NATGUG NEWS

Volume 7 Issue 10 April 1986

OFFICIAL JOURNAL OF THE

National TRS-80

& Genie Users

Group.

INFORMATION ON THE GROUP

Membership of the group is by subscription to the Newsletter, which is published monthly. Membership details are obtainable from the Group Secretary. Membership of the group is open to anyone with an interest in computers but special emphasis is placed on equipment in the TANDY range.

Details of the Group accounts, and the constitution of the Group, are available from the Secretary.

Members requiring assistance with problems related to the TRS-80 / Video Genie may call the Secretary. An attempt will be made to put them in touch with a member who can help with the problem.

Workshops are arranged from time to time in various parts of the country.

Sub-groups exist in many areas. A list is provided in the Newsletter from time to time.

The Group maintains two software libraries (Models I and II) which are free to members. Library lists are available from the Secretary.

For confidentiality reasons, the membership list is not generally available, but members may ask the secretary for a list of members in their area, and mailshots to all members may be arranged.

Back numbers of the Newsletter are available from the Secretary.

Please send all contributions for the Newsletter to the Editor, on disk if at all possible (5.25", NEWDOS-80 v2 or Montezuma Micro CP/M prefered, any combination of density, sides or tracks, but please say what it is). Your disk will be returned.

Newsletter Editor
Geof Smith
17 Homefield Rd.,
Bushey,
Herts WD2 3AP
01-950-6345

Secretary & Newsletter Publisher
Brian Pain
24 Oxford Street,
Stony Stratford,
Bucks. MK11 1JU
0908-564271

CONTENTS

EDITORIAL

Welcome to a different set of biases and opinions on computing in general and Tandy in particular. At present I'm not too certain whether or not I'm presiding over a wake or a re-birth. This week I've had a phone call from Brian saying that he's already had a response from interested Amstrad owners but I've also had a phone call from one of the silent majority saying he was not renewing his membership, had gone to CP/M and was planning to go MSDOS and could we sell his Model 1. What is clear is that NATGUG's collective experience in Level II BASIC, and TRS-80 specific operating systems and hardware is no longer in great demand. Those of us left are in it for the fellowship as much as for guidance in matters computational. If the computer aspect of the group is to survive then we must evolve or we will be ousted from our ecological niche.

The last Swindon weekend seemed to be showing a great upsurge in the use of and interest in CP/M. This probably offers temporary sanctuary especially if the recruitment of PCW 8256/8512 members is successful. We are having Dave Holman's series on CP/M reprinted as a booklet which may be of use to those just starting on the CP/M trail. Has anyone any suggestions for expansion or addition to the topics covered in his articles. It would also be helpful to receive some more reviews and comparisons of the multitude of CPM software, both commercial and public domain. Would anyone care to compare and contrast the merits or otherwise of Wordstar, Spellbinder or Perfect-Writer for instance.

Another possible bridge for crossing the machine divide is with the use of portable languages. In this context C and Turbo Pascal spring to mind. Leon has been extolling the virtues of C for a number of years and its portablity has improved of late. However, I still find that its source code is pretty cryptic for my tastes and the lack of an efficient run-time debugging system has always held up my use of the language for anything other than trivial utilities. Perhaps someone out there is using one of the more modern implementations with proper source level debugging such as Living-C or an interpreted version such as Instant-C. A review or even just a quick note would be very useful. These latter two are 16 bit C's, for us more 'umble mortals MIX-C at $39.95 would seem excellent value.

Turbo Pascal by Borland International gets my vote as one of the best computing environments that I've come across. The speed of the compiler almost offset's the time penalty of compilation, and the close interaction of the error trapping with the editor makes debugging relatively easy. This coupled with the fact that the compiler is available for CP/M-80 & 86, PCDOS & MSDOS, the Mac and the Amiga makes for almost universal portability of code amongst micros. We even port the code to our DEC-20 mainframe with only minor changes to the I/O. When you consider that you get all of this for £49.00, it is no wonder that it has become the de facto standard. In this issue

of the newsletter we have a very brief introduction to Pascal and our first small Turbo Pascal program. I hope that this is the first of many and if any of you have any small utilities or even subroutines written in either C or Pascal I for one would very much like to see (!) them. Whilst on the subject of Turbo, I note that Borland has just released Turbo Prolog for MSDOS systems, this appears to be an incremental compiler which really does give you the best of both interpreted and compiled situations - do we have any members interested in AI? On the CP/M front Borland appear to be on the point of releasing Turbo Modula-2. This language is Niklaus Worths' successor to Pascal and has actually been designed to be used rather than taught as was the case with Pascal. The early indications are the language is very good indeed and may succeed C as a language for system implementation as well as the more usual computational usages.

The area of computer communication is also a field which offers great opportunity for cross system interest. I have started in this issue the first part of three or four articles on Kermit - which is basically a computer to computer communication protocol. I would very much like to see some information on one of the other popular protocols such as MODEM-7 so that we can compare the strengths and weaknesses of the available systems and sort out the horses for courses. Great developments also appear to be taking place in both hardware and software in the form of new super-whizzo do everything modems and terminal programs (do we actually need SYSOPS now I wonder ?). Would anyone like to tell those dithering on the edge about the pros and cons of the new goodies.

