C-Net 64 version 12.0

the official

P-FILE WRITER AND C-NET MODIFIER'S GUIDE

written by Don Gladden

TABLE OF CONTENTS

PREFACE

This manual and p-file disk has been a pet project of mine for over three years, and is finally finished, and released, to share with all who love to program and modify the C-Net program. I hope it makes things easier for all to do this! I thank all who helped with suggestions and information, especially Eric Bratton (THE TREE), for his extensive research into the non-documented source code for the C-Net ML. Many thanks also to Professor, Ray Kelm, for his work.

Don Gladden

1. INTRODUCTION

You are about to enter a whole new world that you never knew
existed! Since you have purchased this manual, you are probably
already a C-Net 64 Sysop who loves to run a BBS, but now comes
the opportunity to add your own personal touches by modifying
your system the way YOU want it! This manual was written to make
it as easy as possible to change the C-Net v12 program to do what
you want it to, and the way that you want it to do it! If you
have a working knowledge of Commodore BASIC, along with this
guide, there is no reason that you cannot write your own program
modules, (hereafter referred to as "P-files"), and modifications
to suit your own, and your user's tastes!

C-Net 64 v12.0 was written with this very goal in mind, to make
things EASY to change and modify. This manual is a complete
guidebook to the internals of the system's software and how to
make it work for you. It assumes that you have a working
knowledge of BASIC. If you are a beginner, we recommend that you
obtain one of the many fine books that get you started into BASIC
programming, and then implement the techniques into programming
C-Net. The differences, as you will see, are very slight, and you
will catch on quickly, and be writing your own on-line games and
utilities in no time. If you already know BASIC fairly well,
you've got a good head start!

The five topics of concentration within this manual are as
follows:

A> What a P-file is, and how to begin writing one yourself. All
of the basic necessities of P-file structure.

B> How and how not to choose variables for your pfiles and
modifications, the best ones to use, re-use, and never use!

C> What built-in system routines (both BASIC and machine
language) exist, and how to incorporate them into your own
programming. Information on some of the available C-64 ml
routines and locations is included for reference.

D> Tips and hints for C-Net programming that we can offer.

E> Structure and description of important disk files used by
C-Net.

2. C-NET PROGRAMMING BASICS:

This chapter will detail the actual structure of a p-file, and then how to incorporate variables and system routines into one effectively.

2.1 WHAT IS A P-FILE?

A p-file is a program written in BASIC, along with certain routines needed by C-net that will run on a C-Net BBS. There are certain rules you must follow, due to the C-Net routines and methods used, to assure you that the p-file will run correctly.

2.2 P-FILE STRUCTURE:

C-Net 64 p-files must begin at BASIC line number 1, and may use line numbers in any order whatsoever after that, up to line number 999. The last line of a pfile must NOT contain any line number references, such as a "goto" or "gosub". The best way to make sure you do not make a mistake here, is to just make sure line #999 of any pfile is simply a "rem".

A p-file may be up to 49 blocks long in version 12.0, but no longer. If you cross the 49 block barrier, It will do some strange things! However, with some tricky programming, it is possible to use more than one module to create a very large program to run as p-files. Two multi-user games included on the accompaning p-file disk include EMPIRE and DRAGONWORLD, both of which were multi file games that have been crunched down to under 49 blocks each without losing any features. This 49 block space is "reserved" by the main program, so filling up memory here really does not make too much difference. The memory is already alloted, though you will want to still follow good programming structure to make the p-file run efficiently.

The main C-Net program always gives control to a p-file using "goto 1", thus, p-file execution must always begin at line 1, even if it is only a GOTO another line number deeper into the file.

When your p-file is finished running, you must return control to the main program, or another pfile. If you return to the main program, you would use a "GOTO 1812" to get back to the "Main: " prompt level. Running a pfile from another pfile requires you to set the p-file up as a subroutine, using "return" rather than a "GOTO" statement to end it. This requires that you use a different routine in the first (original) p-file to load and run the second. There is such a routine built into the main program to allow for this. (See chapter on system subroutines).

2.3 P-FILE PROGRAMMING:

One very important thing to do in your p-files and mods is make sure you are safeguarded against someone dropping carrier or running out of time, and having the program get "locked up", not detecting the fact that carrier is gone. To help with this, we

will explain exactly what happens when a user drops carrier or runs out of time, and how you can check for this and make sure things will be ok.

If carrier is dropped, or a user runs out of time, the system immediately sets time to 0 minutes, and sets "tr$" to chr$(0) Then, the ml coding will send carriage returns repeatedly. If the system is at an input prompt, you can check if tr$ is less than a chr$(1), (control-a), and if so, take the program to the time-up routine (line 1080 in the main program) to log the user off and save all necessary data. If the program is within a subroutine, make sure you "return" to another input prompt that checks for tr$ before the "GOTO 1080". If you are not in a subroutine, you may just "GOTO 1080" to take care of things. See the chapter on tips and tricks for an easy way to do this within a p-file.

One of the most common programming mistakes made in p-files is leaving garbage on the stack, which the above is an example of. There are two main ways that this can happen, which both, with proper programming technique, and being careful, can be prevented. Stack garbage accumulates whenever the program breaks out of a subroutine, (never gets to the "return"), or out of a for-next loop, (never gets to the "next". For example, something like this could be written, and not show up until it runs a few times, then all of a sudden, out of memory errors! And fre(0) shows quite a bit left!

EXAMPLE#1:
```
10 fora=1to500
20 ifna$="SYSOP"then50
30 x=int(rnd(1)*50)+1
40 next
50 ...program continues or...
```

EXAMPLE#2:
```
10 gosub100
20 gosub1006:iftr$<"ctrl-a"then1080
100 a=4:b=6:a$="Hello":syso
110 iftr$<"ctrl-a"then1080
115 a$="Hows it going":syso
120 return
```

These could be corrected in the following way, and still work the same:

EXAMPLE#1:
```
10 fora=1to500
20 ifna$="SYSOP"thena=500:goto40
30 x=int(rnd(1)*50)+1
40 next
50 ...
```

EXAMPLE#2:
```
10 gosub100
20 gosub1006:iftr$<"ctrl-a"then1080
100 a=4:b=6:a$="Hello":syso
```

```
110 iftr$<"ctrl-a"thenreturn
115 a$="Hows it going":syso
120 return
```

In example#2, line 110 would not actually even be needed, since as long as you EVENTUALLY return to a prompt with a check for tr$, you'll be ok.

All other programming structure rules apply as usual. These include:

A> "REMARK" (REM) statements throughout to document large routines and variable usage.
B> A general program "flow" from top to bottom without GOTO's every other line jumping all over the code.
C> Make an attempt to use efficient coding. Pay attention to memory usage and run time. Even though your memory is "reserved" for a p-file, efficient programming, and saving room still makes the program run faster.
D> Error guard your input! Check for numeric input too long, too low, too high, etc., to avoid errors such as division by zero, illegal quantity, etc., from ever happening.
E> Never GOTO a line outside of your subroutine without RETURNing first, or break out of a for-next loop.

2.4 BASIC VARIABLE USAGE:

The third chapter of this manual gives a complete listing of all variables that the C-Net software uses. Refer to that chapter when choosing variables to use within your p-files and modifications--do not simply choose random variables out of nowhere. Serious problems can occur, (and have occured), immediately or hours later, after the system has been running for a while as a result of such practices.

