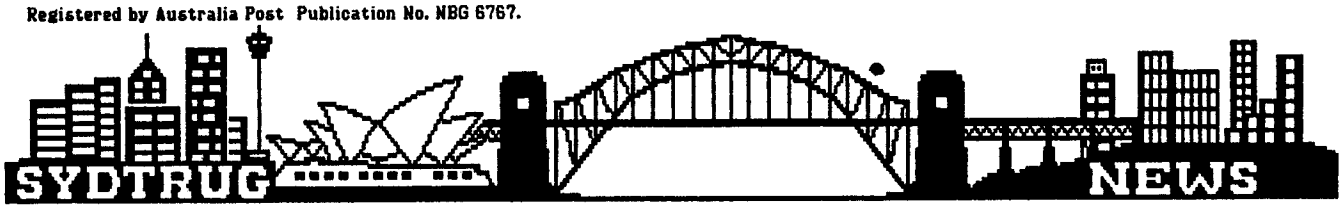
Registered by Australia Post Publication No. NBG 6767.



NEWSLETTER OF SYDTRUG INC.
SYDNEY TRS-80/MS-DOS USERS' GROUP
P.O. BOX 223, BANKSTOWN 2200

Volume 10 Issue 10 AUGUST 1990

CONTENTS

MEETING ARRANGEMENTS

Meetings will be held on Saturday afternoons commencing at 1:00 P.M. as follows:-

11th and 25th of August at the 1st Sefton Scout Hall
2 Waldron Road, SEFTON

WHO'S WHO

Public Officer	T. FOLEY	389-6157
President	Denis J. PAGETT	772-4848
Vice President	Errol ROSSER	796-7646
Secretary	Bruce RAMSAY	580-2217
Treasurer	Gordon SYMONDS	744-1901
CLUB-80 Sysop	Michael COOPER	331-7136
TRUG-86 Sysop	Errol ROSSER	796-7646
Hardware Co-ordinator	Errol ROSSER	796-7646
Newsletter Editor	John MERCER	579-2915

BANKCARD and MASTERCARD

We have the facility to charge your membership fees, or renewal fees to either MASTERCARD or BANKCARD. Additionally, purchases made on your behalf by the club may also be charged to your credit card. If you wish to use this service, please quote your card number, type of card, expiry date of card, and SIGN your request.

Newsletter Closing Dates

Hard Copy only — 4th August 1990 —

On Disk or
Via Bulletin Board — 11th August 1990 —

Membership and BBS Fees were due on July 1st
MEMBERSHIP: $25.00, Bulletin Boards: $15.00
For More details see page 132

NOTE: There are now TWO SYDTRUG Bulletin Boards

TRUG-86, the MS-DOS Bulletin Board, is now functioning on (02) 790-5681, and should cater for all data formats as with CLUB-80 described below. It is currently in the developmental stage, so please bear with us if it is not quite as easy as you might have hoped. For those who wish to use PROCOMM, there are some patches available on the TRUG-86 board to enable it to function with TRUG-86. All members who have paid for access to CLUB-80, should automatically have access to TRUG-86. Initially, your Password is your Membership Number, so it would be a good idea to log on and change your Password to one which only you know.

CLUB-80, the TRS-80 Bulletin Board, operates for members, seven days a week twenty-four (24) hours a day on (02) 332-2494. We use a NetComm 1234sa "intelligent" MODEM, and the following data formats are available :-

CCITT V21 (300/300), V22 (1200/1200),
V23 (1200/75) and V22 bis (2400/2400).
BELL 103 (300 FULL Duplex), BELL 212 (1200/1200)
2400 (2400 FULL Duplex)
All formats utilize 8 DATA Bits, 1 STOP Bit and NO Parity

You should set your Modem and/or software for "ORIGINATE" except for V23 (1200/75) which should be set for VIATEL or 1200 Receive/75 Transmit

Limited access is granted for visitors. Articles for publication should be left in the News Room of CLUB-80 for collection by the Editor. Please leave ASCII files.

DISCLAIMER

No Patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this publication, neither SYDTRUG Inc. nor its appointed office bearers assume no responsibility for errors or omissions. Neither is any liability assumed for damages arising from the use of any information contained herein. Any opinions expressed are those of the author concerned, and not necessarily those of the Group or its committee.

For Sale

TRS-80 Model III with four (4) internal drives (2 double sided 40 and 2 double sided 80), Amber Screen, RS-232 board, 15 Meg external hard drive.

Nearly all MISOSYS software, Profile III+, Lescript V2.0 and lots more.

Many items of hardware.

Offers invited - Phone (02) 523-2453 between 6 pm and 9 pm, or write:

Ian Lange
P.O. Box 326
DAPTO NSW 2530

==========

THE C FORUM

CHARACTER AND BIT MANIPULATION

by Ivan KENNEDY

This is to be the final but one article in the tutorial series of the C Forum. Not before time, I can hear some of you saying. But just for fun, and to see if anyone is actually reading the articles, I have included a competition in this month's C Forum with a free set of The C Tutor disks as the prize. Of course the forum itself could continue indefinitely with articles from time to time, but not as a regular series. There will certainly be an opportunity for other club members to have a go.

Throughout this series I've often pointed out that Gordon Dodrill, of Coronado Enterprises, is the real brains behind it. His "C Tutorial", a Shareware offering, contains nearly all the C programs that have been covered as well as much of the text descriptions. His concept was that each user could find his/her own compiler for the particular computer involved and use the tutorial programs interactively to learn the language. This approach can be successful only because of C's portability. If you really want to learn C it would be worth while getting his two disks from the clubs's PD library or by writing to him (Coronado Enterprises, 12501 Coronado Ave NE, Albuquerque, NM 87122, U.S.A) with $12US plus $4 handling fee. MasterCard or Visa cards are accepted.

Gordon Dodrill also offers Shareware language tutors for Modula-2 and Turbo Pascal similar in concept to the C Tutorial for the same price, as well as laser-printed tutorial books for Ada, C, C++, Modula-2 and Turbo Pascal for $61.95 ea. including airmail. He certainly has provided both hobbyists and serious programmers with excellent means of learning a number of computer languages. The TRS-80 (ALCOR) PASCAL compiler is still available from DiskCount Data in the U.S.A. (with an excellent manual very similar in style to ALCOR C's) and no doubt those interested in trying the stricter, less cryptic, syntax of PASCAL on the TRS-80 Model 4 would find Coronado's PASCAL tutor ideal to learn with.

I don't know of any Modula-2 or Ada compilers available to the TRS-80 world. MS-DOS users would not have much have trouble finding one. C++ is an upgrade of C and any C program is automatically a C++ program, provided it doesn't contain any of the additional keywords of C++ as variables. According to Gordon Dodrill, "C++ adds object oriented programming techniques into the C language" and its where you go when you need more power from C.

In case you have wondered - Gordon does not object to his C Tutor being used in the way I have. Indeed, he has even given me written permission to use it within the SYDTRUG group environment. He says that the C Tutor has been ported to many computers including even a Cray-2 advanced mainframe, and ran well everywhere. I would say he is another one of those generous characters who get more out of actual achievements by way of helping others than by chasing the 'holy dollar'. More thanks to him.

Well, let's launch into the penultimate tutorial. On this occasion, we're going to take a look at some ways of changing the case (upper or lower) of characters in a file, of analysing a file for different classes of characters, of performing bitwise operations using logical comparisons and at the shift instructions. These items have been left behind as we have been going through C's functions.

UPPER AND LOWER CASE

```
#include "STDIO.H"
#include "ctypE.h"  /* Your compiler may not need this */
                    /* ALCORC doesn't */
main()
{
FILE *fp;
char line[80], filename[24];
char *c;

  printf("Enter filename -> ");
  scanf("%s",filename);
  fp = fopen(filename,"r");

  do {
    c = fgets(line,80,fp);  /* get a line of text */
    if (c != NULL) {
      mix_up_the_chars(line); /* call the function */
    }
  } while (c != NULL);

  fclose(fp);
}


mix_up_the_line(line)  /* this function turns all upper case
                          characters into lower case, and all
                          lower case to upper case. It ignores
                          all other characters.          */
char line[];
{
int index;

  for (index = 0;line[index] != 0;index++) {
    if (isupper(line[index]))    /* 1 if upper case */
      line[index] = tolower(line[index]);
    else {
      if (islower(line[index]))  /* 1 if lower case */
        line[index] = toupper(line[index]);
    }
  }
  printf("%s",line);
}
```

The program UPLOW.C(/C) is displayed as an example of a program that does lots of character manipulation. More specifically, it changes the case of alphabetic characters around. It illustrates the use of four functions that have to do with case. It should be no problem for you to study this program on your own and understand how it works. The four functions on display in this program are all within the user written function, "mix_up_the_line". Compile and run the program with the file of your choice. The four functions are;

isupper(); Is the character upper case?
islower(); Is the character lower case?
toupper(); Make the character upper case.
tolower(); Make the character lower case.