I make no aplogies for a rather long first editorial. There was plenty of space to fill and it has given me the opportunity to make some suggestions as to the direction I think the magazine should take. The rest is up to YOU. More ideas in the form of letters will be most welcome, but contributions are essential. The newsletter is being printed as a continuous file on a Daisy wheel printer rather than being a paste-up. I really do not want to have to re-type your contributions and so would greatly appreciate receiving them on a disk. I'm using CPM-Scripsit to prepare the Newsletter so any of the Montezuma or Genie CPM formats would be most convenient. However I can read virtually any 5.25 CP/M Format and also PC-DOS. If you are running under a Tandy operating system, then NEWDOS-80 v2 would be most suitable but again I can read any at a pinch. Your disk's will be returned to you.

I look forward to hearing from ALL of you - Ed.

PS. Thought for the month
- On a clear disk you can seek forever !

MEMBERS LETTERS

FUTURE DIRECTION OF NATGUG/QUANTA

Dear Ed.,

As a longtime user of TRS-I AND III, who has also a QL a Commodore 64 and an Amstrad 8256 within the group, I also wonder how long NATGUG can survive. As one who has had only a few hours on the Amstrad, I was amazed by the quality of the Locoscript (after Scripsit) and the ability to refer to a list of Files whilst retaining the letter in memory. Also the ease of printing in different fonts which my Scripsit is not patched to do.

In short we should jump on the Amstrad Bandwagon, especially in view of the proposed launch of their IBM compatible for around £500. Although I note that DOTWRITER is now again available in the U.K. (re David Washford - March Issue), I am not sure about its compatibility with my Tandy Lineprinter VI or my Seikosha GP100A Mark II, and in any case cheap programs running under CP/M are in plentiful supply. The Amstrad 8512 is the twin disc version with CP/M V1.4 and LocoScript V 1.2 and appears to be better value - at the moment. Again to rival Dotwriter there are the POLY/PROGRAMS which can print up to 25 distinctive Typefaces - are there programs to do this with my Model-III.

Incidently our Computers are used primarily for Business purposes, using Visicalc for designing and operating accounting systems, and Scripsit for letters etc.

We also have a 3M Copier for which we require something like Dotwrier to design Posters and Handbills.

JOHN JEFFERY TEL (DERBY) 0332-881239

PROBLEM PAGE

Alhough most of this problem seems to be directed at the QL, there may be someone with knowledge of obsolete technology - I mean just where do you buy quill pens these days ? - Ed.

SEIKOSA GP100 MARK II PRINTER

Dear Ed.,

As a recent owner of a QL (£200 including Ferguson Monitor - Model No 3MM02G5) I have had some difficulty in interfacing it with the above Printer. Incidently the price originally quoted for the interface was £34.00, but upon checking Sinclair QL World, the Price quoted at MIRACLE SYSTEMS was £19.50. The shop involved (Gordon Harewood - Alfreton (0773)832078 agreed to charge me £19.50 after copying the Advert.

However upon connecting the printer and programming it as a Centronics FX80, it proceeded to print in a very jerky fashion and also left a space between each line. I would like to know the precise instructions to be given when installng PRINTER DAT in QUILL.

Also can my TRS80 Model III, 40 track Disc drives be interfaced with the Q.L., and if so how do I address them and what interfaces do I require.

JOHN JEFFERY DERBY (0332) 881239 SCRIPSIT/NEWDOS

Ever thought about how many paradoxes there are in our everyday life ? As one of the more vociferous attackers of the QL - and I must admit that this was due to downright jealousy after seeing some fantastic screens (particularly when compared to a Model I) - I was quite upset to read of its virtual demise following the sellout to Amstrad. Paradox number 1.

In the many and varied reports that assailed us during the days that followed, most of the so-called experts agreed that the failure of the QL to capture the market had absolutely nothing to do with the machine itself; the flop was solely due to the refusal of software houses to write and publish QL programs. Now for paradox number 2. The so-called NATGUG experts have always asserted that the strength of the TRS-80 (Models I, III, or IV) was in the phenomenal amount of software available for it - so why have Tandy UK deserted us and turned instead to selling the Amstrad, which, for all its good points, doesn't run Tandy software ?

Many members were surprised by the intent by NATGUG to court the Amstrad user, whilst many others were disappointed that we didn't pursue the PC and MS-DOS people. Well, guess what ! Mr. Sugar, of Amstrad, has hinted that he could soon be selling an IBM clone - for around £500 ! Guess where NATGUG will be heading then!

Back to our support for Tandy software houses. I had a phone call recently from the new and highly delighted owner of Zedcor ZBASIC. He assures me that this has completely transformed his Model 4 - and that the publishers have told him that he is the ONLY owner in the UK. Well now, how about that as a bad example of our support ? Those of you who missed Vernon Blackmore's letter in last month's magazine should perhaps read page 8 again

At the time of writing, Os is still making plans for BLANDFORD '86, but it looks as though the date will be two weeks after the August Bank Holiday, so pencil SEPT 7TH in your diary now, and watch this space for further announcements ! You did write down the SWINDON date, didn't you ? 17th - 19th OCTOBER - and it's time to be letting us know what YOU want on the programme.