For consideration of clarity in programming and memory, do NOT use the automatic array declaration technique (for example, simply using the r$(5) in an operation results in dim r$(10) automatically being performed). No real memory is saved using this technique, and only makes your program harder to follow. Use instead the method described below.

C-Net provides a way for you to create and then, when you are finished, dispose of variables for use within your p-files. Before using this technique, you should attempt to re-use existing variables, as permitted in chapter three. When C-Net gives control to a p-file in most cases, it calls a routine (sys52587) which saves the current position of the variable pointers. When the p-file is finished running, you can restore those pointers, thus eliminating all new variables created by the p-file, by doing a "sys52605". The only drawback to this technique is that you may not (obviously) save any values in a new variable to keep track of anything done in that p-file. If you want to save something like this, for example, number of times played, etc., a good way to do this would be to POKE screen memory somewhere. (I use the top of the screen, along the borders

of the upper status screen) DRAGONWORLD, on the accompaning
p-file disk, uses this technique to keep track of number of games
played.

2.5 BASIC INPUT AND OUTPUT:

Generally, you should never use the BASIC statements
"PRINT","INPUT", or "GET" in a p-file. C-Net provides routines
which take care of these functions in the BBS environment.For
example, to output text to the "console", (both screen and user
on-line to the system), the "syso" routine is used. (See chapter
4 for a detailed description) instead of PRINT. Example:

```
a$="Hello world!":syso
```

This will print "Hello world!" to the screen, and leave the
cursor at the end of the output line without performing a
carriage return. A small modification to the line:

```
lp=1:a$="Hello world!":syso
```

will still print "Hello world!", but will perform a carriage
return after the text.

INPUT is replaced by one of two GOSUBS. GOSUB 1006 will allow
input of a line of text, but only allow uppercase and numeric
input. GOSUB 1005 will allow both upper and lowercase. The text
will be returned in an$. To get a numeric input, the VAL(an$)
function must then be used.

GOSUB 1007 will take care of the GET function. The key pressed
will be returned both in a$ and an$.

3. BASIC VARIABLE USAGE

This chapter contains a COMPLETE list of all BASIC variables used by the stock C-Net 64 v12.0 including all system program files. Each variable is given along with a description of how it is used throughout the program.

When you are making modifications to the C-Net program itself, and you require a permenant variable for your routines, (one that must keep a value), you should stay clear of ANY of the variables listed below, as they are all used in one place or another in the program. When you only require a temporary variable for your purposes, you should pick carefully from the list instead of adding another variable to the system, to save on memory and make things run faster. For example, say you wanted to keep a counter of the total number of 2400 baud callers. That would be a variable that you don't ever want C-Net to use, so you would pick a new one, such as a1%. On the other hand, say you wanted to print out ten random numbers before the main prompt. Instead of picking new variables out of the air to bog the system down with, such as "forz9=1 to 10:r9=rnd(0)", you could save the extra memory usage and use "for x=1 to 10:r=rnd(0)" instead.

! Denotes an operating system variable, which must not be interfered with by external programming! There is no problem, however, with reading these variables or using them in calculations, as long as no assignment occurs.

* Denotes a variable that is used in one or more of the program modules of C-Net. (Main areas of the BBS, such as subs, UDs, news, etc.), These variables are usually O.K. to use in a P-file or modifications to the main ("cn") program. - Denotes a variable that is used and/or altered by one or more system routines, as described in the text following the variable. Is is only safe to use one of these variables if one of the routines that uses it is not called at the time that you need it. (Temporary use). Check the "NOTES" section under any subroutines that you use in the main program to make sure that no variables you are using are altered by the routine.

AI	- a	Temporary use.
	- a$	Temporary use. MCI variable 5.
	- a1	Temporary use.
	- a2	Temporary use.
	- a3	Temporary use.
	- a4	Temporary use.
AC I	! ac%	Access group number (0-9) of current user
	* ac%(30)	Subboards and UDs- Access codes for each subboard.
	- a%(44)	Temporary use.
	! ag$	Access group name of current user.
	! ak$	Thirty-eight -'s, (ASCII) or chr$(192)'s (C/G) followed by a carriage return. MCI variable j.
	- an$	Temporary use. MCI variable 7.
	! ao%	Access group of user at logon. Used to check

```
                     if access was changed by sysop while user is
                     on-line, to read in new access data if necessary.
  BI- b              Temporary use.
    - b%(44)         Temporary use.
    - b$             Temporary use. MCI variable 6.
    - b1$            Temporary use.
    * bb$(30)        Used in "p.s" (subboards), "p.u/d" and "p.diredit"
                     to hold names of subboards. In subboards, if name is
                     preceded with an "a/" or "p/", that board is either an
                     anonymous or a password board.
    ! bd             Total number of blocks that current user has ever
                     downloaded.
  BFL - bf           Temporary storage of blocks free.
    ! bf(15)         Blocks free on all system devices. (1-6="system"
                     devices, 7-15=device numbers). Will contain blocks
                     free of last drive accessed (0 through 9) if it is a
                     dual drive or Lt. Kernal hard drive.
    - bl             Unneeded, free to use, but reset to 0 during input
                     routines.
    * bn             Subboards and UDs- current board number.
                     weed- counter for how many users weeded.
                     copier- error flag.
    - bp             Temporary use.
    * br             Subboards and UDs- pointer to current bulletin/file
                     being accessed.
                     Email- Current message number being accessed.
                     News, P and G files- How many items in current
                     directory list.
    ! bu             Total number of blocks that current user has ever
                     uploaded.
    * bz             Subboards and UDs- Number of subboards the current
                     user has access to.
  CI - c             Temporary use.
    * c%(60)         Subboards and UDs- Holds ID# of user who
                     posted/uploaded each file on directory.
                     Email- How many lines into "cm." file that each
                     message begins at.
                     News, P and G files- access code for each entry.
    - c$             Temporary use.
    ! ca             Total number of calls since system was started.
    ! cf             Chat flag. (Was chat requested?) 1=yes 0=no.
    ! ch$            16 character chat message.
    * cm$            Set to current "Area"- UD#2, Email, etc.
    ! cn             Total number of calls since the system was booted.
    ! co$            Computer type (name) of current user.
  CTYPE! co$(9)      All computer types- (1-9).
    ! co%            Computer type (number) of current user.
    ! cr             Credit points of current user.
    ! ct%            Number of calls current user has made today.
  DI - d             Temporary use.
    * d%(60)         Subboards- contains number of responses to each
                     bulletin. UDs- contains computer type for each file on
                     directory. Email- contains ID number of user who sent
                     each message. News, P and G files- Flag for whether
                     user has access to specific entries.
  DI1 - d1%          Device number if dr used to specify a particular
```

device for a disk command.

D2% d2%   Drive number if dr=0. (Used to specify a particular drive for a disk command.

! d1$ The current date and time. This string is 11 ASCII characters in length, in the form: WYYMMDDHHMM where W is the day of the week (1-7, 1=Sunday), YY is the year, MM the month, DD the date, HH the hour (80 is added if the hour is PM) and MM the minutes. MCI variable 0.

* d2$ Board name for entry files. MCI variable 8.

! d3$ Last caller to system. MCI variable 9.

! d4$ Name of the last protocol loaded. MCI variable 1.

! d5$ The current users true last call date. MCI variable k.

! d6$ Date and time of log restart.