Whatever filename you give, the program writes to the screen having changed all upper case letters to lower case and vice versa. This works fine with ALCOR C and using the ZSHELL piping technique described below, you could automatically write the screen output to a file on disk. Notice the function mix_up_the_line and the calling code (mix_up_the_chars) are not the same but it still runs perfectly. How can this be? (A free set of two The C Tutor disks to whoever writes (c/- the Newsletter Editor, John Mercer) first with the answer).

Continued on page 124

For Sale

Tandy DMP 105 Printer - As New

Asking Price - $180

Leon Aroustian - Bus. Hrs. - 550-8284
Evenings - 607-4637

==========

CLASSIFICATION OF CHARACTERS

Load and display the next program, CHARCLAS.C(/C) for an example of character counting.

```
#include "stdio.h"
#include "ctype.h"  /* Your compiler may not need this */
                    /* ALCORC doesn't */
main()
{
FILE *fp;
char line[80], filename[24];
char *c;

  printf("Enter filename -> ");
  scanf("%s",filename);
  fp = fopen(filename,"r");

  do {
    c = fgets(line,80,fp);  /* get a line of text */
    if (c != NULL) {
      count_the_data(line);
    }
  } while (c != NULL);

  fclose(fp);
}

count_the_data(line)
char line[];
{
int whites, chars, digits;
int index;

  whites = chars = digits = 0;

  for (index = 0;line[index] != 0;index++) {
    if (isalpha(line[index]))  /* 1 if line[] is alphabetic  */
       chars++;
    if (isdigit(line[index]))  /* 1 if line[] is a digit     */
       digits++;
    if (isspace(line[index]))  /* 1 if line[] is blank, tab, */
       whites++;               /*         or newline         */
  }  /* end of counting loop */

  printf("%3d%3d%3d %s",whites,chars,digits,line);
}
```

We have repeatedly used the back slash \n character representing a new line. There are several others that are commonly used, so they are defined in the following table;

\n	Newline
\t	Tab
\b	Backspace
\"	Double quote
\\	Backslash
\0	NULL (zero)

By preceding each of the above characters with the back slash character, the character can be included in a line of text for display, or printing. In the same way that it is perfectly all right to use the letter "n" in a line of text as a part of someone's name, and as an end-of-line, the other characters can be used as parts of text or for their particular functions.

The program on your screen uses the functions that can determine the class of a character, and counts the characters in each class. The number of each class is displayed along with the line itself. The three functions are as follows;

isalpha();	Is the character alphabetic?
isdigit();	Is the character a numeral?
isspace();	Is the character any of, \n, \t, or blank?

This program should be simple for you to find your way through so no explanation will be given. Compile and run this program with any file you choose. Here is the output (imported with VED) I obtained with a simple JCL file that I use to set up subDISKs (diskDISKs) for ALLWRITE and ELECTRIC WEBSTER and to load PROWAM.

```
                                                        00:12:36
LS-DOS Ready
RUNC CHARCLAS

Enter filename -> AL/JCL
  2 11  0 TIME (CLOCK=ON)
  3  4  2 SWAP :5 :6
  3  4  2 SWAP :6 :7
  3  4  2 SWAP :4 :5
  3  4  2 SWAP :5 :6
  3  4  2 SD6 4 AL
  3  4  2 SD6 5 EW
  2  7  1 PROWAM (B=2)

PROGRAM TERMINATED AT #90DC
STACK USED =  455 OF 11886  HEAP USED =  1376 OF 11884
LS-DOS Ready
```

As the program directs, a table is written to the screen of each line in the file giving the number of spaces (whites), the number of characters (chars), the number of digits and the line itself.

THE LOGICAL FUNCTIONS

Load and display the program BITOPS.C(/C).

```
main()
{
char mask;
char number[6];
char and,or,xor,inv,index;

  number[0] = 0X00;
  number[1] = 0X11;
  number[2] = 0X22;
  number[3] = 0X44;
  number[4] = 0X88;
  number[5] = 0XFF;

  printf(" nmbr  mask  and   or   xor   inv\n");
  mask = 0X0F;
  for (index = 0;index <= 5;index++) {
    and = mask & number[index];
    or = mask | number[index];
    xor = mask ^ number[index];
    inv = ~number[index];
    printf("%5x %5x %5x %5x %5x %5x\n",number[index],
         mask,and,or,xor,inv);
  }

  printf("\n");
  mask = 0X22;
  for (index = 0;index <= 5;index++) {
    and = mask & number[index];
    or = mask | number[index];
    xor = mask ^ number[index];
    inv = ~number[index];
    printf("%5x %5x %5x %5x %5x %5x\n",number[index],
         mask,and,or,xor,inv);
  }
}
```

The functions in this group of functions are used to do bitwise operations, meaning that the operations are performed on the bits as though they were individual bits. No carry from bit to bit is performed as would be done with a binary addition. Even though the operations are performed on a single bit as is, an entire byte or integer variable can be operated on in one instruction. The operators and the operations they perform are given in the following table;

&	Logical AND, if both bits are 1, the result is 1.
|	Logical OR, if either bit is one, the result is 1.
^	Logical XOR, (exclusive OR),if one and only one bit is 1, the result is 1.
~	Logical invert,if the bit is 1, the result is 0, and if the bit is 0, the result is 1.

The example program uses several fields that are combined in each of the ways given above. The data is in hexadecimal format. It will be assumed that you already know hexadecimal format if you need to use these operations. If you don't, you will need to study it on your own. Teaching the hexadecimal format of numbers is beyond the scope of this tutorial.

Run the program and observe the output.

```
READY                                                    22:26:42
RUNC BITOPS

 nmbr mask  and  or  xor  inv
    0    F    0   F    F   FF
   11    F    1  1F   1E   EE
   22    F    2  2F   2D   DD
   44    F    4  4F   4B   BB
   88    F    8  8F   87   77
   FF    F    F  FF   F0    0

    0   22    0  22   22   FF
   11   22    0  33   33   EE
   22   22   22  22    0   DD
   44   22    0  66   66   BB
   88   22    0  AA   AA   77
   FF   22   22  FF   DD    0

PROGRAM TERMINATED AT #90DC
STACK USED =  340 OF 11767  HEAP USED =  1032 OF 11764
Hit (BREAK) to enter DOS command, any other key for SHELL
directory.
```

Maybe you can make more sense of this than I can. With more time?

THE SHIFT INSTRUCTIONS

The last two operations to be covered in this article are the left shift and the right shift instructions. Load the example program SHIFTER.C(/C) for an example using these two instructions. The two operations use the following operators;

```
<< nLeft shift n places.
>> nRight shift n places.
```

```
main()
{
int small, big, index, count;

  printf("      shift left      shift right\n\n");
  small = 1;
  big = 0x4000;
  for(index = 0;index < 17;index++) {
    printf("%8d %8x %8d %8x\n",small,small,big,big);
    small = small << 1;
    big = big >> 1;
  }

  printf("\n");
  count = 2;
  small = 1;
  big = 0x4000;
  for(index = 0;index < 9;index++) {
    printf("%8d %8x %8d %8x\n",small,small,big,big);
    small = small << count;
    big = big >> count;
  }
}
```

Once again the operations are carried out and displayed using the hexadecimal format. The program should be simple for you to understand on your own, there is no tricky code.

When compiled and run, this output is obtained with RUNC.

```
    shift left     shift right

         1        1   16384    4000
         2        2    8192    2000
         4        4    4096    1000
         8        8    2048     800
         6       10    1024     400
         2       20     512     200
        64       40     256     100
       128       80     128      80
       256      100      64      40
       512      200      32      20
      1024      400      16      10
      2048      800       8       8
      4096     1000       4       4
      8192     2000       2       2
     16384     4000       1       1
    -32768     8000       0       0
         0        0       0       0

         1        1   16384    4000
         4        4    4096    1000
        16       10    1024     400
        64       40     256     100
       256      100      64      40
      1024      400      16      10
      4096     1000       4       4
     16384     4000       1       1
         0        0       0       0

PROGRAM TERMINATED AT #90DC
STACK USED =  395 OF 11112  HEAP USED =  1032 OF 11110
```

That just about takes care of these housekeeping operations. Although not exactly gripping, they do form a necessary part of the C language.

