Lazywriter, which with this printer must be done by the two-pass method, the printer always did a line feed before making the second pass, despite what I knew were correct parameters in the software driver.

To do this underlining on the Copal, you also have to switch the auto line feed off with dip switch #7 on the printer, so the software can control line feeds. So, I went back to NEWDOS READY to check that the switch was indeed turned off by dumping the screen to printer with JKL. To my amazement, it doggedly still did line feeds! OK, I thought, maybe the Model 3 ROM image had some surprises by the way of differences to the code I was used to on the System 80. No, that idea proved fruitless. OK, try TRSDOS 6.2 -- what, still line feeds with the dip switch #7 off! The answer MUST be in the hardware!

Looking up the printer interface connections for the 4P (for the umpteenth time) and an extended phone call to Peter Goed, found the culprit -- Line 26! The Model 4/4P Technical manual gives the connections and they look normal enough. But, some of the lines are designated as NC and to me, for the last 40 or so years, this has meant that those pads aren't connected to anything (i.e.

electrically isolated)! It seems that over there in Fort Worth, NC has a different meaning -- such as "DON'T connect anything to this line!" However, of course, the Tandy printers are obviously wired with lines open at the printer end or some subterfuge such as missing pins on the plug -- a typical Tandy solution. Possibly these lines have some use in more (or maybe less) sophisticated printers, as they turned out (from an inspection of a circuit diagram -- obtained with great difficulty from a Workshop Manual) to be, in fact, connected to logic gates on the interface -- but Tandy are keeping the relevant info under their hats by hiding behind the NC in the Technical Manual. If we were in possession of the facts before making up a cable, we could follow their method of removing the unwanted pins from the plug instead of having to mutilate the cable. I am undecided as to which method is really the best. At least theirs looks tidier, but the cuts I made are concealed by the rear flap, anyway.

It turns out that the lines that definitely have to be cut or disconnected when interfacing a 4P to this type of printer which is marketed under a variety of badge engineering -- BMC1ØØ etc) are: 33 NC), 31 (NC), 28 (Fault), and 26 (NC) -- the others (32,3Ø,29 -- all N/As) were also cut, just in case. Once Line 26 was cut, I got the proper underlining function to work. This line is grounded at the printer end and this somehow caused an automatic conversion of ALL carriage returns to line feeds (pretty clever and rather unexpected for a grounded NC!).

To assist any reader with similar problems, here are the Tandy pad designations of the printer interface of the 4/4P, in case they are not in your possession (Pad 1 is at the power switch side of the computer, 34 on the disk drives side):

UPPER SIDE PADS	LOWER PADS
1. DATA STROBE	2. GND
3. DATA BIT Ø	4. GND
5. DATA BIT 1	6. GND
7. DATA BIT 2	8. GND
9. DATA BIT 3	1Ø. GND
11. DATA BIT 4	12. GND
13. DATA BIT 5	14. GND
15. DATA BIT 6	16. GND
17. DATA BIT 7	18. GND
19. N/A	2Ø. GND
21. BUSY	22. GND
23. OUTPAPER	24. GND
25. UNIT SELECT	26. NC
27. GND	28. FAULT
29. N/A	3Ø. N/A
31. NC	32. N/A
33. NC	34. GND

To further confuse the issue, the 36-pin D-connector at the printer end of the cable has a completely different numbering convention compared with the computer end. Whereas the CABLE lines were numbered in sequence at the computer end, the D-connector is numbered radially, the only common number with the other end being line 1. Here is a comparison of the plug pin numbers for each line of the cable:

Computer end/Printer end: 1/1, 2/36, 3/2, 4/35, 5/3, 6/34, 7/4, 8/33, 9/5, 1Ø/32, 11/6, 12/31, 13/7, 14/3Ø, 15/8, 16/29, 17/9, 18/28, 19/1Ø, 2Ø/27, 21/11, 22/26, 23/12, 24/25, 25/13, 26/24, 27/14, 28/23, 29/15, 3Ø/22, 31/16, 32/21, 33/17, 34/2Ø

(18 & 19 are not used on the D-connector because we are coming from a 34-way interface to a 36-way connector and the 34-line cable doesn't reach pins 18-19).