- da$ Temporary use.

! dc   Downloads the current user has made this call.

D2B - dr   Used to designate "system" device/drive when calling disk routines. 1-system disk, 2-email disk, etc., or dr=bn+6 when accessing subboard/UD devices.

- dr$ Holds number of drive plus a colon ("0:") for disk routines.

* dt$(60)   Subboards- contains a 22 digit string for each bulletin, 1-11=date of bulletin, 12-22=date of last response. UDs- contains 11 digit date of upload for each file. Email- Date of each message. P and G files- Holds names of directories to create a "path" to follow backwards on "<" command. "p.t" (term)- phone numbers in phonebook file.

DVI - dv%   Device number used in disk routines.

! dv%(36)   Holds device numbers of all system devices.
dv%(1)=system disk   dv%(2)=email disk
dv%(3)=etcetera disk   dv%(4)=gfiles disk
dv%(5)=pfiles disk   dv%(6)=user disk
dv%(7)-dv%(36) contain subboard device numbers while in subboards or UDs.

EI - e   Temporary use.

* e%(30)   Subboards and UDs- contains the "actual" subboard number for C-Net to find files on the disk for any given subboard (because different groups can access different subboards). "p.t" (term)- baud rates from phonebook file.

* eb   Subboards- End of bulletin flag.

* ed$(60)   Subboards- contains titles of bulletins. UDs- contains descriptions of files. Email- Subject of each message. News, P and G files- Title of each entry.

- ef   Temporary use.

! el   Number of editor lines current user is allowed.

! em   Expert mode flag.

! er$(29)   Holds system (BASIC) error messages.

* f%(60)   UDs- contains number of times downloaded for each file.

- f1   Used for a flag to enter a pfile and jump to a certain line number within the pfile.

* f2   Subboards- flag for "are you on an anonymous subboard".

! f9	Picks which screen that will be displayed at "System Idle."
	1= Clock
	2= B.A.R. screen
	3= Blocks free screen
* fb	UDs- temporary blocks free variable.
- ff	Temporary use. Used to print number of blocks free on screen.
! ff$	First name of current user
! fl$	Access flag string of current user.
! fl$(9)	Default access flags of all access groups.
- fr$	Temporary use.
- g	Temporary use.
* gt	Email- flag for sending mail to more than one user.
! h	Number of times current user has guessed on password board.
- hi%	Temporary use.
- i	Temporary use. Usually used in for-next loops.
! i%	Instant logon flag.
! i1$	Holds data for sysop instant logon. First character is access group number, 2-xx = Sysop's handle.
! i2$	Data for instant logon. First character is expert mode flag, chars 2-15=phone number, chars 16-xx =real name.
! i3$	Data for instant logon. First four characters are ASCII characters representing: 1= minutes per call (0 =unlimited). 2= calls per day (0 = unlimited). 3= idle time (0 = unlimited). 4= downloads per call (0 = unlimited). chars 5-xx = access group name.
! id	ID number of current user.
- id$	P and G files- used to hold either "P" or "G" depending on which you are in. Temporary use.
! ii$	System two-letter login identifier.
- j	Temporary use.
- k	Temporary use.
- kk	Temporary use. After using editor, contains number of lines used+1. (0 if editor was aborted).
- kk$	Temporary use.
! lc	Location.
	0=System Idle
	1=Main
	2=Subboards
	5=U/Ds
	6=Email
	7=News or P/G files
	9=Logon
! ld$	Requested last date of current user. Set to true last call if "LD" command was not used, or date was not saved at last logoff. See d1$ for format. MCI variable 1.
! lf	Linefeed flag. 1=yes 0=no
* ll	P and G files- Number of directories that were accessed.
! ll$	Last name of current user.
ll%	Line length of current user. (22-80 columns)
- lp	Set lp=1 before "syso" to send a carriage return

Handwritten margin notes: "Instant" near ! i%, "THousand" near ll%

automatically after output.

-	lr	Temporary use.
!	lt$	Logon time and date of current user. See dl$ for format.
-	ml%	Temporary use.
!	mm	Used in ml input routine.
!	mq	Mail flag. mq=1 if current user has mail.
-	ms$	Temporary use.
!	mt	Modem type of current user.
-	mw	syso; if mw is set to 1 before a syso, the file will abort immediately upon pressing either the space bar or the "/" key.
!	mx	Modem xor identifier.
!	na$	Handle of current user. MCI variable 2.
!	nc	Number of credits to start new users at.
!	nl	ASCII-C/G mode flag. 1=C/G mode 0=ASCII mode
*	nn$	Subboards and Email- Name of message sender. Temporary use.
*	nn$(60)	Subboards- Handle of user who posted a message. UDs- Handle of user who uploaded file. Email- Handle of user who sent each message. News- Date of each entry. P and G files- Source of files.
*	nr%	Subboards- flag for whether a board has new activity.
!	o	Constant for output routine. (sys52904)
*	oc$	Subboards and UDs- 30 character string of zeros and ones telling whether boards are open or closed. 1=closed 0=open.
-	p	Temporary use.
!	p1%	Number of minutes per call for prime time. (0=no prime time configured)
!	p2%	Beginning time for prime time period. Format: Number of minutes since midnight. (60= 1:00 AM etc.)
!	p3%	Ending time for prime time period.
!	ph$	Phone number of current user. Format: (313)/437-9486 MCI variable 4.
-	pl	Input routine. (GOSUB 1028) Set pl=1 for uppercase only.
-	pp	Printer offline flag. 1=offline 0=online
!	pp$	Password for password board.
-	pq	Temporary use.
!	pr	Number of the last protocol loaded.
!	pr$	Name of the last Pfile loaded.
!	ps	Total number of bulletins the current user has ever posted.
!	pt%	Prime time flag. 1=prime time 0=no
*	pt$	Email- user handle to send message to.
!	pw$	Password of current user.
-	q	Temporary use.
!	qb	Baud rate of current user.
-	qe	Subboards and UDs- flag for any new activity.
*	r	Subboards and UDs- board number requested.
!	r$	chr$(13) (Carriage return).
-	rc	syso; rc will be set by ml coding to 1 if the spacebar was hit during output.
-	rc$	Temporary use.

* rn	Subboards and UDs- Number of bulletins/files on current subboard. Email- number of messages waiting. News, P and G files- Current entry number.
! rn$	Real name (first and last) of current user. MCI variable 3.
! rp	Total number of responses the current user has ever made.
* rq	Subboards and UDs- Has a request to access "new" been made? 1=yes 0=no
- rs	Main: - flag for whether an "all levels" command was issued. 1=yes 0=no Subboards- Number of current response.
- s	Temporary use.
- s$	Temporary use.
* sa	Subboards and UDs- Subop access flag. 1=current user is the subop of current board.
- sh	syso; if sh=47 after output, the "/" key was pressed.
* so%(30)	Subboards and UDs- The account number of subop of the current board.
* sp	Subboards- Direction for scan/read/about. 1=forward -1=backward
! st	System drive status variable.
! st(38)	Status variables for the B.A.R. screen.

feedback :	1	12	23	30
sysop mail:	2	13	24	31
user mail :	3	14	25	32
posts :	4	15	26	33
responses :	5	16	27	34
uploads :	6	17	28	35
downloads :	7	18	--	36
new users :	8	19	29	--
calls/log :	9	20	--	--
time used :	10	21	--	37
time idle :	11	22	--	38

! sy$	Holds name of last configuration loaded. (Sub or U/D).
- t	Temporary use.
! tl	Time of last logoff/logon for setting st() variables.
! tc%	Total number of calls current user has ever made to the system.
! tr$	contains ASCII character representing time remaining on system. To read, use asc(tr$).
- tt$	Temporary use.
- tt$(101)	Editor text. "p.t" (term)- contains names of BBS's in phonebook file.
- tz	Temporary use.
! uc	Uploads the current user has made this call.
! uh	Number of current active accounts on the system. (ur minus number of deleted accounts).
! ul	Uppercase only flag. 1=yes 0=no
! ur	Number of users on file. (includes deleted accounts)
- uu$	Temporary use.