Getting this final screen output into the article was easily done, even though it's more than one screen full. I did it by using ZSHELL, MISOSYS' utility, by which one one can 'pipe' the screen print into an ALLWRITE file (SHIFTER/NWS), using the command:

```
RUNC SHIFTER ! AL SHIFTER/NWS
```

Once the ZSHELL command is given, a linked sequence follows in which output from RUNC SHIFTER is placed in a temporary /PI file, ALLWRITE is loaded and the data in the temporary file is 'piped' into an ALLWRITE text file, SHIFTER/NWS. You can then save the new file. When the program output is not more than one screen full, the output data stays on the screen and one can simply use PROWAM's video editor (VED) to capture the screen and place it in another file. But this can't be done if there is more than a screen full, as the early lines then scroll off the screen. ZSHELL's piping function (it has several others) solves the problem neatly by capturing all the output no matter how many lines are printed to the screen. Notice that the command lines to run the program don't get into the final product, unlike using VED as above. This is just another example of the power of LS-DOS and one of the vast number of utilities that are available.

Finding such solutions is part of the fun of using LS-DOS. Even after three and a half years, I'm still enjoying discovering new and better ways to use these Model 4/4Ps and wild horses couldn't drag them away from me. It's my impression that all the fancy features of the newer MS-DOS (and Macintosh) machines, while very fine for users, tend to isolate one from the nuts and bolts of using a computer. The result seems to be less interest in putting together a better system to work with and less pride and pleasure from telling other people about it. For example, members who have crossed the bridge to MS-DOS as their main environment seem to contribute less to "SYDTRUG News" than formerly.

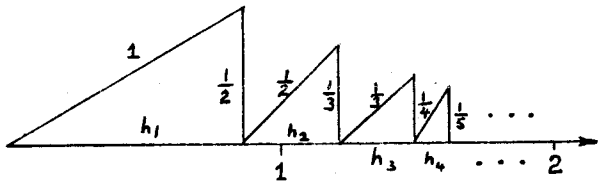
Maybe I'm plain wrong about that and they've simply been busy working on a vast number of MS-DOS projects, articles about which are about to descend on the editor like a tidal wave.

IN DEFENCE OF DOUBLE PRECISION COMPUTING

by Edward SIKE - (02) 533 3130

There is much to be said in favour of Double-Precision Computing. We shall see that its usefulness is beyond dispute in any detailed, high-precision study of numerical phenomena. - By sharing my more interesting experiences in this area, I hope to awaken some curiosity among members of SYDTRUG, in what really is a very important and often very challenging subject ...

Originally, the idea for the remarkable problem to be described here, occurred to me a few years back. The present results emerged rather slowly, mainly due to the lack of a suitable computing facility. To begin with, the successive terms of the numerical sequence we wish to investigate are represented by the "horizontal" sides of smaller and smaller Pythagorean Triangles, as shown in the sketch below. There, one can clearly see, that the successive triangles are neatly arranged along the so-called Number Line, 0 ... 1 ... 2 ... 3 ... Also, it can be seen that the problem is really of a GEOMETRICAL-NUMERICAL nature.



/1/

One by one, the relative sizes of the mentioned Horizontal Terms,

$$h_1 < h_2 < h_3 < h_4 < \ldots \text{ ad inf.,} \qquad /2/$$

can be calculated - by hand, or preferably with the aid of a computer. For instance the fourth successive term ("h" four squared),

$$h * h = (1/4)*(1/4) - (1/5)*(1/5) \qquad /3/$$

$$= \frac{5 * 5 - 4 * 4}{(4 * 4) * (5 * 5)} = (.15) * (.15)$$

In this way, ".15" is a rather exceptional result, because this particular result raises the question of the existence of some other ... "RATIONAL" Triangles of this kind. That is to say, triangles associated with the fundamentally important sequence of the Reciprocals of Natural Numbers, i.e. Natural Fractions :

$$(1/1) < (1/2) < (1/3) < \ldots \text{ ad inf.} \qquad /4/$$

1. THE INITIAL MINI-PROGRAM

The other calculations can be much more complicated. Consequently, for reasons of speed and (more importantly) freedom from errors, such calculations are best done on a suitable computer. For that purpose, the following short program, in BASIC, is used to obtain some preliminary results:

```
10 REM * * * Successive Natural Triangles (SNT).        /5/
12 REM
22 DEFDBL A-Z : CLS
100 REM (the first loop)
111 M=M+1 : N=M+1
133 GOSUB 1100
155 PRINT M;L;S
199 GOTO 100
1000 REM
1100 REM * * * Double Precision  SQUARE ROOT SUBROUTINE
1111 W=(1/(M*M))-(1/(N*N))
1122 L=SQR(W) : L=(L+W/L)/2 : L=(L+W/L)/2
1133 S=S+L
1155 RETURN
9999 END
```

The generous line-numbering arrangement will facilitate future extensions of this short Mini-Program (to be identified by the Reference Number /5/). Incidentally, this program works also in GWbasic (i.e. on IBM Compatible P.C.).

The successive lengths of the adjacent "horizontal" sides (along the mentioned Number Line) are now represented by the variable "L". At the same time, the successive Sums are represented by the variable "S" (also accompanied by a specific subscript). Thus, step by step, as the variable "M" increases, (becoming equal to 1, 2, 3, ...), we obtain the following, slowly increasing Sums:

$$S_1 = L_1 \qquad /6/$$
$$S_2 = L_1 + L_2$$
$$S_3 = L_1 + L_2 + L_3$$
$$\ldots \quad \ldots \quad \ldots \text{ ad inf.}$$

While running the above-listed Mini-Program /5/, we soon notice that the slowly increasing Sums (displayed on the screen) TEND TO APPROACH a certain, apparently finite Ultimate Sum, $(S)_{ult}$. In fact, our principal objective will be, to compute this finite Ultimate Sum, as accurately as possible.

Surprisingly, such a task is not as easy as it may seem at first sight - even for a reasonably powerful computer. Indeed, it took a long time before the author of this article made some really substantial progress with this Geometrical-Numerical Problem, after buying a TRS-80 Computer (Model 1). As it turned out, even with this relatively powerful machine (though rather slow), the outlined problem still remained a very hard nut to crack, so to speak.

As we shall soon see, in the end the DOUBLE-PRECISION mode of operation was quite essential to the eventual success; and that means calculations to the theoretical precision of 16 decimal figures. Just the same, after exploiting to the full, this easy-to-use Double Precision mode of computation, I sometimes wished for a faster machine.

By the way, the procedure used in Line 1111 is largely due to Pythagoras. Most of the other lines are self-explanatory. From now on, it will be convenient to refer to our Ultimate Series (which, in a way is a ... "Naturally Infinite" Series), simply as the SNT-Series. In other words, a N.I.Series, derived from the described Successive Natural Triangles (S.N.T). The expression NATURALLY INFINITE is used deliberately, to emphasize the fact, that the Natural Numbers,

$$1 < 2 < 3 < 4 < \ldots \text{ ad inf.,} \qquad /7/$$

are the Reciprocals of Natural Fractions (mentioned earlier). Thus, the above Natural Numbers are inseparably involved in this Geometrical-Numerical Problem.

With some extra care, observe that there is what might be described as the

PERFECT one-to-one correspondence, /8/

between the successive terms of the Ultimate Series, $(S)_{ult}$, and the above sequence of Natural Numbers. We are therefore justified to regard these two, and any similar infinite structures, as the "NATURALLY INFINITE" numerical structures (in short "N.I.").

Whenever this will be desirable, the mentioned one-to-one correspondence will be indicated by the convenient abbreviation "ad inf.", already used on one or two occasions. To gain some direct feel for what the suggested Mini-Program /5/ can do right now (and later on), the reader is strongly urged to run the suggested short program on his, or her own computer, and the sooner the better. After all, there is no substitute for first-hand, direct experience ...

Be it as it may, the successive numerical values of our variables, "M", "L", and "S" are as follows:

```
 1  .8660254037844386      .8660254037844386          /9/
 2  .372677996249965       1.238703400034404
 3  .2204792759220492      1.459182675956453
 4  .15                    1.609182675956453
 5  .1105541596785133      1.719736835634966
 6  8.584645893961879D-02  1.805583294574585
 7  6.916041689656102D-02  1.874743711471146
 8  .0572653559113564      1.932009067382502
 9  4.843221048378526D-02  1.980441277866288
10  4.165977904505309D-02  2.022101056911341
```

To obtain a nice print-out like this, you would of course require the following line: 144 LPRINT M ; L ; TAB (35) S. Just the same, whether you have or have not a printer, the very same results can be observed on the screen of your monitor (see line 155).