David Washford, 6 Houston Way, FROME, Somerset BA11 3EU. (0373) 72739

(With respect to Dave's note on NATGUG's amorous intentions, we DO intend to court MSDOS users but the AGM felt that we needed more funds for advertising etc than were at present available and that a successful recruitment of PCW 8256/8512 users might fund this. Also the group's expertise with MSDOS is nowhere near as extensive as with CPM and therefore we had little to offer in terms of a knowledge base for either hardware or

software. Six months or so would enable those of us with MSDOS/PCDOS machines to become expert - the definition of an expert being someone who is one chapter ahead of you - Ed.)

CUMANA DISK DRIVES - David Roberts

Last November I wrote to the Newsletter commenting that by greasing the guide rails of my Cumana Shugart twin DS drives, the noise level had been reduced. I also stated that it seemed to cure the NOT READY ON DRIVE A: message. How wrong I was on the latter. Soon afterwards the trouble reappeared and was so persistant that I lost confidence in their use and was beginning to use the infernal Model 4 40 track drives. As some of you might know, we are fortunate here in Northern Ireland to have Don Bannister and Trevor Hutchinson as the local 'Gurus'. Even these two esteemed gentlemen were foxed by the problem.

I finally wrote to the Engineering manager of Cumana, Mr C. Magee, who wrote back suggesting that I should first check that the terminating resistor pack was in place on the last drive and failing that, check the two 40/80 track switches for correct position. Apparently the middle position is a reset position. These were checked out with no avail.

One evening Trevor had a final look at the drives and with surprise in his voice said something to the effect: 'Cor... who ever saw power regulators of a power supply plugged into sockets!!!'. He had struck gold. A follow-up phone call to Cumana confirmed that this was an old practice (my drives are only one year old) and that they had discontinued this due to flexing of the board causing intermittent faults. He also promised to send a metal shield to fit over the power supply as in later models. This I recieved two days later.

Congratulations Cumana for super service. So the final answer was to remove the socket and solder regulators to the board directly. 100% operation and confidence thanks to Trevor, Don and Cumana.

So, if anyone in the Group has Cumana Drives, remove the cover, look under the perforated power supply cover and if you find the same setup, you know what to do.... otherwise you are living on borrowed time and someday when you are in the middle of a long article to NATGUG NEWS (come on ...let's hear your problems and solutions), you will be looking in disbelief at the NOT READY ON DRIVE A: message.

David Roberts. Bangor, Co. Down

A TASTE OF PASCAL - Steve Ridall

Pascal, in the same way as BASIC, was designed as a teaching aid. It was not however, designed with another language in mind but rather as a method of encouraging structured programming. Contrary to popular opinion, PASCAL is not a difficult language to learn, in fact I would argue that learning PASCAL is easier than learning BASIC. Some features of the language, ie. pointers can be a little daunting, but very effective use of the language can be made without some of the more exotic features (pointers are in fact similar to the VARPTR command in BASIC.

Subroutines (known as procedures) are very straitforward and can be nested within other procedures. Variables can be of any length, depending on the compiler (usually 99 characters may be specified but only the first 8 or so are significant). One aspect that does take a little getting use to is that 'assignment' and 'equality' are treated differently. The '=' is used give the value to constants ie. pi = 3.1416.. or in tests for equality IF a = b THEN..... Assignment of values to variables is differentiated by use of a preceding colon ie. VALUE := 20 ; COST := VALUE * VAT. Remarks are enclosed in curly brackets or '(*...*)'. The terminator of an expression is ';' (in BASIC a similar symbol is ':'). Perhaps the best way to give the flavour of PASCAL is to show a small program.

Two different types of loops are demonstrated in the program below, the REPEAT and DO WHILE. You may notice that portions of the program are indented. You will find this with most PASCAL programs as it makes them easier to read and the structure somewhat clearer. This is not a requirement of PASCAL as long as a terminator (;) is used to separate statements then there is no reaso why they could not be written in one continual stream.

```
PROGRAM EXAMPLE (INPUT OUTPUT)
VAR cash,income,expenditure,total:INTEGER;
finished:BOOLEAN; (*finished can only be TRUE or FALSE*)
BEGIN
   finished:=FALSE;
   expenditure:=0;
   income:=0;              (*variables do NOT start out with
                              zero values by default*)
   WHILE NOT finished DO (*the loop BEGIN ... END will be
                          executed until finished is set to
TRUE*)
   BEGIN
      WRITELN('Cash Flow Program');
      WRITELN;               (*Print a blank line*)
      WRITELN('Enter Income or 0 to finish ');
      READ (cash);           (*input cash value*)
      IF cash = 0 THEN finished := TRUE;
      income := income + cash;
   END;

   finished := FALSE;
```

```
   REPEAT
   BEGIN
      WRITELN('Enter expenses or 0 to finish');
      READ(cash);
      IF cash = 0 THEN finsihed := TRUE;
      expenditure := expenditure + cash;
   END;
   UNTIL finished := TRUE;
   WRITELN('Total income is ',income);
   WRITELN;
   WRITELN('Total expenses ',expenditure);
   total := income - expenditure;
   WRITE('Total = ',total);
END.        (*All PASCAL programes must end with a full stop*)
```

One or two points that may need clarifying are .....