! w		Used in ml input routine.
! ww		Used in ml input routine.
- x		Temporary use.
! x$		ASCII string of drive numbers. 1st digit is system disk, 2nd email disk, etc. 7 through 36 are for subboards.
- y		Temporary use.
- y$		Temporary use.
- z		Temporary use.
- z$		Temporary use.
! zz		Local and psuedo local mode flag. 1=on 0=off

NOTE: sh$() is used in "p.s" for the purpose of editing bulletins, but should NOT be used, as it is not kept in memory, but is used as a temporary array only. This can cause a "redim'd array" error in the system if it is used.

SYNOPSIS:System variables NEVER to use:

```
ac%    ag$    ak$    ao%    bd     bf()   bu     ca     cf     ch$    cn
co$    co$    co%    cr     ct%    d1$    d3$    d4$    d5$    d6$    dc
dv%()  el     em     er$()  f9     ff$    fl$    fl$()  h      i%     i1$
i2$    i3$    id     ii$    lc     ld$    lf     ll$    ll%    lt$    mm
mq     mt     mx     na$    nc     n      lo     p1%    p2%    p3%    ph$
pp$    pr     pr$    ps     pt%    pw$    qb     r$     rn$    rp     st
st()   sy$    t1     tc%    tr$    uc     uh     ul     ur     w      ww
x$     zz
```

System variables used in system P-files only. (OK to use in your own P-files and at Main level):

```
ac%()  bb$()  bn     br     bz     c%()   cm$    d%()   d2$    dt$()
e%()   eb     ed$()  f%()   f2     fb     gt     ll     nn$    nn$()
nr%    oc$    pt$    r      rn     rq     sa     so%()  sp
```

System variables used temporarily by system routines: (Use carefully, check any routines in Main program that may use one or more of these before calling. They may alter your variable!):

```
a      a$     a1     a2     a3     a4     a%()   an$    b      b%()
b$     b1$    bf     bl     bp     c      c$     d      d1%    d2%
da$    dr     dr$    dv%    e      ef     fl     ff     fr$    g
hi%    l      id$    j      k      kk     kk$    lp     lr     ml%
ms$    mw     p      pl     pp     pq     q      qe     rc     rc$
rs     s      s$     sh     t      tt$    tt$()  tz     uu$    xy
y$     z      z$
```

4. SYSTEM SUBROUTINES:

Just as it is a waste of memory to use unnecessary variables in your modifications and p-files, so it would also be a waste of memory to re-write code that is already available for you. This chapter contains a COMPLETE listing of the subroutines in the "cn" (main) program. Each subroutine is documented in the following manner:

```
LINE NUMBER
FUNCTION: Description of the routine.
INPUT   : What to do before calling the routine.
RESULTS : What comes back from the routine.
NOTES   : Variable usage and special considerations.
```

4.1 GENERAL INPUT/OUTPUT ROUTINES:

OUTPUT ROUTINES:

GOSUB1001 (or "syso")

FUNCTION: Output (print) a line of text. (up to 255 chars).
INPUT   : a$="(text to print)", lp=1 will print a carriage return after the text, mw=1 will allow aborting output upon pressing spacebar or "/" key.
RESULTS : rc=1 if spacebar was pressed during output, sh=47 if "/" key was pressed during output. rc will automatically reset to 0, sh will not.
NOTES   : lp,mw,rc,sh altered. Putting a control-d in text to be output immediatly before a "date string" will translate the 11 digit numeric string into expanded form. Example:
"18702030405" will read "Sun Feb 3, 1987 4:05 AM"
The british pound key has special meaning in output text. (See the MCI chapter in your C-Net manual for more details).

GOSUB1017

FUNCTION: Output a carriage return.
NOTES   : a$ altered.

GOSUB1018

FUNCTION: Output two carriage returns.
NOTES   : a$ altered.

INPUT ROUTINES:

GOSUB1005

FUNCTION: Allow the user to input a line of text in upper and/or lowercase.
RESULTS : an$="(text entered by user)".
NOTES   : bl,pl altered, tr$=chr$(0) if time has run out, or carrier/local mode dropped. If a limited amount of characters are desired, poke53252, (number of characters max).

GOSUB1006

FUNCTION: Allow the user to input a line of text in uppercase/numeric only.
RESULTS : an$="(text entered by user).
NOTES   : bl,pl altered. tr$=chr$(0) if time has run out, or carrier/local mode dropped. If a limited amount of characters is desired, poke53252, (number of characters max).

GOSUB1028

FUNCTION: Same as above two routines, except pl must be set

before calling to specify upper or lowercase.

GOSUB1007

FUNCTION: Get one character (uppercase/numeric only) from user.

RESULTS : a$ and an$="(character entered)".

GOSUB1499

FUNCTION: Input an$, then translate any characters from an$ to true Commodore ASCII. (Used for patterns in disk commands).

RESULTS : an$ will contain true Commodore ASCII characters needed.

NOTES : bl,pl,x,a$,an$ altered. Certain characters are translated to different character values by C-Net, to allow the use of these characters in the names of posts, and such. This routine changes those translated characters to their actual values.

4.2 DISK COMMAND ROUTINES:

*SPECIAL NOTE: To set the drive/device numbers for any of these routines, you may enter "dr=x" to specify any of the "system" devices, (e.g. 1="system" disk, 2 = "email" disk, etc.), OR you may set dr=0, d1%=device number, d2%=drive number to address any particular device/ drive.

GOSUB1002

FUNCTION: Position relative file pointer open to file number 2.

INPUT : x=file record number.

NOTES : a,s$ altered. This routine has been updated. See chapter 6.

GOSUB1003

FUNCTION: Check disk error status of device with command channel opened to file #15.

INPUT : Open command channel to file #15 before calling.

RESULTS : a$=two byte ASCII code, eg-"00" for "ok", b$=error text message; an$=two byte ASCII for track#; z$=two byte ASCII for sector.

NOTES : a$,an$,b$,z$ altered.

GOSUB1009

FUNCTION: Set device & drive variables for disk access.

*INPUT : dr=code for which "system" device to set variables for. e.g. 1="system" disk, 2 = "email" disk, etc. OR dr=0, d1%=device number, d2%=drive number.

RESULTS : dv%(0)=d1%, dv%=device number of whatever "system" disk "dr" was set to before calling, dr$=drive number followed by a colon. Example: "0:".NOTES : dv%,dv%(0),dr$ altered.

GOSUB1010

FUNCTION: Set device & drive variables, then open command (error) channel to specified device.

*INPUT : Same as 1009.

NOTES : dv%,dv%(0),dr$ altered.