"MID$ =" FUNCTION

by Alf West

Most Members would be familiar with the "MID$" function, but those graduating from Level II to Disk Basic may not be so familiar with the "MID$=" function. Perhaps the best way of illustrating the difference between the two functions is by running and examining the following little program.

```
100 CLS: PRINT"USING THE 'MID$' FUNCTION :-"
110 A$="ABCDEFGHIJKL": B$="12345"
120 B$=MID$(A$,4,5)
130 PRINT"A$ = "A$,"B$ = "B$

200 PRINT: PRINT"NOW USE THE 'MID$=' FUNCTION :-"
210 A$="ABCDEFGHIJKL": B$="12345"
220 MID$(A$,4,5)=B$
230 PRINT"A$ = "A$,"B$ = "B$
```

In both sections of the program, we start off with the same strings for A$ and B$ (in Lines 110 and 210). The only difference is that :-
In Line 120 we make B$ equal to MID$(A$,4,5) whereas
In " 220 " MID$(A$,4,5) equal to B$.

With the "MID$" function, A$ is unchanged, and B$ takes the value of that part of A$ we have nominated.
With the "MID$=" function, B$ is unchanged, but A$ is modified and now includes B$ in the part of it that we specified.

With both functions, it is normal to include two numbers in the bracket after the string (A$ in this case). What happens if the second number is not given? Try it out and see for yourself - delete ",5" from lines 120 and 220. You will note that with the "MID$" function, the answer for B$ is all the remainder of A$ from position 4. On the other hand, omitting the second number in the "MID$=" function doesn't make a scrap of difference to the answer for A$.

But let's look at "MID$=" a little more and now go back to the original program, and in Line No. 210 make:

```
B$="1234567890123456789"
```

With Line 220 as it was initially, namely "MID$(A$,4,5)=B$" the answer for A$ is the same as it was originally, as you might expect. Now delete the second number, making MID$(A$,4)=B$. You might have anticipated that the answer for A$ would be "ABC" plus the whole string of numbers in B$, but not so. You'll note that the length of A$ is no greater in length than it was originally - all the additional characters in B$ have been ignored.

It follows from this that by using "MID$(A$,1)=B$", we can change the whole of A$, if B$ has the same number of characters in it (or more) than did the original A$ - 12 characters in this example.

"MID$" and "MID$=" with the appropriate numbers, or number, both have their uses in ordinary string manipulation. However the "MID$(A$,1)=B$" form of the "MID$=" function is particularly valuable in manipulating large amounts of data and avoiding that dreaded string "hang-up" which can occur.

What causes these "hang-ups". Perhaps it can best be explained by imagining a teacher in a class room, and asking the kids to give him a list of nouns. He starts off with a clear blackboard, and on the first answer, he writes "A$ = DOG". On the next answer, he crosses out DOG and writes CAT, then crosses that

out and writes HORSE and so on and so on. But then the blackboard is full of crossed out names, so the kids have to wait until the teacher cleans up the blackboard before he can write the next name. That is virtually what happens in the computer and with "silly" programming, I've had a "hang-up" last for nearly three quarters of an hour, before it cleaned up the blackboard.

The situation can be worse if you say A$="DOG", B$="CAT", C$="HORSE" etc. because the computer is asked to keep these names as separate variables. The amount of string space that is CLEARed must cater for all the different string variables that are used in the program otherwise an "out of string space" error will occur. Beyond that, then the more you CLEAR, the longer it will take for a "hang-up" to occurr, but when it does, the longer it will take before the program resumes running.

Coming back to the teacher and the blackboard, instead of what he did before, we'll assume that he allocates a space on the blackboard for say 12 letters. He puts the first answer DOG in that space. On the second answer of CAT, he rubs out the previous letters and puts CAT in the same space. On the next answer, HORSE, he rubs out and overwrites again and so on and so on and he can keep going that way ad infinitum (assuming he has enough chalk.) This in effect is what happens using the "MID$(A$,1)=B$" form of the "MID$=" function and like the teacher, the computer doesn't run out of blackboard or string space.