2. VARIOUS EXTENSIONS OF OUR MINI-PROGRAM

In calculations like these, one should always be careful and not allow accidental errors to occur. The following ... "reverse" Subroutine can be used for checking the correctness of our preliminary results:

```
1000 REM * * * EXAMPLE OF A CHECK SUB-ROUTINE        /10/
1002 REM       FIRSTLY, INCLUDE THE LINES
1004 REM       140 GOSUB 1011
1006 REM       155 PRINT M ; L ; , 1/T
1008 REM
1011 L = S - S8 : S8 = S
1022 V = L * L + 1/( N * N )
1033 T = SQR(V) : T = (T+V/T)/2 : T = (T+V/T)/2
1055 RETURN
```

Later on, similar Check Subroutines can be devised to check other results. As we continue our present computations (based on the Mini Program /5/), we soon notice that the variable "S" increases at a slower and slower rate. This can be seen more clearly in the following examples:

$$(S)_{101} = 2.594\ 9793\ 5820\ 9850 \qquad /11/$$
$$(S)_{102} = 2.596\ 3421\ 8145\ 9734$$
$$(S)_{103} = 2.597\ 6853\ 0081\ 9391$$

Much further on, along the way towards the mentioned Ultimate Sum, $(S)_{ult}$, this SLOWER AND SLOWER progress is even more evident:

$$(S)_{1001} = 2.785\ 6824\ 6773\ 1005 \qquad /12/$$
$$(S)_{1002} = 2.785\ 7270\ 2191\ 9094$$
$$(S)_{1003} = 2.785\ 7715\ 0952\ 5567$$

Note also, that there is no change in the leading four decimal figures in this example. Thus, we realise that the rate of change tends to become very slow indeed. Therefore, probably only the first decimal figure (i.e. "2") may be assumed as a correct approximation of the repeatedly mentioned Ultimate Sum. As one would expect, this snail-like progress is not improved when we eventually arrive, after about two hours of continuous computing, at the next example of successive sums:

$$(S)_{10001} = 2.846\ 7544\ 3647\ 1622 \qquad /13/$$
$$(S)_{10002} = 2.846\ 7558\ 5015\ 5024$$
$$(S)_{10003} = 2.846\ 7572\ 6362\ 6452$$

At this stage, it is fairly obvious that a conventional approach to the problem in hand will not get us anywhere near the intended target. We must therefore find some other, more effective strategy. - For it is fairly obvious by now, that brute computing power alone is almost useless in this particular case of number crunching. Fortunately, there are certain unusual ways and means to overcome this difficulty of a seemingly insuperable kind. The more enterprising reader is now provided with an opportunity to confirm the preceding results and to give some additional thought to this kind of a problem; possibly, also to some other equally challenging GEOMETRICAL-NUMERICAL PROBLEMS ...

As early as possible, the author wishes to thank Graham O'Connor for his very substantial help with the preparation of the present article (entitled "In Defence of Double Precision Computing"). The intention is to continue this article with a second installment, third installment, etc.

==========

For Sale

2 x Grey Case CoCo plus double diskdrive with some software

Contact: Rhonda HULL (069) 22-6517

==========

Worth Repeating

Being a woman is a terribly difficult task, since it consists principally of dealing with men. -- Joseph Conrad

I not only use all the brains I have, but all that I can borrow. -- Woodrow Wilson

Man is always ready to die for an idea, provided the idea is not to clear to him. -- Paul Eldridge

==========

Exchange Newsletters Listing

Listed below are newsletters available for borrowing from our library at Sefton as at 14th July, 1990 :-

"Adelaide Micro Users News"
1989 - Feb, Mar, Apr, Sep, Oct, Nov.
1990 - Feb, Mar, Apr, May, Jun, Jul.

"Bits & Bytes"
(TRS-80 System 80 Computer Users Group Inc.)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov.
1990 - Jan, Feb, Mar, Apr, May, Jun.

Canberra Micro 80 Users Group Inc.
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
1990 - Feb, Mar, Apr, May, Jun, Jul.

"Computer News 80"
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul.

"Hawtug News"
(Hawaii Tandy Users Group)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
1990 - Feb, Mar, Apr, May, Jul.

"Interface"
(The San Gabriel Valley Tandy User's Group)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
1990 - Jan, Mar, Apr, May, Jun, Jul.

"LLIST"
(Calgary Color Computer Club)
1989 - Jan, Feb, Mar, Apr, May, Jun, Sep, Oct, Nov, Dec.
1990 - Jan, Mar, Apr, May, Jun.

"LPRINT"
(The Micro 80 Computer Club of Ottawa)
1989 - Apr, Jul, Nov, Dec.
1990 - Feb, Mar, Apr, May.

"NATGUG News"
(National Amstrad Tandy and General User Group)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
1990 - Jan, Feb, Mar, Apr, May, Jun.

National Capital Tandy Computer Users Group
1989 - Jan, Feb, Mar, Apr, May, Jul/Aug, Sep, Oct, Nov.
1990 - Jan, Feb, Mar, Apr, May.

"SVCS Newsletter" or "SVCS Journal"
(Silicon Valley Computer Society)
1989 - Jan, Feb, Mar, May, Jun, Jul

"TCTUG"
(Twin Cities Tandy Users Group)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Sep.

"TC/PC"
(Twin Cities PC User Group)
1989 - Oct, Nov, Dec.

"Thuggery"
(The Hobart Users Group Inc.)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Oct, Nov, Dec.
1990 - Jan, Feb, Mar, Apr, May, Jun, Jul.

"TRSTimes"
1989 - Jan/Feb, Mar/Apr, May/Jun, Jul/Aug, Sep/Oct, Nov/Dec.

"Voice of the '80" or "The Voice of FCUG"
(The Fairfield County Computer Users Group) 1989 - Jan, Feb, Mar, Apr, Jun, Jul, Aug, Oct, Nov, Dec.
1990 - Feb, Mar, Apr, May, Jun, Jul.

"WNYTUG News"
(West New York Tandy Users Group)
1989 - Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec.
1990 - Jan, Feb, Mar, Apr, May.

==========

TASGAIN4

by Bruce Wall
390 Cadell Street,
HAY. NSW. 2711
(069) 93 2079

[SYDTRUG Editor's Note: This article has been derived from the "README" file which accompanies the programs which are described here. Bruce has obviously put a great deal of effort into preparing this package and has very generously allowed it to be included in our public domain library. This sort of donation is what makes public domain software so worthwhile. Thank you Bruce!]

TASGAIN4 is comprised of four interacting BASIC program modules and one JCL file. Program operation is menu driven and does not require reference to documentation.

< START/JCL > facilitates commencement of program operation by loading Basic and running the control menu, TASCT001. (Simply type and enter "DO START".)

1) <TASCT001>: Control program which facilitates set-up. This program module presents the operator with the main menu. An option available from this menu [for 128K machines] allows the operator to automatically install MEMDISK and load the remaining three program modules. (The presence of MEMDISK is not essential to the operation - but if available, can be used to facilitate faster interaction of the four chained program modules.)

2) <TASIN001>: This first application module leads the operator through the process of entering data and carries out extensive verification of data inputs. This module also provides a facility for displaying, editing and filing current data inputs and for the retrieval, display, editing, and re-filing of previously recorded data.

3) <TASCA001>: This module carries out calculation and collation procedures.

4) <TASPR001>: Provides a formatting facility and carries out the printing procedure.

It is of significant interest to pilots that, for a given power setting, flight at higher altitude results in an increase in airspeed. This gain in airspeed is available to aircraft with normally aspirated piston engines, (using normally accepted engine rpm) progressively to about 7000', whilst the gain available to aircraft with supercharged piston engines extends to significantly higher altitudes. (Turbo Prop/Jet not considered here).

The rate of net increase, in the lower levels of the atmosphere is, in temperate latitudes, approx. 1.70% of IAS per 1000' gain in altitude. The magnitude of this increase, as it affects light aircraft, is not large, but assumes greater significance as the duration of flight increases.

The net advantage [increase in TAS] changes in accordance with the combined effect of a number of [variable] factors and this computer program enables the operator to enter the initial value of these variable factors and to subsequently determine the cruise altitude for a given flight that will return the shortest overall flight time. Notwithstanding the effects of different wind velocities at different altitudes, the optimum altitude thus determined, will be, in effect, the consequence of a trade-off between;

(a) the increased TAS available at higher altitudes (with a corresponding decrease in fuel consumption).

(b) the cost of slower TAS incurred during the climb (with a corresponding increase in fuel consumption).

This determination is made by comparing total flight times, (through climb, cruise and descent) utilising all available cruise altitudes.

This particular version of TASGAIN4 (01.01.03), imposes a ceiling limitation of 10,000 ft. and is configured to suit the performance profiles of aircraft fitted with normally aspirated piston engines having variable pitch propellers. The program also imposes limitations that are a consequence of observing normal engine [rpm] constraints - (the effect of which limits the magnitude of increase in TAS [available from increasing Density Altitude values]). When ISA [International Standard Atmosphere] conditions exist, these inherited [maximum] limitations are as detailed below.