GOSUB1011

FUNCTION: Set device & drive variables, open command
channel, then open file to specified device, to
file number #2.
*INPUT : a$="(filename and arguments)" e.g.:a$="file,s,r",
Do not include the drive specifier (like "0:"),
it is added by the routine. Otherwise, this
routine takes the same input as 1009.
(dr,dl%,d2%).
NOTES : dv%,dv%(0),dr$ altered.

GOSUB1081

FUNCTION: Update blocks free on specified device/drive.
*INPUT : Same as 1009.
RESULTS : bf=blocks free, bf(device#)=blocks free if drive#
is 0.
NOTES : a,bf,bf(device#),dr,dv%,j,x,dv%(0),an$,b$,dr$ altered.

GOSUB1085

FUNCTION: Check for write error on specified device/drive.
(Usually directory full error).
*INPUT : Same as 1009.
RESULTS : a=1 if error occured, a=0 if not.
NOTES : a,dv%,dv%(0),a$,b$,an$,dr$,z$ altered.

GOSUB1090

FUNCTION: Read and display directory from specified
device/drive.
INPUT : dv%=device number, dr$="(drive number + a
colon)".
(Example: "0:") an$="(pattern)".
NOTES : a$ altered. File number 2 is used in this
operation and closed when finished.

4.3 FILE HANDLING ROUTINES:

GOSUB1023

FUNCTION: Scratch sequential file, then reopen it to
replace.
*INPUT : a$="(filename)", Same as 1009.
NOTES : dv%,dv%(0),dr$ altered. File number 2 is left
open for re-writing.

GOSUB1027

FUNCTION: Open file for read/write/append on "etcetera"
disk.
*INPUT : b$="(filename)", a$=argument: "a"=append,
"r"=read, "w"=write., Same as 1009.
NOTES : dv%,dv%(0),dr$ altered. No argument needed for a
relative file. (a$="").

GOSUB1060

FUNCTION: Open "etc.stats" relative file for read/write.
NOTES : dr,dv%,dv%(0),a$,dr$ altered.

GOSUB1061

FUNCTION: Open "u-alpha" relative file for read/write.
NOTES : dr,dv%,dv%(0),a$,dr$ altered.

GOSUB1062

FUNCTION: Open "cm." (email) sequential file for
read/write/append.
INPUT : tt$="(Handle of user)",a$=(argument):
"r"=read, etc.
NOTES : dr,dv%,dv%(0),a$,dr$ altered.

GOSUB1063

FUNCTION: Open "etc.data" relative file for read/write.
NOTES   : dr,dv%,dv%(0),a$,dr$ altered.

                        GOSUB1065
FUNCTION: Open "u.config" relative file for read/write.
NOTES   : dr,dv%,dv%(0),a$,dr$ altered.

                        GOSUB1030
FUNCTION: Read a sequential file currently opened as file
          number 2 until a "^" is encountered in text,
          space bar or "/" key is hit, or end of file is
          reached.
INPUT   : Open file number 2 for read.
NOTES   : lp,rc,s,sh,a$ altered. File may be opened with a
          different number than 2 by "poke42241,number",
          make sure to poke42241,2 when finished.

                        GOSUB1075
FUNCTION: Read a sequential file using file number 5, clear
          the screen first.
*INPUT   : a$="(Name of file)", Same as 1009.
NOTES   : lp,rc,s,sh,a$,an$,b$,z$ altered. MCI reset.
          Fileread will abort upon pressing the space bar
          or "/" key, or upon encountering a "^" in the
          text, or a line over 80 columns.

                        GOSUB1076
FUNCTION: Read a sequential file using file number 5, do
          NOT clear the screen first.
*INPUT   : a$="(Name of file)", Same as 1009.
NOTES   : lp,rc,s,sh,a$,an$,b$,z$ altered. MCI reset.
          Fileread will abort upon pressing the space bar
          or "/" key, or upon encountering a "^" in the
          text, or a line over 80 columns.

                        GOSUB1350
FUNCTION: Read a sequentail file starting with "sys." from
          the "system" disk.
INPUT   : a$="(filename without the 'sys.')"
NOTES   : a,dr,rc,s,sh,dv%,dv%(0),a$,an$,b$,dr$,z$ altered.

                        GOSUB1351
FUNCTION: Read a "menu" file from the "system" disk, then
          read "menu 8", (Commands available at all
          prompts).
INPUT   : lc=number of menu.
NOTES   : dr,dv%,dv%(0),lp,rc,s,sh,a$,an$,b$,dr$,z$
          altered.

                        GOSUB1490
FUNCTION: Append a string to the "etc.log" file.
INPUT   : a$="(string to append)"
NOTES   : dr,dv%,dv%(0),an$,b$,c$,dr$,z$ altered.

4.4 RELATIVE FILE HANDLING ROUTINES:
"etc.stats"

                        GOSUB1025
FUNCTION: Add one to a status (BAR) variable and write to
          "etc.stats" file.
INPUT   : x=number of record:

	last call	log	crnt	system
Feedback :	1	12	23	30
Sysop Mail:	2	13	24	31

(16)

User Mail :	3	14	25	32
Posts :	4	15	26	33
Responses :	5	16	27	34
Uploads :	6	17	28	35
Downloads :	7	18	--	36
New Users :	8	19	29	--
Calls/log :	9	20	--	--
Time Used :	10	21	--	37
Time Idle :	11	22	--	38

RESULTS : One added to st(x).
NOTES : s$ altered.

                    GOSUB1026
FUNCTION: Write status (BAR) variable to "etc.stats" file.
INPUT : x=number of record. (See 1025 above).
NOTES : s$ altered.

"etc.data"

                    GOSUB1634
FUNCTION: Update access data in from "etc.data" relative
          file for user that is on line.
INPUT : ac%=access group# (Usually set by system...see
          below).
RESULTS : Necessary data is poked to proper locations in
          system, fl$ will be reset to new access flags of
          that access group, el set to number of editor
          lines for that group, ag$ to the name of the
          group.
NOTES : ao%,dr,dv%,dv%(0),el,x,a$,ag$,dr$,fl$,s$ altered.
          ac% is changed by the ml if the access is changed
          using the lightbar from the console.

                    GOSUB1638
FUNCTION: Get "ca" (Total calls to system) from "etc.data"
          file.
RESULTS : ca=number of calls
NOTES : ca,dr,dv%,dv%(0),x,a$,dr$,s$ altered.

"u.config"

                    GOSUB1070
FUNCTION: Write all data of current user on-line to "u.config"
          relative file.
INPUT : None.
NOTES : dr,dv%,dv%(0),x,a$,dr$,s$ altered.

"u.alpha"

                    GOSUB1034
FUNCTION: Delete a user from the "u.alpha" file, update "uh"
          (Number of users in file).
INPUT : an$="(handle of user to delete)"
RESULTS : i=0 if user not found in file.
NOTES : b,c,d,dr,dv%,dv%(0),i,x,z,a$,dr$,s$ altered.