Just to show the difference in the action of the computer, with the two different methods used by the teacher, try firstly the following program :-

```
100 CLEAR 10000: DIMA$(200): CLS
110 PRINT@128,"INITIAL NAMES";
120 FOR J=1 TO 200
130 A$(J)="ABCDEEFGHIJKLMNOPGRSTUV"+CHR$(64+RND(26))
140 PRINT@148,CHR$(30)"NO."J TAB(35)A$(J)
150 NEXT
160 I=I+1
170 PRINT@128,"CHANGE NO."I;
180 GOTO120
```

Here we are making up a little list of only 200 names and to make it easy, in Line 130 a random letter is added to a string just to make the names different. After that, from Line 180 we are in effect, going to cross them out and put another 200 names on the blackboard. Each name is 23 characters long, so one set is 4600 bytes, and we will make our blackboard big enough to give us 10,000 bytes (CLEAR 10000). Now RUN this program, and as can be expected from the blackboard analogy, the computer hangs up for a while during the second change while it cleans the blackboard, or cleans up the string space.

Now let us use the "MID$=" function in the following program :-

```
100 CLEAR 10000: DIMA$(200): CLS
102 FOR J=1 TO 200
104 A$(J)=STRING$(23," ")
106 NEXT
110 PRINT@128,"INITIAL NAMES";
120 FOR J=1 TO 200
125 MID$(A$(J),1)=STRING$(23," ")
130 MID$(A$(J),1)="ABCDEEFGHIJKLMNOPGRSTUV"+CHR$(64+RND(26))
140 PRINT@148,CHR$(30)"NO."J TAB(35)A$(J)
150 NEXT
160 I=I+1
170 PRINT@128,"CHANGE NO."I;
180 GOTO120
```

In this case we start with allocating a space on the blackboard (in the string space) for our 200 names, using Lines nos. 102 to 106. Notice the difference in Line 130 where by using the "MID$=" function, the relevant name will go in the space which has been allocated to it. From Line 180, we get to Line 125, which is analogous to the teacher rubbing out the space ready to accept the new name. (In this example, Line 125 is not strictly necessary because the names are all the same length, and the new name would completely over-write the old one, but it is necessary if the strings were different lengths.) RUN this second program and you will find that you have no "hang-ups".

Perhaps the following comments may clear up some misconceptions about how the free or CLEARed string space is used in the computer.

If the program has a line :- 120 A$ = "THIS IS A NAME"
no free string space is used at all as the String A$ is located in Memory at the point where it starts in the program line # 120.
On the other hand if the line was :- 120 INPUT" INPUT A NAME ";A$
whatever the operator input at this stage goes into the free string space as A$.

If the program includes DATA lines with words or strings which are later READ as A$, B$, etc., no free string space is used as the pointers for A$, B$, etc. will take on the addresses in Memory where those words or strings appear in the program. This <u>also</u> applies if the words or strings are READ into an arrray such as A$(1), A$(2), etc.

In general terms, if a string is specifically defined by word(s) within a program no free string space is used. However if the string is defined in the program indirectly, such as :- A$ = STRING$(10,42)
or A$ = CHR$(191) + CHR$(145)
then the string A$ is set out in full in the free string space.

With the statements :-
A$ = LEFT$(etc... A$ = RIGHT$(etc... A$ = MID$(etc...
the newly defined A$ is put into the free string space.
If a program statement was :- B$ = A$
the newly defined B$ is put into the free string space.

Any string entered by the operator in response to :-
INPUT LINE INPUT INKEY$ (1 chara.)
will be put in the free string space (there is nowhere else). The first blackboard analogy applies every time these statements are used, even if the variable say A$, was the same in every case.

No <u>additional</u> string space is used with the "MID$=" function, - only the particular characters within the specified string, which already exists in the string space, are changed.

In reading a data file, all the information in the file may be read and printed on the screen or the printer, and the only free string space which will be used is that of the strings in the last record in the file. (the data is read and printed from the file buffer.)
However if the items are put into arrays as the data file is read, then every string in the data file will be put into the string space.

A data file buffer may be used for the <u>manipulation</u> of strings to reduce the use of the free string space with such statements as :-
FIELD 1, 15 AS B$ then LSET B$ = A$ or RSET B$ = A$
No free string space is used for B$, but it depends upon how A$ was defined (see lines # 120 above) whether string space is used for A$.

----------oOo----------