(a) TAS gain over IAS during the climb is limited to 15%. (If IAS for climb is maintained, then this limitation is normally reached at 8000'.)

(b) TAS gain over IAS during cruise is limited to 11% and normally occurs at 6000'.

(c) TAS gain over IAS during descent is limited to 15%.

(When values diverge from the ISA scale (or the nominated reference scale), then the altitudes referred to above in (a) and (b) will vary in accordance. Notwithstanding other variable factors [W/V and excessive high/low airspeed values etc.], the attainment of these altitudes is further dependant on the set of conditions which determine "Highest Attainable Cruise Altitude".)

Projected values, upon which a Density Altitude coefficient is established, are derived from either one of two scales. [The operator makes this selection during program execution.]

(a) The ISA temperature and atmospheric pressure lapse rate scale,

(b) A scale established from long term data averages obtained from the Bureau of Meteorology taken at 35 deg. 10' south latitude (Eastern Australia).

ASSUMPTIONS:

* The aircraft altimeter is set to QNH [indicating elevation of Departure Airfield above mean sea level], is set to Pressure Altitude, or is set 1013.2 mb [default Pressure Altitude sub scale setting].

* The flight is deemed to commence at the reference point [runway level] of the Departure Airfield and is deemed to finish at 1500' above the reference point of the Destination Airfield.

* Runway heading at the Departure Airfield is the same as the track to be flown.

* Indicated Airspeed [for climb] is established at the reference point of the Departure Airfield and is maintained [at a constant indication for the duration of the climb].

* Normal climb rate is established at the reference point of the Departure Airfield.

(It is a further assumption that if CLIAS is established as defined above, then 'normal rate of climb' will also be established at the reference point of the Departure Airfield.)

The changing "rate of climb" values used in the time to climb calculations are derived from the nominated value input for "Initial Rate of Climb at Sea Level").

DEFINITIONS:

QNH is used to define a subscale reference in altimetry which enables a correction to be made for atmospheric pressure divergence. (An altimeter with the correct QNH value set on the subscale will indicate height above mean sea level.)

Pressure Altitude refers to the altitude amsl. indicated on an aircraft altimeter when the subscale reference is set to standard sea level ISA value (1013.2 mb).

Density Altitude refers to the altitude amsl. where observed air density values would occur in the ISA. Atmospheric density varies in accordance with variations in Pressure, Temperature and Humidity. Subsequently, Density Altitude, at a given level, is ever changing. (Humidity values are not readily available and their influence, within the formulae used in this program, would not be significant if included, so are subsequently ignored.)

Arrival Altitude is determined by adding 1500' to the elevation of the Destination Airfield.

Initial Climb refers to the climb from the Departure Airfield to the first whole thousand feet level above the elevation of the Departure Airfield.

Final Descent refers to the descent from the last whole thousand feet level above Arrival Altitude - to Arrival Altitude.

Minimum Available Cruise Altitude is defined as the greater of the following two;

(a) The first whole thousand feet level at or above Arrival Altitude.

(b) The first whole thousand feet level above the elevation of the Departure Airfield.

Mid-Altitude Whole thousand feet level plus 500'. [Minimum Available Cruise Altitude plus 500'~9500'.]

Highest Attainable Cruise Altitude is dependent on the combined effect of the following variable factors.

(a) Elevation of Departure Airfield
(b) Rate of Climb (overall average)
(c) Elevation of Destination Airfield
(d) Distance
(e) Ground Speed (overall average).
(f) Rate of Descent

(In most cases the Highest Attainable Cruise Altitude will extend to 10,000' (program limitation).

Density Altitude coefficient represents any deviation from the ISA scale (or nominated reference scale) and is expressed in feet [plus/minus]. This coefficient is derived from the following data inputs.

(i) Surface Temperature [at the Departure Airfield] together with either;

(ii) Pressure Altitude or,

(iii) QNH.

(Data value (i) may be either factual or default, data values (ii) and (iii) may be either factual, default or deduced. This data is subsequently used to establish a precise value for Density Altitude.)

CRITERIA:

Heading and Ground Speed values are calculated with reference to Wind Velocity at each whole thousand feet level, whilst TAS and Rate of Climb values are calculated with reference to Density Altitude at each Mid-Altitude (whole thousand plus 500).

Rate of Descent value remains constant during program execution and assumes (by default) the last used value - (initial value 500 fpm.).

The lowest altitude referenced is Minimum Available Cruising Altitude.

The time to climb each 1000' step (or part of) is calculated using the climb rate applicable to Mid-Altitude; i.e., the time to climb from 2000' to 3000' is calculated using the climb rate that exists at 2500'.

True airspeed during climb is re-calculated [increased] at each Mid-Altitude [whole thousand plus 500] level, by an increment derived from the following formula;

((IASCLIMB/100)*1.7 * ((MID ALTITUDE + DENSITY ALTITUDE COEFFICIENT)/1000)

The reference altitude is the next Mid-Altitude [Mid-Altitude modified by 'Density Altitude Factor']; i.e., True Airspeed that exists at 2500' is contained in the formula used to calculate the applicable ground speed and the distance travelled during the climb from 2000' to 3000'. Wind Velocity criteria that exists at the next (higher) 1000' level are referenced in conjunction for inclusion in this formula.

True Airspeed during cruise is calculated at each whole thousand feet level in the following manner.

TASCRUISE [level] = IASCRUISE [Sea Level] plus an increment derived from the following formula.

((IASCRUISE/100)*1.7 * ((MID ALTITUDE + DENSITY ALTITUDE COEFFICIENT)/1000)

Calculations are undertaken at each whole thousand feet level between "Minimum Available Cruising Altitude" and "Highest Attainable Cruising Altitude" [inclusively].

True Airspeed during descent is re-calculated [attenuated] at each Mid- Altitude [whole thousand plus 500] level, by an increment derived from the following formula;

((IASDESCENT/100)*1.7 * ((MID ALTITUDE + DENSITY ALTITUDE COEFFICIENT)/1000)

The reference altitude is the next Mid-Altitude [Mid-Altitude modified by 'Density Altitude Factor']; i.e., True Airspeed that exists at 2500' is contained in the formula used to calculate the applicable ground speed and the distance travelled during the descent from 3000' to 2000'. Wind Velocity criteria that exists at the previous (higher) 1000' level are referenced in conjunction for inclusion in this formula.

(Thus, within the same lateral band of airspace, the same criteria are used to calculate climb rates and airspeeds for all facets of flight.)

Observed and forecast Wind Velocity values are normally attributed to whole thousand feet levels and this convention is maintained.

LIMITATIONS AND CONTROLS IMPOSED ON DATA INPUTS:

(All data inputs must be numerical characters; the value of the input must be an integer and must comply with the minimum and maximum limitations listed below, all else rejected).

	Total Distance of proposed flight . .	20 ~	999	(naut.miles)
%	Direction (Track and Wind)	001 ~	360	(deg.mag.)
	Initial rate of climb (ISA at sea level)	100 ~	2000	(fpm)
	TAS (Climb, Cruise and Descent) . . .	70 ~	500	(Kts.)
!	Wind Speed	1 ~	60	(Kts.)
$	Climb rate attenuating factor	30 ~	100	(fpm/1000)
$	Rate of Descent	200 ~	2000	(fpm)
	Fuel allocation - startup to departure	1 ~	60	(ltrs)
	Fuel consumption rates; climb	9 ~	140	(ltrs/hr)
	cruise . . .	9 ~	140	(ltrs/hr)
	descent . . .	9 ~	140	(ltrs/hr)
	Fuel allocation - circuit and landing .	1 ~	60	(ltrs)
#	Elevation of Departure Airfield . . .	0 ~	9999	(ft.amsl)
	Elevation of Destination Airfield . .	0 ~	8499	(ft.amsl)
&	Temperature at Departure Airfield (OAT)	-20 ~	+50	(deg.C)
*	Pressure Altitude at Departure Airfield	-1000 ~	9999	(ft.b/amsl)
*	QNH at Departure Airfield	980 ~	1040	(mb)

% Direction - 3 digits required. Leading "0"[s] are subsequently required when the value is less than 100. The value must be between 1~360.

! Wind Speed - 2 digits required. Leading "0" is subsequently required when the value is less than 10. Wind speed inputs are subjected to one further test, before acceptance, to ensure acceptable relationship with airspeed inputs [previously input]. Wind speed cannot exceed 50% of any nominated airspeed - [Climb IAS, Cruise IAS, Descent IAS].

$ Non mandatory - (Input retains last used value by default).