                    GOSUB1038
FUNCTION: Add user to "u.alpha" file, update "uh" (Number of
          users in file).
INPUT : an$="(handle of user to add)", id=id number of user to
          add.
RESULTS : If handle is already in file, i=id# of user with that

(17)

```
                         handle.
    NOTES    : b,c,d,dr,dv%,dv%(0),i,x,z,a$,dr$,s$ altered.
                         GOSUB1046
    FUNCTION: Search "u.alpha" file for a specific handle.
    INPUT    : an$="(handle of user to find)
    RESULTS  : i=id# of user if found in file, 0 if not.
    NOTES    : b,c,d,dr,dv%,dv%(0),i,x,a$,dr$,s$ altered.
```

4.5 P-FILE ROUTINES:
```
                         GOSUB1013
    FUNCTION: Load pfile into memory from "pfile" device (5). Save
                 variable pointers first.
    INPUT    : a$="(filename without the 'p.')"
    NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
                         GOSUB1302
    FUNCTION: Load "p.lo" (logon)
    NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
                         GOSUB1303
    FUNCTION: Load "p.s" (sub-boards)
    NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
                         GOSUB1304
    FUNCTION: Load "p.u/d" (up/download)
    NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
                         GOSUB1305
    FUNCTION: Load "p.E" (edit parameters), and run as a
                 subroutine.
    NOTES    : dr,dv%,tz,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,uu$,z$
                 altered.
                         GOSUB1316
    FUNCTION: Load "p.em" (email)
    NOTES    : dr,dv%,tz,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,uu$,z$
                 altered.
                         GOSUB1317
    FUNCTION: Load "p.n" (news)
    INPUT    : None.
    NOTES    : dr,dv%,tz,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,uu$,z$
                 altered.
```

4.6 SYSOP SCREEN (CONSOLE) DISPLAY ROUTINES:
```
                         GOSUB1360
    FUNCTION: Print a string in the "Area" section of the console
                 screen.
    INPUT    : cm$="(string to print)"
                         GOSUB1371
    FUNCTION: Position cursor on console to second line from bottom
                 of screen.
                         GOSUB1370
    FUNCTION: Print a number (to five digits) with leading zeroes to
                 the second line from the bottom of the console screen.
    INPUT    : a=number to print, b=# digits, c=# of spaces to print
                 from the left column.
    NOTES    : x,a$ altered.
                         GOSUB1378
    FUNCTION: Print blocks free last device/drive accessed on second
                 line from bottom of console screen.
    NOTES    : a,b,c,x,a$ altered.
                         GOSUB1374
```

FUNCTION: Print total calls, calls since bootup, and current
number of users on second line from bottom of
console screen.
INPUT   : g=calls since bootup.
NOTES   : a,b,c,x,a$ altered.

GOSUB1377

FUNCTION: Clear center of bottom line of console screen between
"R:" and "T:" windows.
NOTES   : a$ altered.

GOSUB1376

FUNCTION: Print a string to the center of bottom line of console
screen.
INPUT   : a$="(string to print)".

GOSUB1375

FUNCTION: Print computer type current user on-line in center of
bottom line of console screen.
NOTES   : a$ altered.

4.7 EDITOR ROUTINES:

GOSUB1581

FUNCTION: Load proto 3 (ml editor).
NOTES   : a,dr,dv%,dv%(0),pq,pr,dr$,d4$ altered.

GOSUB1604

FUNCTION: Load and enter ml editor for new message. (No text in
tt$()).
INPUT   : el=number of lines available.
RESULTS : kk=number of lines that were entered+1 (0 editor was
aborted)., tt$(1) to tt$(kk-1) will contain all text
that was entered.
NOTES   : a,b,c,pr,dr,dv%,dv%(0),kk,ml%,pq,x,a$,dr$,d4$,tt$()
altered, MCI reset.

GOSUB1610

FUNCTION: Load/enter ml editor with text in tt$() (Edit existing
message/file).
INPUT   : el=number of lines available, existing text should be
loaded into the tt$() array. kk=number of current
lines of text.
RESULTS : kk=number of lines of new text+1 (0 if editor was
aborted)., tt$(1) to tt$(kk-1) will contain all text
that was entered.
NOTES   : a,b,c,pr,dr,dv%,dv%(0),kk,ml%,pq,x,a$,dr$,d4$ alt, MCI
reset.

4.8 ROUTINES USED FROM PROMPTS:

Note Some of the following routines upon entrance will check an$
from the third character for a device and drive number, and
set device and drive accordingly. (see gosub1470).

#GOSUB1354

FUNCTION: Read a sequential file ("RDx,y").
NOTES   : a,bl,dr,d1%,d2%,dv%,dv%(0),lp,pl,s,sh,rc,x,a$,b$,
an$,dr$,z$ altered.

#GOSUB1450

FUNCTION: Send disk command ("DCx,y").
NOTES   : a,bl,dr,d1%,d2%,dv%,dv%(0),pl,x,a$,an$,b$,dr$,z$
altered.

#GOSUB1088

FUNCTION: Update blocks free (BFx,y).

```
NOTES     : a,b,bf,bf(device#),c,dr,d1%,d2%,dv%,dv%(0),ff,j,
            x,a$,an$,b$ altered.
```
                              #GOSUB1089
FUNCTION: Read directory (DRx,y")
```
NOTES     : a,bl,dr,d1%,d2%,dv%,dv%(0),pl,x,a$,an$,b$,dr$,z$
            altered.
```
                           GOSUB1372
FUNCTION: Update free memory and print on second line from
          bottom of console screen ("MM").
```
NOTES     : a,b,c,x,a$ altered.
```
                           GOSUB1346
FUNCTION: Read "sys.inst" file from "system" disk ("H").
```
NOTES     : a,dr,rc,s,sh,dv%,dv%(0),a$,an$,b$,dr$,z$ altered.
```
                           GOSUB1347
FUNCTION: Read "sys.cred" file from "system" disk ("CR").
```
NOTES     : a,dr,rc,s,sh,dv%,dv%(0),a$,an$,b$,dr$,z$ altered.
```
                           GOSUB1348
FUNCTION: Read "sys.new user" file from "system" disk ("NU").
```
NOTES     : a,dr,rc,s,sh,dv%,dv%(0),a$,an$,b$,dr$,z$ altered.
```
                           GOSUB1349
FUNCTION: Read "sys.config" file from "system" disk ("I").
```
NOTES     : a,dr,rc,s,sh,dv%,dv%(0),a$,an$,b$,dr$,z$ altered.
```
                           GOSUB1640
FUNCTION: Request chat "(C)".
```
NOTES     : a,bl,cf,pl,pp,x,y,a$,an$,ch$ altered, + same as 1678
            (Feedback) if feedback is left.
```
                           GOSUB1656
FUNCTION: Display logon time and time remaining ("T").
```
NOTES     : a$,an$ altered.
```
                           GOSUB1678
FUNCTION: Request to leave feedback ("F").
```
NOTES     : a,b,bf,bf(device),c,j,dr,dv%,dv%(0),i,kk,ml%,pq,
            pr,x,a$,an$,b$,dr$,d4$,tt$(),z$ altered.
```
                           GOSUB1054
FUNCTION: Read "etc.log" ("LOG").
```
NOTES     : a,dr,rc,sh,dv%,dv%(0),a$,an$,b$,dr$,z$ altered.
```
                           GOSUB1870
FUNCTION: Toggle local mode.
```
RESULTS   : zz=new mode value, 1 (on), or 0 (off).
NOTES     : zz,a$,an$ altered.
```
                           GOSUB1880
FUNCTION: Toggle expert mode ("X").
```
RESULTS   : em=new mode value, 1 (on), or 0 (off).
NOTES     : em,a$,an$ altered.
```
                           GOSUB1890
FUNCTION: Toggle ASCII-C/G mode ("AT").
```
RESULTS   : nl=new mode value, 1 (C/G), or 0 (ASCII).
NOTES     : nl,a$ altered.
```
                           GOSUB1910
FUNCTION: Display a "fortune" ("SAY").
```
NOTES     : dr,dv%,dv%(0),x,a$,an$,b$,c$,dr$,s$,z$ altered.
```
                           GOSUB1352
FUNCTION: Read local commands menu 4 ("??").
```
NOTES     : dr,dv%,dv%(0),lp,rc,s,sh,a$,b$,an$,dr$,z$ altered.
```
                           GOSUB1460
FUNCTION: Add credits to user on-line ("CA").
```
NOTES     : cr,a$,an$ altered.
```