Line 1 In most high level languages, including PASCAL you must name the program in the first line, this one is called EXAMPLE. The hardware devices used and any files must be stipulated in the first line of the program. In this case keyboard (input) and VDU (output) are declared.

Line 2 All variables must be declared at the beginning of the program. This can be beneficial since a glance at the top of the program makes it very much less likely that you will choose the same variable twice. Variable must be declared as integer/real/boolean/character or you may specify your own. Arrays can be any of these types.

Line 4 Every block of code begins with a BEGIN and ends with an END, makes sense ?

Line 5 As BASIC except that you must use ':=' instead of '=' to reassign a value to the variable.

One of the most useful features of PASCAL is the records which are part of the specification of the language. Any field may of course be accessed within the record and in fact records may be enclosed within one such field. You could consider a date as an example of a record. A date has three components: the day of the month, the month and the year. Sometimes we are interested in the date as a whole ie what is your date of birth. Other times we are interested in particular components of the date, ie in what year will Halley's comet appear (NO FT, no comet ? Ed.). Thus the date can be considered as a record with 3 fields. Records must be declared at the beginning of the program with the TYPE statement.

```
TYPE dates  =  RECORD
               day  :1..31;
               month:1..12;
               year :0..9999;
               END
```

This could then be used to declare variables whose values are

```
dates
eg.   today,lastcomet,nextcomet : dates
```

a field designator would then be

```
                    nextcomet.year
```

and the statement

```
          nextcomet.year := lastcomet.year + 82;
```

would update the year field in the record variable nextcomet.

A list of some BASIC vs PASCAL statements is given below.

```
================================================================
BASIC              PASCAL                 COMMENTS

Print"Hello"       WRITELN('Hello);       The LN at the end gives a
                                          carriage return
PRINT(a + b)       WRITE(a + b);
INPUT(x)           READ(x);               x defined as
                                          integer/real/own-numeric
LINEINPUT a$       READLN(a);             A declared as an array of
                                          characters
x%/y%              x DIV y                DIV used for integer
                                          division
IF..THEN..ELSE     IF..THEN..ELSE
AND/OR/NOT         AND/OR/NOT
+,-,*,/            +,-,*,/                Except for DIV above

'FOR' loops
FOR x=1 to 10      FOR x:=1 to 10         If more than one statement
co = co + x        BEGIN                  is used then they must be
.......              co := co + x;        enclosed in BEGIN ... END
NEXT X             END

================================================================
```

PASCAL is a very nice language and I have only covered a few of its merits. One of its limitations is in string handling capabilities although the UCSD P system includes strings (Borland's Turbo PASCAL has quite reasonable extensions for strings - Ed.). A good introduction to PASCAL is 'PASCAL an introduction to methodical programming' by W. Findlay & D. Watt, published by Pitman, ISBN 0-273-02188-5.

Steve Ridall. - Hornsey, London

Execute from a Menu in Turbo PASCAL - Dave Roberts

Now that the dark winter days are receding (look outside at the snow, rain and wind!!) and spring is in the air, a married man's thoughts turn to..... well.... what can he usefully program in Pascal.

Wouldn't it be nice to have a little program which would run on booting up and would display a menu of the main programs on the disc and which could be run by pressing only one key. Now I'm not saying that I'm too lazy to type the program name but I'm thinking rather of my spouse who from time to time wants to use Wordstar. Come to think of it, some of my colleagues at work might appreciate the same facility on the Sirius.

Now writing normal self-contained Pascal programs using data files isn't too difficult, but where do I begin when I want to call external programs from within the Pascal program? Most of the books on the subject of Pascal don't have much if anything to say on the problem. After several nights thumbing through various books and manuals I stumbled across the pre-declared procedure EXECUTE. The following is the first 'play - program' which I used to test whether I can call programs such as Wordstar, Supercalc, dBase or whatever.

```
PROGRAM MENU;
VAR
  FILENAME: FILE;
BEGIN
   CLRSCR;
   GOTOXY(28,7);
   WRITELN('M E N U ');
   ASSIGN(FILENAME,'WS.COM');
   EXECUTE(FILENAME)
END.
```

```
PROGRAM MENU;
VAR
  FILENAME: FILE;
BEGIN
 CLRSCR;
 GOTOXY(28,7);
 WRITELN('M E N U ');
 ASSIGN(FILENAME,'DBASE.COM');

 EXECUTE(FILENAME)
END.
```

```
PROGRAM MENU;
VAR
FILENAME: FILE;
BEGIN
   CLRSCR;
   GOTOXY(28,7);
   WRITELN('M E N U ');
   ASSIGN(FILENAME,'SC2.COM');
   EXECUTE(FILENAME)
END.
```

```
PROGRAM MENU;
VAR
FILENAME: FILE;
BEGIN
 CLRSCR;
 GOTOXY(28,7);
 WRITELN('M E N U ');
 ASSIGN(FILENAME,'SUBMIT.COM
                       WS.SUB');
 EXECUTE(FILENAME)
END.
```

Of these four little examples only those which call SC2.com and DBASE.COM work. (I find that I have to put .COM after the filename). The problem with the one which calls WS.COM is that it will certainly call Wordstar but Wordstar continues on into the document mode and opens a file called WS.COM. Now this is a dangerous thing!!. Saving this file would overwrite Wordstar itself, hence a quick exit without saving. Why does this

happen? Anyone out there with any ideas how to overcome it?