If Pressure Altitude at Departure Airfield is factual, (i.e., either input, edited from a default value or derived from factual QNH, then during any subsequent data editing, (Elevation of Departure) the limitation [during editing] is; Pressure Altitude -900'~+800'.

& If OAT has been initially entered by default, then OAT will automatically be changed [in accordance with the applicable default lapse scale] if Elevation of Departure Airfield is changed during data editing.

* Choice of either one during initial data input. Pressure Altitude may be entered by default [same as Elevation of Departure]. The limitation on subsequent editing (Pressure Altitude) is; Elevation of Departure -800'~+900'.

When Pressure Altitude at Departure Airfield is factual [input, or later changed using the data edit facility] or when Pressure Altitude is allotted a value by default, [same as Elevation of Departure] then QNH is allotted a value according to the applicable default lapse scale.

When QNH at Departure Airfield is factual [input, or later changed using the data edit facility], then Pressure Altitude at the Departure Airfield is allotted a value according to the applicable default lapse scale.

Values for (a) Pressure Altitude at Departure Airfield and (b) Elevation of Departure Airfield are reconciled (both during initial input and during data editing) to ensure that the variation between the two remain realistic. This verification procedure is necessary to ensure the maintenance of realistic [subsequently deduced, or default] QNH values - (980mb~1040mb).

Criteria used in this program when establishing initial values [deduced and or default] for Pressure Altitude, Density Altitude, QNH and OAT [at Departure Airfield].

Elevation	Pressure (mb)		Temperature (C)		Feet per mb	
	ISA	*	ISA	*	ISA	*
10000	696.9	695.77	-05.00	-00.1878	36.90	35.0347
9000	724.0	724.31	-03.00	+01.4488	35.09	34.2898
8000	752.5	753.47	-01.00	+03.0854	34.02	33.5449
7000	781.9	783.28	+01.00	+04.7220	33.33	32.8000
6000	811.9	813.77	+03.00	+06.2464	32.26	32.0351
5000	842.9	844.99	+05.00	+07.7708	31.06	30.8400
4000	875.1	877.42	+07.00	+09.3921	30.49	30.2636
3000	907.9	910.46	+09.00	+11.0134	29.24	29.2000
2000	942.1	944.70	+11.00	+12.5020	28.41	29.2000
1000	977.3	978.95	+13.00	+13.9906	27.86	29.2000
SL	1013.2	1013.2	+15.00	+15.0000	-----	-------

* Derived from Bureau of Meteorology observations [long term, annual averages] obtained at 35 deg. 10' South Latitude (Eastern Australia).

==========

Help Wanted

Bruce Wall, the author of the previous article, has run into a problem with WORD5, and would appreciate any help that may be available from other members. His problem is outlined below:

> When running the spelling checker and encountering a word in the document that is not in the **Standard** Dictionary, SPELL usually provides a display of alternative [similar] word(s) from the **Standard** Dictionary, however, in circumstances where the word is not contained in the **Standard** Dictionary but is contained in a **User** Dictionary, and a slight misspelling occurs, then alternative [similar] word(s) are <u>never</u> displayed, [I believe they should be]. Even when **Option** is selected from Spell menu, [to invoke the added User.CMP dictionary file, where a range of similar words does exist] <u>none</u> are ever displayed from the **User** Dictionary. However, when proceeding and selecting **Correct**, the new replacement word <u>is</u> examined by the **User** Dictionary before acceptance. Page 526 of WORD5 user manual indicates "if the word is not in the dictionaries, **Spell** displays a list of alternatives" (note dictionaries <u>plural</u>).

Editor's Note: This may, of course, be a design fault with **WORD5** (or should that be SPELL?), but if anyone knows how to get around the problem they may like to give Bruce a call to discuss it with him. His home phone number is (069) 93-1488. It would be greatly appreciated if any successful method of overcoming this problem could be reported in "SYDTRUG News" for the benefit of other readers who may also have run into this problem.

This is the sort of thing that group membership is all about, all of us at some time or another have had problems which we were unable to fix on our own. They are very rarely unique problems. Someone else has almost always experienced the same trouble. They may have found a solution. If you don't ask, it is unlikely that you will be told the answer. If you do get a solution to your problem then let the rest of us know, don't keep it to yourself.

==========

Your Computer Tutorial - Part 7

by Les BELL

[Reprinted by courtesy of "Your Computer", from the May 1986 issue.]

I'll begin this month by temporarily diverting your attention from disk files to the point where the whole process starts - with the information being input - and show you how inputting can be made neater and more professional.

To do this we're going to tackle screen handling and user-definable functions.

The BASICs in some machines include simple screen-handling functions, in the form of the CLS and PRINT@ statements, so certain parts of this information will not apply to users of TRS-80, System 80 and similar systems. However, some of the other techniques may prove useful.

Inputting data to a computer is like filling in a form - the whole process can be made far more comfortable if the screen is organised to look just like a form. To do this with a standard 'dumb' computer terminal we must output special control codes or escape sequences to the terminal. These codes command it to perform such functions as clear screen, position cursor and so on.

Different terminals have different sequences of control codes for their functions; that's why programs like WordStar, which makes extensive use of terminal functions, comes with an INSTALL program. INSTALL sets up the appropriate codes in the program to make it work on a particular terminal. You can do something similar in your programs by setting the appropriate codes in in string variables and functions at the beginning of a program.

For example the LEAR-SIEGLER ADM-3A terminal will clear its screen if you send a CHR$(26). In the case of a Televideo TVI 910, the appropriate character sequence is CHR$(27) followed by CHR$(26). For an ADDS Regent terminal, it is CHR$(12).

Times Getting Tougher Than...

Okay, imagine someone gives you a program written for the ADDS with PRINT CHR$(12) statements all the way through it and you own a TVI910.

You're in luck if you have a text editor with global search and replace function: otherwise, you'll have to go through and find every occurrence of PRINT CHR$(12) and change it to PRINT CHR$(27);CHR$(26). You'll also have to watch out for LPRINT CHR$(12) statements - they send form feeds to the printer....

Life was meant to be easier than this , wasn't it?

Surely it would have been better if your friend had written at the beginning of the program:

```
10 CLS$ = CHR$(12)
```

and then used print CLS$ all the way through. All you would have to do to be in business is change line 10 to:

```
CLS$ = CHR$(27) + CHR$(26)
```

As a bonus this even works faster, though you're unlikely to notice the difference.

You can use a similar technique to send [the] cursor home, up, down, left and right commands to any terminal. For example, for the TVI910:

```
10 HM$ = CHR$(30)
20 CLS$= CHR$(27) + CHR$(26)
30 UP$ = CHR$(11)
40 DN$ = CHR$(10)
50 LE$ = CHR$(8)
60 RT$ = CHR$(12)
```

Now we are starting to get somewhere! But how about more complex jobs, like moving the cursor to a particular row and column? For the ADM-3A this means sending an escape character, CHR$(27), followed by an equals sign, then row number plus an offset of 31 (as a binary number), then the column in the same fashion.

How do we treat this case?

That's where **user-definable functions** come in. In Microsoft BASIC and CBASIC-2 you can define your own functions by using the DEF FN statement. For example, we can define a function which converts Centigrade to Fahrenheit. The formula is F = 1.8 by C + 32 (we used it earlier in a simple program).

To define a function called FNF(C) we write:

```
70 DEF FNF(C) = 1.8 * C + 32
```

In this case, the function is really called F, or FNF (function F) in full. The bracketed C is a **dummy variable**. It bears no relationship to the variable C which may be used elsewhere in the program.

It simply means the number appearing in brackets when the function is called should be used as the dummy variable C in the calculation.

Use Of The Dummy

To show the use of this simple function in a program, here's an example:

```
10 DEF FNF(C) = 1.8 * C + 32
20 INPUT "Centigrade";X
30 PRINT "equals";FNF(X);"Fahrenheit"
40 END
```

Note the function must be defined before it is used, and that although in the definition C is used as the dummy variable, when it is called [it] will operate on whatever variable is passed to it.

Now in this example the function returned a real number. However, functions can return other data types too - and function names follow the same rules as variable names. So an integer function could be defined and named FNDO%(X), or a string function named FNHS$(A$).

To return to our problem of PRINTing the escape sequence which positions the cursor, here's a possible solution:

```
10 DEF FNGXY$(X,Y)=CHR$(27)="="+CHR$(Y+31)+CHR$(X+31)
```

Here we've defined a function GXY$ (goto XY) which is to be passed two dummy variables X and Y. It then constructs a string consisting of ESCape, =, Y plus offset, X plus offset. On an ADM-3A then, the line

```
240 PRINT FNGXY$(40,12);
```

will position the cursor near the centre of the screen.