4.9 MISCELLANEOUS ROUTINES:

GOSUB1850

FUNCTION: Entry to "commands available at all levels" area.
INPUT   : an$="(command)"
NOTES   : Will execute the following commands when entered:
           local mode   :
"ZZ","RD","DC","CA","??","DR","BF","NL","CD","MM".
           remote   :
"LD","BC","ST","EX","E","LOG","AT","SAY","H","CR","NU",
"I","C","T","F".

GOSUB1004

FUNCTION: Get a digit from the access flags. (fl$).
INPUT   : a=position in string to read:
             1=Non-weed status - 0=no 1=yes
             2=Credit ratio /1.
             3="ZZ" (local) maintainence capability. 0=no 1=yes
             4=Post/respond capability. 0=no 1=yes
             5=U/D capability. 0=no 1=yes
             6=Value+1 * 10 = # of editor lines available.
             7=Unlimited downloads. 0=no 1=yes
             8=General remote maintainence capability.0=no 1=yes
             9=Email capability. 0=no 1=yes
             10=User list capability. 0=no 1=yes
             11=BAR/Log capability. 0=no 1=yes
             12=Sub-board maintainence capability. 0=no 1=yes
             13=G and P-file maintainence capability. 0=no 1=yes
             14=MCI capability. 0=no 1=yes
             15=Downloads at prime time. 0=no 1=yes
RESULTS : a=value.

GOSUB1008

FUNCTION: Check for carrier.
RESULTS : a=0 if carrier present, 16 if not.
NOTES   : a altered.

GOSUB1012

FUNCTION: Pause.
INPUT   : x=2 for each second to pause.
NOTES   : k altered.

GOSUB1019

FUNCTION: Read and set prime time.
RESULTS : pt%=1 if it is prime time, 0 if not.
NOTES   : a,pt%,a$ altered.

GOSUB1091

FUNCTION: Update blocks free in array bf().
INPUT   : bf=blocks free,dv%=device number,dr$=("drive
           number+':'").
RESULTS : bf(device#)=bf.
NOTES   : bf(device#),j altered.

GOSUB1096

FUNCTION: Print a string on the printer if on-line.
INPUT   : a$="(string to print)"
RESULTS : pp=0 if printer on-line, 1 if not.
NOTES   : pp altered.

GOSUB1343

FUNCTION: Get a board number from input to an$.
RESULTS : r=board number.
NOTES   : r altered. Will find board number from third character

of an$ if "UD" was entered, second character if
anything else was entered.

GOSUB1520
FUNCTION: Translate time taken from an 11 digit numeric "date
string" into number of minutes since 12:00 AM.
INPUT   : a$="(11 digit string)"
RESULTS : a=number of minutes.
NOTES   : a altered.

GOSUB1530
FUNCTION: Sound beeps.
INPUT   : x=number of beeps.
NOTES   : a,y altered.

GOSUB1470
FUNCTION: Translate numeric input from third character on in an$
to a device and drive number and set system variables.
RESULTS : d1%,dv%,dv%(0)=device number, d2%=drive number, OR
dr="system" disk.
NOTES   : a,dr,d1%,d2%,dv%,dv%(0),x altered. If number is less
than 8, will set "system" disk variables. If no
numeric input found, will default to device#8 drive#0.
Examples: "xx1" will set dr=1 "xx10,1" will set
dv%,d1%,d1%(0)=10,d2%=1. "xx9" will set
dv%,d1%,d1%(0)=9,d2%=0. "xx" alone will set
dv%,d1%,d1%(0)=8,d2%=0.

GOSUB1652
FUNCTION: Enter chat mode.
RESULTS : cf=0 (chat flag) upon exiting chat.
NOTES   : a,ch$,a$,an$ altered. Time is restored upon exit, or
raised if time was added in chat mode.

GOSUB1662
FUNCTION: Display name of current Sub/UD board.
INPUT   : bn=board number, sy$="(Sub or U/D)".
RESULTS : an$="(name of board)"
NOTES   : a$,an$ altered.

GOSUB1668
FUNCTION: List available Sub/UD boards.
INPUT   : sy$="(Sub or U/D)".
NOTES   : kk,x,a$,an$,b$ altered.

GOSUB1736
FUNCTION: Load proto file.
INPUT   : a=proto number.
RESULTS : d4$="(Name of proto-Punter, Xmodem, Editor or
Copier.)"
NOTES   : dr,dv%,dv%(0),pr,d4$,dr$ altered.

GOSUB1742
FUNCTION: Send a string (modem command to Hayes compat.) to the
modem, pause 2 seconds.
INPUT   : a$="(string to send)"
NOTES   : k,x altered.

GOSUB1744
FUNCTION: Display subop of current Sub/UD board.
INPUT   : bn=board number.
NOTES   : dr,dv%,dv%(0),x,a$,an$,b$,dr$,z$ altered

GOSUB1752
FUNCTION: Inform user if he/she is subop here.
NOTES   : sa,a$ altered.

GOSUB1903

```
FUNCTION: Translate/print "Yes" or "No" from one letter string.
INPUT    : a$=("Y" or "N")
RESULTS  : a=1,a$="Yes." if "Y", a=0,a$="No." if "N".
NOTES    : a,a$ altered.
                         GOSUB1904
FUNCTION: Translate and print "Yes" or "No" from value.
INPUT    : a=1 if "Yes", 0 if "No".
NOTES    : a$ altered.
                         GOSUB1906
FUNCTION: Add up to 5 leading zeroes to a number and place in a
           string.
INPUT    : x=number of digits,a=number.
RESULTS  : a$="(number with leading zeroes.)"
NOTES    : a$ altered.
                         GOSUB1907
FUNCTION: Reset line links for BASIC.
NOTES    : j altered.
                         GOSUB1908
FUNCTION: Display minutes left if not infinate.
NOTES    : a,a$,an$ altered.
                         GOSUB1914
FUNCTION: Reset MCI to default.
NOTES    : a$ altered.
                         GOSUB1915
FUNCTION: Turn checkmark on right of "Loc" on if in true local
           mode.
                         GOSUB1916
FUNCTION: Turn checkmark on right of "Loc" on or off.
INPUT    : zz=1 for on, 0 for off.
```