A little known fact about Wordstar is that one can go straight into the document from the CP/M command line by typing :WS ½filename¾. Is this a clue to what is happening?

The fourth little example doesn't work either. All I get is the message 'Command Buffer Overflow' then it bombs out. Logically, it should work because SUBMIT.COM WS.SUB is used in the normal command line. I normally use a submit file to set up Wordstar. This includes re-defining the keyboard and PIPing the overlays to drive M:.

I would be grateful if any one can help me sort out this problem. They can then have a copy of the completed program.

For those interested, here is my submit file and Key definition file.

(NB. The daisy wheel I'm using does not have a caret, control is therefore represented by an upright bar.

WS.SUB

```
KEYDEF  WSKEY.KDF     (*this  copies  the  key  definition
file*)
PIP M:=A:WS*.OVR    (*copy overlays to MEMdisk*)
M:                  (*log onto drive M: MEMdisk*)
A:WS                (*call wordstar*)
ERA M:WS*.*         (*clears MEMdisk at finish*)
A:                  (*log onto drive A:*)
KEYDEF DEFAULT.KDF (*reset keyboard on exit*)
```

WSKEY.KDF

```
CL=|G       (*clear key = delete *)
BK=|C       (*break = Ctrl C*)
UP=|E       (*up arrow = Ctrl E*)
DN=|X       (*down arrow = Ctrl X*)
LF=|S       (*left arrow = Ctrl S*)
RT=|D       (*right arrow = Ctrl D*)
!UP=|[      (*shift up = escape*)
F1=|KD      (*Func 1 = Save & Done*)
F2=|B       (*Func 2 = reformat paragraph*)
F3=|OG      (*temporary para indent*)
```

DEFAULT.KDF

```
CL=|X
BK=|C
DN=|X
LF=|H
RT=|I
!UP=|[
!RT=I
!LF=|X
F1=MDIR A:|M
F2=MDIR B:|M
F3=MDIR C:|M
```

By copying the Wordstar overlay files to memdisk and logging onto drive M: you will find that wordstar runs much faster.

David Roberts. 6 Plantation Rd. Bangor, Co. Down.
Tel. 0247-462564 Prestel Mail Box 247462564

(The answer to your problem may lie in the way in which CP/M normally deals with command line tags. Having processed the main .COM filename the CCP looks to see if there are any other tags or names on the command line. If there are any, it constructs up to two more FCB's in the system parameter area starting at 005CH. The main program can look here to see if you specified anyfile and do something about it - like load it. Now Turbo Pascal may be doing its own processing of command lines and not proceesing tags in the same way as the CCP. I'd advise a good hack with Z8E to see what was going on - Ed.)

PUTTING 5.25 INCHES ON ARNOLD.

This small nugget of information is aimed at the 1 or 2 Amstrad users that might belong to the group. The information as it stands applies directly to the CPC464 and comes from an article in COMPUTING August 15 & 22, 1985. It may also also apply to the PCW 8256 but that is not guaranteed. The Amstrad's come with nice little 3 inch thingies, that certainly are not floppy but are a trifle expensive and can be difficult to obtain - disks I'm talking about. Also, there is rather more software on the 5.25 format. 5.25 inch drives are pretty cheap these days and it would be quite nice to just bolt one on and get the best of both worlds. Apparently it is quite a simple job since all the signals are present to conform to the Shugart SA400 Standard. The problem is that the interface does not present them in the required order. On the Shugart interface all the odd numbered pins are signal grounds and the even numbered pins carry the interface signals. The Amstrad reverses this sequence with odd numbered pins carrying the signals and evens the grounds. The Table gives the pin correspondence and function.

SA400 pins	Function	Amstrad interface pins
2	Not used	1
4	Head load	3
6	Drive 3 Select	5
8	Index Pulse	7
10	Drive 0 Select	9
12	Drive 1 Select	11
14	Drive 2 Select	13
16	Motor On	15
18	Step direction	17
20	Step Pulse	19
22	Write Data	21
24	Write Gate	23
26	Track 0 sense	25
28	Write protect	27
30	Read Data	29
32	Side 1 Select	31
34	Ready	33

In the case of the CPC464 the blue stripe on the cable coming from the interface identifies pin 34 NOT pin 1 as is usual.
A suitable cable can be made from a 34 way IDC connector to mate with the disk drive and 34 plug header which would normally be intended for soldering to PCB's. The latter mates with the connector already on the cable. These connectors may vary with the CPC 664 & PCW 8256.

Anon.II.

KERMIT - A file transfer protocol. - Geof Smith

This discussion of Kermit is based in part on the two articles published in Byte June/July 1984 and partly from my own experiences in setting up and using Kermit on a number of different computer systems.

What is Kermit ?