Different functions will be required for other terminals; on a Hazeltine terminal you would use this function:

```
10 DEF FNGXY$(X,Y)=CHR$(27)+CHR$(17)+CHR$(X+31)+CHR$(Y+31)
```

On the ADDS Regent, it would be

```
10 DEF FNGXY$(X,Y)=CHR$(27)+"Y"+CHR$(Y+63)+CHR$(X+63)
```

This technique can be used to produce rudimentary graphics on serial terminals. Try this short program - after modifying it for your terminal, of course.

```
100 DEF FNGXY$(X,Y)=CHR$(27)+"="+CHR$(Y+31)+CHR$(X+31)
110 CLS=CHR$(27)+CHR$(26)
120 AS$=CHR$(27)+CHR$(72)+CHR$(72)
130 PRINT AS$
140 PRINT CL$
150 FOR X=1 TO 80
160  FOR Y=1 TO 24
170   PRINT FNGXY$(X,Y);"*";
180  NEXT Y
190 NEXT X
200 PRINT AS$
```

The AS$ is a string to turn auto-scrolling on and off on the TVI910. Without it the terminal will automatically scroll up periodically, spoiling the display.

Well It's Round-ish, Sir

To display a circle (of sorts) try this:

```
100 DEF FNGXY$(X,Y)=CHR$(27)+"="+CHR$(Y+31)+CHR$(X+31)
110 CLS$=CHR$(27)+CHR$(26)
120 AS$=CHR$(27)+CHR$(72)
130 PRINT AS$
140 PRINT CLS$
150 FOR X=-10 TO 10
160  YC1=12+SQR(100-X^2)
165  YC2=12-SQR(100-X^2)
170  XC=60+X
180  PRINT FNGXY$(XC,YC1);"*";FNGXY$(XC,YC2);*;
190 NEXT X
200 PRINT AS$
210 PRINT FNGXY$(1,1);:LIST
```

Many terminals have other functions. On the TVI910, for example, the strings 'ESC)' and 'ESC (' turn half-intensity on and off. So we can define a function to print a string in half intensity:

```
30 DEF FNHT$(A$)=CHR$(27)+")"+A$+CHR$(27)+"("
```

To print a string of underlines:

```
40 DEF FNLN$(L)=STRING$(L,95)
```

Here the STRING$ function is used to generate a string of length L, composed of underlines (CHR$(95)). To print a string, but underlined (on the TVI910):

```
50 DEF FNUL$(A$)=CHR$(27)+"G8"+A$+CHR$(27)+"G0"
```

Take a look through the manual for your terminal; you will find many functions that can be controlled this way. It's possible to lock and unlock the keyboard, turn a printer on and off, make characters blink or inverse video, and so on.

The most important use of these functions is in the creation of forms for input and formatted output. In the case of our telephone directory, we are dealing with fixed length records, which cannot be exceeded. It would be handy to know how much space is available for a name before we start filling it in. We can do this by printing up a blank form with underlines indicating the space available for data entry.

To take us out this month, here's an example of a short routine which could be used in our telephone directory program.

```
10 DEF FNGXY$(X,Y)=CHR$(27)+"="+CHR$(Y+31)+CHR$(X+31)
12 DEF FNHT$(A$)=CHR$(27)+A$
13 DEF FNLN$(L)=STRING$(L,95)
14 HOME$=CHR$(30)
15 CLS$=CHR$(27)+CHR$(26)
17 PRINT CLS$
20 PRINT FNGXY$(1,3);FNHT$("Surname    : ");FNLN$(20)
30 PRINT FNHT$("First Name : ");FNLN$(20)
40 PRINT FNHT$("Street     : ");FNLN$(30)
50 PRINT FNHT$("Town/City  : ");FNLN$(20)
60 PRINT FNHT$("Postcode   : ");FNLN$(4)
70 PRINT FNHT$("Telephone  : ");FNLN$(15)
75 PRINT FNHT$("Comment    : ");FNLN$(19)
80 PRINT FNGXY$(13,3);:INPUT N$
90 PRINT FNGXY$(13,3);": ";N$;SPACE$(20-LEN(N$))
100 PRINT FNGXY$(13,4);:INPUT C$
110 PRINT FNGXY$(13,4);": ";C$;
120 PRINT FNGXY$(13,5);:INPUT A1$
130 PRINT FNGXY$(13,5);": ";A1$;SPACE$(30-LEN(A1$))
140 PRINT FNGXY$(13,6);:INPUT A2$
150 PRINT FNGXY$(13,6);": ";A2$;SPACE$(20-LEN(A2$))
160 PRINT FNGXY$(13,7);:INPUT PC%
170 PRINT FNGXY$(13,7);":";PC%;SPACE$(16)
180 PRINT FNGXY$(13,8);:INPUT TEL$
190 PRINT FNGXY$(13,8);": ";TEL$;SPACE$(15-LEN(TEL$))
200 PRINT FNGXY$(13,9);:INPUT CT$
210 PRINT FNGXY$(13,9);": ";CT$;SPACE$(19-LEN(CT$))
```

==========

Exchange Newsletters

Some of what is included in our library. These newsletters may be borrowed by members. Members attending meetings at Gefton should see our Librarian. Other members may apply to our P.O. Box. Postage will, of course, be charged for those forwarded by mail.

March 1990

"Adelaide Micro User News"
Newsletter of the Adelaide Micro User Group
G.P.O. Box 214, ADELAIDE S.A. 5001

Echo Blank Lines - Reprinted from NCTCUG Journal:
XCOPY Xpertese - Reprinted from NCTCUG Journal:
Do We Count From Zero - More discussion on the logic of whether decades or centuries start with years ending in zero or one:
More Answers to Rod's Ramblings - As with the previous item:
More on Input Routines - Continuing with a series:
A 'C' Tutorial - It looks like being a series:
PCTOOLS Deluxe Version 5 - Part 1 - Reprinted from Canberra Micro-80 Newsletter:
All your Files Dated 31 Dec 1987? - Reprinted from "SYDTRUG News":
What Do I Do With It? - Reprinted from Canberra Micro-80 Newsletter:
Using MEMDISK For the Model 4/4P - Reprinted from NCTCUG Journal:
Print Visicalc Formulas on the 4/4P - Reprinted from "Bits & Bytes".

"LLIST"
Newsletter of the Calgary Color Computer Club
Box 22 STN."M", CALGARY. ALBERTA. T2P 2G9, CANADA.

Just Leave it to Nobody - Humorous item:
Rascan Video Digitizing Package - An overview.

"Canberra Micro-80"
Newsletter of the Canberra Micro-80 Users Group
18 Callabonna Street, KALEEN ACT 2617

PUBLISH IT! In Use - Overview of a desktop publishing package:
Clod's Guide to MS WORD - First of a series.

"Eastern Suburbs 80 Users Group Newsletter"
Newsletter of the Eastern Suburbs 80 Users Group
17 Douglas Avenue, BOX HILL SOUTH VIC 3128

Introduction to C - First of a series ??:
RISC vs CISC - Some discussion (ex BBS??):
The Other Side - A retailer's horror story.

"The Voice of FCUG"
Newsletter of The Fairfield County Computer Users Group
10 Richlee Road, NORWALK CT 06851, US of A

Bill's Bumblings No. 49 - More of a Pascal program (I think!):
Hard Disk Tools - More on the subject of utilities etc. for maintaining hard drives. Mainly refers to MS/PC-DOS:
Novice Nook #20 - Stocks and Computers (as in Stocks and shares):
The Cache Stash - Some thoughts on MS/PC-DOS commands FOR, DO EXIST etc.:
Friendly Computer Glossary - Flowchart to Fuzzy Logic:
Old Man of the Screen IV - A mention of music and computers:
SWFTE Glyphix Fonts - An overview of a product recommended for use with Laser Printers and Word Processors:
The Temporary WORDPERFECT Macro - A useful tip for users of WordPerfect.

"HAWTUG NEWS"
Newsletter of the Hawaii TRS-80 User Group,
366 Elelupe Road, HONOLULU HAWAII 96821

Grammatik III - An overview of an "Electric Webster"-like proofreader for use with MS/PC-DOS wordprocessor files:
Printer Claims - Some words of caution about some of the "Hype" in ads for printers relating to their alleged speed and throughput:
Infectious Diseases - About Trojan Horses, Viruses, and Worms.

"Thuggery"
Newsletter of The Hobart Users Group
P.O. Box 420, MOONAH TASMANIA 7009

Presidents Page - A few thoughts about hard drives:
Bert's Ravings - Special Edition - Transferring files from TRS-80 Models 3/4 to MS-DOS 3.x:
Review of Personal C - A shareware version of "C":
Short History of Fractals - The title tells it all:
Programmers Dictionary - Humorous item.