One thing that it is NOT is a little green frog although the documentation does claim that it is named after the Muppets Mega Star. The documentation also claims that it is the Celtic for 'free' which seems to make more sense since it is in the public domain. It is also NOT a high powered modem - communications program. There are many of these available both commercially and in the public domain, some of which employ the Kermit protocol and I hope to get some reviews on these from other members.

As the title says Kermit is a file transfer protocol - ie. a set of rules that enable a file to be transferred via a communication link from one computing environment to another with 100% fidelity. How the set of rules are implememted is irrelevant, as long as the rules are obeyed the file will get through unchanged. In the case of Kermit the protocol has been implemented in virtually every language and on machines ranging from a Beeb to a Cray (see end of article for complete list). There are other protocols in use, the most common being the X-MODEM protocol put forward by Ward Christensen. However MODEM-7 and successors were initially developed for communications between micros and so its use and universality of implementation are different to that of Kermit's. What I hope to do in this and the next couple of issues is to discuss the development of and thinking behind the Kermit protocol and detail the features and use of Kermit on the Model 4 & MSDOS systems.

Spawning of Kermit.

Kermit was developed initially at Columbia University in response to what is quite a common situation. The campus had a number of large main-frames in a central computing facility with many smaller systems scattered around laboratories and departments. With the explosive increase in the number of micros there was much demand for ways in which to exchange files between the central and peripheral computers. Additionally, with increased student interest in computing, usage of the mainframe on-line-storage was growing at an uncontrollable rate and the need to economically archive files was an urgent consideration. Given a reliable way to transfer files between mainframe and back, micros with floppy disks could prove an inexpensive way of achieving this. What was needed was a file transfer system that could work across the complete range of computers. The commercially available packages would only serve a strictly limited number of machines, and would be financially

crippling given the number of machines that needed to be supported. The solution was to develop their own system.

The Communication medium.

The only communication medium common to all computers, large and small, is the asynchronous serial telecommunication line, used for connecting terminals to computers. Certain aspects of this medium are almost universally followed -- voltages, signals, character encoding (ASCII, ANSI) and bit transmission sequence. Serial connections can be made in many ways: dedicated local lines (ie. null modem cables), leased telephone circuits, dialup connections. Dialup connections can be initiated manually from the home or office using an inexpensive modem, or automatically from one computer to another using a programmable dialing mechanism. The asynchronous serial line offers the ordinary user a high degree of convenience and control in establishing intersystem connections, at relatively low cost.

Once two computers are connected with a serial line, information can be transferred from one machine to the other, provided one side can be instructed to send the information and the other to receive it. But right away, several important factors come into play:

Potential Problems.

1. Noise -- It is rarely safe to assume that there will be no electrical interference on a line; any long or switched data communication line will have occasional interference, or noise, which typically results in garbled or extra characters. Noise corrupts data, perhaps in subtle ways that might not be noticed until it's too late.

2. Synchronization -- Data must not come in faster than the receiving machine can handle it. Although line speeds at the two ends of the connection may match, the receiving machine might not be able to process a steady stream of input at that speed. Its central processor may be too slow or too heavily loaded, or its buffers too full or too small. The typical symptom of a synchronization problem is lost data; most operating systems will simply discard incoming data they are not prepared to receive.

3. Line Faults -- A line may stop working for short periods because of a faulty connector, loss of power, or similar reason. On dialup or switched connections, such intermittent failures will cause the carrier to drop and the connection to be closed, but for any connection in which the carrier signal is not used, the symptom will be lost data. The serial telecommunication line provides no safeguards against such problems, and therefore must be regarded as an intrinsically unreliable medium.

Getting Reliable Communication over an Unreliable Medium

To determine whether data has been transmitted between two machines correctly and completely, the two machines can compare the data before and after transmission. A scheme that is commonly used for file transfer employs cooperating programs running simultaneously on each machine, communicating in a well-defined, concise language. The sending program divides outbound data into discrete pieces, adding to each piece special information describing the data for the receiving program. The result is called a "packet". The receiver separates the description from the data and determines whether they still match. If so, the packet is acknowledged and the transfer proceeds. If not, the packet is "negatively acknowledged" and the sender retransmits it; this procedure repeats for each packet until it is received correctly.

The process is called a communication protocol -- a set of rules for forming and transmitting packets, carried out by programs that embody those rules. Protocols vary in complexity; the programmers at Columbia chose a simple approach that could be realized in almost any language on almost any computer by a programmer of moderate skill, allowing the protocol to be adapted easily to new systems.

Accommodating Diverse Systems

Most systems agree how to communicate at the lowest levels - the EIA RS-232-C asynchronous communication line and the ASCII character set - but there is rarely agreement beyond that. To avoid a design that might lock out some kinds of systems, the important ways in which systems can differ must be considered.

Design Considerations

1. Mainframes vs Micros

The distinction made in the context of this project was that a micro is any single-user system in which the serial communication port is strictly an EXTERNAL device. A mainframe is any system which is "host" to multiple simultaneous terminal users, who login to "jobs", and the serial line is INTERNAL, ie. is the link between CPU and VDU/Console. In the latter case where the serial line is internal the appearance of certain characters on the line such as CTL/C may signify special events to the command processor or all input may be translated to upper case etc. The moral here is that care must be taken to disable special handling of a mainframe job's controlling terminal when it is to be a vehicle for interprocessor communication. But some systems simply do not allow certain of these features to be disabled, so file transfer protocols have to be designed around them.

2 Line Access

Line access can be either full or half duplex. If full duplex, transmission can occur in both directions at once. If half duplex, the two sides must take turns sending, each signaling the other when the line is free; data sent out of turn is discarded, or it can cause a break in synchronization. On mainframes, the host echoes characters typed at the terminal in full duplex, but not in half duplex. Naturally, echoing is undesirable during file transfer. Full duplex systems can usually accommodate half duplex communication, but not vice versa.

3. Buffering and Flow Control

Some systems cannot handle sustained bursts of input on a telecommunications line; the input buffer can fill up faster than it can be emptied, especially at high line speeds. Some systems attempt to buffer "typeahead" (unrequested input), while others discard it. Those that buffer typeahead may or may not provide a mechanism to test or clear the buffer. Systems may also try to regulate how fast characters come in using a flow control mechanism, either in the data stream (XON/XOFF) or in parallel to it (modem control signals), but no two systems can be assumed to honour the same conventions for flow control, or to do it at all. Even when flow control is being done, the control signals themselves are subject to noise corruption.

4. Character Interpretation

Systems can differ in how they interpret characters that arrive at the terminal port. A host can accept some characters as sent, ignore others, translate others, take special action on others. Communications front ends or multiplexers might swallow certain characters (typically DC1, DC3) for flow control, padding (NUL or DEL), or for transfer of control ("escape"). The characters that typically trigger special behavior are the ASCII control characters, 0-31 and 127. However, virtually all hosts and communication processors allow any "printable" character (ASCII 32-126) to reach an application program, even though the character maybe translated to a different encoding for internal use.

Some operating systems allow an application to input a character at a time, others delay passing the characters to the program until a "logical record" has been detected, usually a sequence of characters terminated by carriage return or linefeed. Some record oriented systems like IBM VM/370 discard the terminator, others keep it, and there are different ways of keeping it -- UNIX translates carriage return into linefeed; most DEC operating systems keep the carriage return but also add a linefeed

5. Timing Out

Hosts may or may not have the ability to "time out". When exchanging messages with another computer, it is desirable to be able to issue an input request without waiting forever should the incoming data be lost. A lost message could result in a protocol "deadlock" in which one system is waiting forever for the message while the other waits for a response. Some systems can set timer interrupts to allow escape from potentially blocking operations; others, including many microcomputers, can not do so. When timeouts are not possible, they may be simulated by sleep-and-test or loop-and-test operations, or deadlocked systems may be awakened by manual intervention.

6. File Organization

Some computers store all files in a uniform way, such as the linear stream of bytes that is a UNIX file. Other computers may have more complicated or diverse file organizations and access methods: record-oriented storage with its many variations, exemplified in IBM OS/360 or DEC RMS. Even simple microcomputers can present complications when files are treated as uniform data to be transferred; for instance under CP/M, the ends of binary and text files are determined differently. A major question in any operating system is whether a file is specified sufficiently by its contents and its name, or if additional external information is required to make the file valid. A simple generalized file transfer facility can be expected to transmit a file's name and contents, but not every conceivable attribute a file might possess.

Designers of expensive networks have gone to great lengths to pass file attributes along when transferring files between unlike systems. For instance, the DECnet Data Access Protocol supports 42 different "generic system capabilities" (like whether files can be preallocated, appended to, accessed randomly, etc), 8 data types (ASCII, EBCDIC, executable, etc), 4 organizations (sequential, relative, indexed, hashed), 5 record formats (fixed, variable, etc), 8 record attributes (for format control), 14 file allocation attributes (byte size, record size, blocksize, etc), 28 access options (supersede, update, append, rewind, etc), 26 device characteristics (terminal, directory structured, shared, spooled, etc), various access options (new, old, rename, password, etc),in addition to the better known file attributes like name, creation date, protection code, and so on. All this was deemed necessary even when the designers had only a small number of machines to worry about, all from a single vendor !!.

7. Binary Files versus Parity

Each ASCII character is represented by a string of 7 bits. Printable ASCII files can be transmitted in a straightforward fashion, because ASCII transmission is designed for them: a serial stream of 8-bit characters, 7 bits for data and 1 for

parity, framed by start and stop bits for the benefit of the hardware. The parity bit is added as a check on the integrity of a character; some systems always transmit parity, others insist on parity for incoming characters, still others ignore the parity bit for communication purposes and pass it along to the software, while still others discard it altogether. In addition, communication front ends or common carriers might usurp the parity bit, regardless of what the system itself may do.

Computer file systems generally store an ASCII file as a sequence of either 7-bit or 8-bit bytes. 8-bit bytes are more common, in which the 8th bit of each byte is superfluous. As well as files composed of ASCII characters, computers also have "binary" files, in which every bit is meaningful; examples include executable "core images" of programs, numbers stored in "internal format", databases with imbedded pointers. Such binary data must be mapped to ASCII characters for transmission over serial lines.