"NATGUG News"
Newsletter of the National Amstrad, Tandy & General User Group
11 Elizabeth Road, SUTTON COLDFIELD ENGLAND B73 5AR

MS-DOS Mysteries Unravelled - Part 3 - Continuing the series.

"National Capital Tandy Computer Users Group"
Newsletter of the National Capital Tandy Computer Users Group
P.O. Box 2826, FAIRFAX VIRGINIA. 22031, US of A

The President's Column - Some generalisations about tax return preparation programs:
The Milliken Storyteller - About an MS/PC-DOS educational program for children learning to read:
Fractools - About an MS/PC-DOS program which develops fractals and permits modifications:
Your CONFIG.SYS and AUTOEXEC.BAT - Second part of the article reprinted from "Voice of FCUG":
Formatting 720K Diskettes to 1.44M - A Bad Idea - A technical reason why this should not be done:
How to be a Computer Geek - A humorous item.

"MICROBITS"
Newsletter of New Zealand TRS-80 Users Group,
P.O. Box 19000, Auckland 7, New Zealand

Introduction to the C Tutorial - The first of a series.

"The Interface"
Newsletter of The San Gabriel Valley Tandy Users Group
P.O. Box 6818, BURBANK CA 91510, US of A

Disk Check - A brief discussion of various disk formats:
Switched On BATch - A couple of BATch files for MS/PC-DOS:
Starview - A review of an MS/PC-DOS shareware program.

"Bits & Bytes"
Newsletter of the TRS-80 System 80 Computer Group
41 Montclair Street, Aspley Qld. 4034

PATCH/CMD - For NEWDOS/80 Version 2.x:
NewDos86 Program Save Routine - A BASIC routine to change the version number of a program each time it is saved with NewDos86:
40 Meg Hard Drive - About the author's installation of Roy Soltoff's Hard drive interface package with his own drive:
Hanging Indents in Superscripsit - Model 4 - An explanation of how it's done:
Notes on Club Helpdisk Project -
Memscanner - A program for scrolling through memory.

"WNYTUG News"
Newsletter of Western New York Tandy Users Group
80 Lockwood Ave., Buffalo NY 14220, US of A

4, 3, 2, 1, OPTIMIZE - A routine to automatically optimize your hard disk every five times you boot up your MS/PC-DOS machine. It could be adapted for other type machines:
Protect Your AUTOEXEC.BAT File - Hints to prevent other programs from overwriting you own special version of this file:
BATch File Basics - Some tips for MS/PC-DOS users:
Best Bytes - Working at Home With Computers - How to go about establishing a computer-based office at home.

==========

Membership and BBS Fees

Members who have not yet paid their fees for the 1990 to 1991 financial year are reminded that these fees were due for payment on 1st July. Attached to the back of this newsletter is a form for you to complete and return with your membership fees in case you have mislaid the one sent previously.

When our group became incorporated last May we adopted the model rules as set out in the regulations of the Association Incorporation Act 1984.

Rule 4 states that a member will only cease to be a member if:
(a) They die;
(b) they resign their membership; or
(c) they are expelled from the association.

Rule 6 (2) states, in part, that a member who's fees are paid up to date, may resign by giving not less than one month's notice in writing to the secretary.

This means that membership DOES NOT CEASE JUST BECAUSE MEMBERSHIP FEES HAVE NOT BEEN PAID. If you do not wish to continue to be a member YOU MUST TENDER A WRITTEN RESIGNATION. HOWEVER, although membership will continue even though fees have not been paid, this DOES NOT mean that privileges will not be curtailed. The group's largest expense is the publication and distribution of "SYDTRUG News", and we cannot continue to send copies to those members who are unfinancial.

If you have not paid your membership fees for the current year, THIS IS THE LAST COPY OF "SYDTRUG News" YOU WILL RECEIVE, until such time as your fees are paid. Access to the Bulletin Boards may also be limited.

YOU HAVE BEEN WARNED!

Of course if you are fully paid up then you have no worries.

==========

SYDTRUG Inc.

The Sydney TRS-80 / MS-DOS Users Group

Payment of Annual Membership Fees for 1990/1991

Family Name : _____ Membership No. : _____

Given Name(s) : _____

Preferred Name : _____

Telephone No. Home : _____ Work : _____

Residential Address		Postal Address	
Street	: _____	Street	: _____
	: _____		: _____
Suburb	: _____	Suburb	: _____
State	: _____	State	: _____
Post Code	: _____	Post Code	: _____

Please fill in details of your computer equipment and interests on the back of this form so that the Group may be better able to support you.

Payment Details

Group Membership Fee	$ 25.00
BBS Membership Fee ($15.00) (payment covers both BBS's)	$___.___
Total Amount Payable	$___.___
I enclose cheque/money order for	$___.___

or

Please CHARGE my BANKCARD ☐ MASTERCARD ☐

No. : _____ Card EXPIRY Date. ___/___

with the amount of $___.___

Signature : _____ Date ___/___/___

Cheques should be crossed and made payable to "SYDTRUG Inc.".
CHARGE CARD payments MUST BE SIGNED and should be returned in a sealed envelope.
This completed form should be returned to :

The Secretary, SYDTRUG Inc., P.O. Box 223, BANKSTOWN NSW 2200

Office Use Only :

Received ___/___/___ Updated : Members Database ___/___/___

Receipt No. : _____ BBS - Club-80 ___/___/___

Badge No. : _____ - Trug-86 ___/___/___

SYDTRUG Inc. *The Sydney TRS-80 / MS-DOS Users Group*

Member's Computer System(s) Details

Please enter the details of your computer equipment on this questionaire.
This is not mandatory, but will enable us to determine the services best suited to your needs / interests.
Use [1] to indicate your Primary (most used) System and [X] for any other equipment that you use / own.

Tandy 8 bit Computers & Compatibles

Model I ☐ Model 4 ☐ Model II ☐ Coco 2 ☐ Model 100 ☐
System 80 ☐ Model 4P ☐ Model 12 ☐ Coco 3 ☐ Model 200 ☐
Genie ☐ Model 16 ☐
Model III ☐ Other _____

Storage

Cassette ☐ Drives 5 1/4" 3 1/2" Hard Disk _____ Megs
Cartridge ☐ Sides SS ☐ DS ☐ SS ☐ DS ☐
Density SD ☐ DD ☐ SD ☐ DD ☐
Density 40 ☐ 80 ☐ 40 ☐ 80 ☐

Operating System

Basic Level I ☐ TRS-DOS 2.x ☐ NewDOS 80 ☐ LDOS 5.x ☐ CPM ☐ OS-9 ☐
Level II ☐ TRS-DOS 6.x ☐ _____

MS-DOS based Computers

Brand		Type		Memory	Style		Monitor	
IBM	☐	PC	☐	_____K	Laptop	☐	Mono	☐
Tandy	☐	XT	☐	_____K	Portable	☐	Hercules	☐
Amstrad	☐	AT	☐	_____K	Desktop	☐	CGA	☐
Commodore	☐	386	☐	_____K	Tower	☐	EGA	☐
Other _____		Other _____		_____K	Other _____		VGA	☐

Storage

Disk Drives
5 1/4" 180K ☐ 3 1/2" 720K ☐ Hard Disk _____ Megs MFM ☐ Cartridge _____ Megs
360K ☐ 1.44M ☐ RLL ☐ Optical Drive _____ Megs
720K ☐ ESDI ☐
1.2M ☐ SCSI ☐ CD-ROM ☐
Tape Drives Cassette _____ Megs Cartridge _____ Megs Other _____ _____ Megs

Printer

Dot Matrix		Daisy Wheel		Laser	
C.Itoh & compatible	Model _____	Tandy DWP	Model _____	Postscript	Model _____
Epson & compatible	Model _____	NEC	Model _____	HP	Model _____
Tandy LP / DMP	Model _____	Qume	Model _____	TI	Model _____
IBM & compatible	Model _____	Other _____		Epson	Model _____
Other _____				Brother	Model _____
				Other _____	

Modem

V21 ☐ V23 ☐ V22 ☐ V22bis ☐
Hayes Compatible ☐ Trail Blazer ☐ Other _____

Interests and/or Expertise

Hardware		Applications		Programming			Games	
Design	☐	Utilities	☐	JCL/Batch	☐	Basic ☐	Graphics	☐
Repair	☐	Word Processors	☐	Assembly	☐	C ☐	Text Adv.	☐
		Business	☐	PostScript	☐	Pascal ☐	Educational	☐
		Spread Sheets	☐	Fortran	☐	Forth ☐		
		Databases	☐	Cobol	☐			
		Educational	☐					

Other _____
_____
_____

Office Use Only.
Entered into Database: ____/____/____