8. Software

Finally, systems differ in the application software they have. In particular, no system can be assumed to have a particular programming language. Even widespread languages like FORTRAN and BASIC may be lacking from some computers, either because they have not been implemented, or because they are proprietary and have not been purchased. Even when two different systems support the same language, it is unrealistic to expect the two implementations of the language to be totally compatible. A general purpose file transfer protocol should not be geared towards the features any particular language.

The above discussion should have given you some insight into the problems that face us in the world of computer communications and perhaps you now understand why we can't always just plug them together for a happy and fruitful union!. Next month I will review the protocol itself but in case any of you want to try out KERMIT, on receipt of a disk and return postage I can let you have a number of implementations that will work on Model 1/3 (tested under NEWDOS/TRSDOS) Model 4 (CPM) and PC-Clones (PCDOS).

Geof Smith.

Space saving in filing routines revisited - M.C. Matthews

I read with interest Mr. Baust's comment on my filing routines, and would first say that he obviously missed my second instalment which appeared in the same newsletter. This covers his point about the concatenation of fixed length items.

His main point seems to be that he cannot see where the space saving comes in. Most of my filing is in the form of arrays of one sort or another, and to take a simple example, I have an array which holds information in the form of integers. Each integer has four pieces of information packed into it, and there are 300 of them.

Filing these is staightforward. We can file them in blocks of 127 in each sector, using A as the variable and A$ as the field string. If we do this we have the dimension A$(127), which takes 393 bytes, and we are left with these strings in the string space. If we file in the method I suggest, that is

```
FIELD 1, 255 as GF$
FOR I=1 TO 127
    LSET GF$=LEFT$(GF$,(I-1)*2)+MKI$(A)
NEXT I
```

we only occupy 25 bytes of string space. This is determined by practical experiment using a simple program as follows:-

Program	Result	Difference
10 OPEN"R",1,"TEST":0"		
20 PRINT FRE(Z$)	27918	
30 FIELD 1,255 AS A$:PRINT FRE(Z$)	27911	7
40 FIELD 1,100 AS B$,100 AS C$,55 AS D$:PRINT FRE(Z$)	27890	21
50 FIELD 1,255 AS GF$:PRINT FRE(Z$)	27883	7
60 DIM A(127):PRINT FRE(Z$)	27362	521
70 FOR I=1 TO 127:LSET GF$= LEFT$(GF$,(I-1)*2)+MKI$(A(I)) NEXT:PRINT FRE(Z$)	27346	18

If we change lines 50 to 70 we get the following result:-

Program	Result	Difference
At line 30	27869	
50 X=0:DIM A$(127):FOR I=1TO 127: FIELD 1,(X)AS T$,2AS A$(I):X=X+2:NEXT :PRINT FRE(Z$)	27453	416
60 DIM A(127):PRINT FRE(Z$)	26932	521
70 FOR I=1TO 127:LSET A$(I)=MKI$(A(I)) :NEXT:PRINT FRE(0)	26932	0

Since the second method is 42 bytes LONGER than the first we can see that the free memory is in fact 414 bytes greater just on this one routine. Looking at the early part of the program, line 40 shows that each subdivision of the field takes an extra 7 bytes of memory. The real saving is in the fact that it is no

longer necessary to dimension field strings for arrays, and removing these from one of my programs reduced its memory demands by nearly 6kb.

I have in fact abandoned the use of GF$="" at the end of each routine, but since fielding is usually just a matter of putting FIELD 1,255 AS GF$, ALL my random filing routines re-field when called. This has the further advantage of reducing errors resulting from calling a routine with an incorrect field.

M.C. Matthews, Dorset.

While on the subject of FIELDing things, I recently wrote a quick and dirty BASIC program to split a large (360 sectors) text file into more manageable pieces.

```
10 CLEAR 1000 : OPEN "R",1,"FILE/TXT:1"
15 FIELD 1,255 AS A$ : FIELD 2,255 AS B$
20 I=0 : RI = 0 : RO = 0
30 TAG$ = CHR$(65+I)
40 NM$ = "FILE"+TAG$+"/TXT:1"
45 OPEN "R",2,NM$
50 RI = RI + 1 : RO = RO + 1
60 GET 1,RI : LSET B$=A$ : PUT 2,RO
70 IF RI = 360 THEN CLOSE : END
80 CLOSE 2 : RO = 0 : GOTO 30
```

Quite crude but should work I thought, although line 15 bothered me. However, the manual does say (p 7-47) that the buffer is 255 bytes and of course Basic will not let you have a string of greater length. It does appear to work, but you only get the first 255 characters of your file when you load it into Scripsit. The answer is of course that the buffer is NOT 255 but 256 bytes (since 0 counts as 1 - logical (!) isn't it?) and the 256th byte gets zapped to zero. Lines 15 AND 60 have to be

```
15 FIELD 1,128 AS A$,128 AS AA$ : FIELD 2,128 AS B$,128 AS BB$
60 GET 1,RI : LSET B$=A$ : LSET BB$=AA$ : PUT 2,RO
```

Most habitual BASIC users will probably have discovered this eons ago but it may save one or two casual users the odd hour of hair pulling - Ed.