4.10 TEXT OUTPUT ROUTINES
```
     (a$ is altered in each case)
                         GOSUB1920
FUNCTION: Print "(Aborted)"
                         GOSUB1921
FUNCTION: Print entry message for subboards.
                         GOSUB1922
FUNCTION: Print entry message for U/D boards.
                         GOSUB1924
FUNCTION: Print entry message for editor.
                         GOSUB1927
FUNCTION: Print "returning" message for editor.
                         GOSUB1928
FUNCTION: Print "Chat page is already on."
                         GOSUB1930
FUNCTION: Print "Enter reason for chat".
                         GOSUB1932
FUNCTION: Print "Paging the sysop".
                         GOSUB1934
FUNCTION: Print "Entering chat mode".
                         GOSUB1936
FUNCTION: Print "Exiting chat mode".
                         GOSUB1938
FUNCTION: Print "Available boards".
                         GOSUB1940
FUNCTION: Print "To change boards" message.
                         GOSUB1942
```

FUNCTION: Print "Leave feedback?" message.
                    GOSUB1944
FUNCTION: Print "'Y' to logoff" message.
                    GOSUB1946
FUNCTION: Print "You are subop" message.
                    GOSUB1948
FUNCTION: Print "Mail in your box" message.
                    GOSUB1950
FUNCTION: Print "New users, enter "?" for help" message.
                    GOSUB1980
FUNCTION: Print "Sorry, ".
                    GOSUB1981
FUNCTION: Print "That area is presently closed".
                    GOSUB1982
FUNCTION: Print "Your time for this call is up"
                    GOSUB1983
FUNCTION: Print "Not enough disk space"
                    GOSUB1984
FUNCTION: Print "This library is full"
                    GOSUB1985
FUNCTION: Print "You have left too much feedback".
                    GOSUB1986
FUNCTION: Print "Restricted function"
                    GOSUB1987
FUNCTION: Print "Not cleared for that function"
                    GOSUB1988
FUNCTION: Print "That function is temporarily not available"
                    GOSUB1989
FUNCTION: Print "No mail today"
                    GOSUB1990
FUNCTION: Print "The sysop is not available"
                    GOSUB1991
FUNCTION: Print "Unknown board number"
                    GOSUB1992
FUNCTION: Print "No directory space"
                    GOSUB1995
FUNCTION: Print "ERROR-device not present"

4.11 "GOTO'S"
     The following routines in the "cn" program are not subroutines,
     so should be called with a "goto" instead of a "gosub" if
     needed.
                    GOTO1080
FUNCTION: Logoff for time out/carrier drop.
NOTES   : Should goto here whenever carrier loss is detected, or
           time runs out.
                    GOTO1016
FUNCTION: Load a pfile into memory from the "pfile" device (5),
           then "goto 1" (run). Do NOT Save variable pointers
           first.
INPUT   : a$="(filename without the 'p.')"
NOTES   : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
                    GOTO1300
FUNCTION: Load and run a pfile as a subroutine. (Pfile must exit
           with "return" rather than "goto1812"). Usually used to
           run a pfile from another pfile, and then return to the
           original.

```
INPUT    : z$="(filename without the 'p.')"
NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,uu$,z$
           altered.
                        GOTO1067
FUNCTION: Load and run a pfile from the pfile library.
INPUT    : a$="(filename without the 'p.')"
NOTES    : dr,dv%,dv%(0),f1,a$,an$,b$,c$,cm$,da$,dr$,pr$,uu$,z$
           altered. If pfile is not found on disk, will return to
           pfile in memory (usually "p.f" (files)) with f1=1 to
           flag the error.
                        GOTO1301
FUNCTION: Load and run pfile/ filename same as first two chars
           of an$
NOTES    : This is used for most of the two-letter pfiles, such
           as VF, ST, etc. Can be used with more than two
           letters, as long as the pfile is named with the first
           two. (Example: the "BBS" command loads and runs
           "p.BB".
                        GOTO1306
FUNCTION: Load and run "p.cf" (configure)
NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
           Used only when initially configuring the BBS.
                        GOTO1307
FUNCTION: Load and run "p.su" (setup)
NOTES    : dr,dv%,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,z$ altered.
           Only used when booting the BBS.
                        GOTO1309
FUNCTION: Load and run "p.nu" (new user)
NOTES    : dr,dv%,tz,dv%(0),a$,an$,c$,cm$,da$,dr$,pr$,uu$,z$
           altered.
                        GOTO1310
FUNCTION: Load and run either "p.f" (files) or "p.n" (news),
           depending on what was entered into an$.
NOTES    : if an$="N", "p.n" (news) will be run. Anything else
           will run "p.f" (files).
                        GOTO1318
FUNCTION: Load and run "p.em" (Enter email subsystem).
NOTES    : Will check access first.
                        GOTO1320
FUNCTION: Load and run "p.u/d" (Enter u/d subsystem).
NOTES    : Will check access, light bar flag, and prime time
           first.
                        GOTO1336
FUNCTION: Load and run "p.lo" (logon/off).
                        GOTO1340
FUNCTION: Load and run "p.s" (Enter sub-boards).
                        GOTO1351
FUNCTION: Read "menu 1" file from the "system" disk, then read
           "menu 8", (Commands available at all prompts), Then
           return to "Main: ".
INPUT    : lc=number of menu.
NOTES    : dr,dv%,dv%(0),lp,rc,s,sh,a$,an$,b$,dr$,z$ altered.
                        GOTO1694
FUNCTION: Logoff sequence after "O" or "Q" was entered.
NOTES    : Jumping here will query user for feedback, and read
           "sys.end" file.GOTO1704FUNCTION: Logoff sequence
           without query.
```

GOTO1710

FUNCTION: Reset system after call is updated.
NOTES : Does not update stats, usually jumped to if a false connection occurs. (No logon).

GOTO1800

FUNCTION: Entry point to the main program once logon has been established.
NOTES : Check for mail, news, then go to the Main prompt.

GOTO1812

FUNCTION: Entry point to the "Main: " prompt.

GOTO1900

FUNCTION: Run a pfile from the "Main :" prompt.
NOTES : Will return to "Main: " if the pfile is not on the disk.

GOTO2000

FUNCTION: Error trapping routine. ML routine will jump here if any BASIC error is detected.
NOTES : Will print error, line number, and pfile to printer if on-line, and to "etc.errlog" file on etcetera disk.

GOTO9999

FUNCTION: Reset routine to use in direct mode if run/stop restore is used for any reason.

5. C-NET MACHINE LANGUAGE ROUTINES AND VARIABLES

It is advisable not to change the ml portion of C-Net, since nearly all the memory is used, and this can cause many problems, sometimes impossible to trace. However, the ml routines can be used by the BASIC portion of the program if you have the knowledge of how to use them. This chapter is for that purpose. To detail the routines that are normally used, and show you how to implement them when needed.

5.1 MEMORY MAP:

The ML portion of C-Net resides between $A000 and $D000 in memory. Here is a map of the area:

```
$9E00-$9EFF  256 byte transmit buffer
$9F00-$9FFF  256 byte receive buffer
$A000-$AFFF  routines to control basic text, and the constant
             date-time-I/O buffers.
$B000-$B0FF  output character lookup table for ASCII and
             special character translation during output.
$B100-$B1FF  input character lookup table for ASCII and
             special character translation during input.
$B200-$B2FF  screen character lookup table, used to store
             characters on the screen whose codes change from normal
             Commodore character set codes.
$B300-$BFFF  special functions ML coding. (on-line user data
             editing, lightbar control)
$C000-$CAFF  "proto" area. (punter, xmodem, editor, and
             copier)
$CB00-$CBFF  modem output routines.
$CC00-$CFFF  jump tables and entry locations for all sys
             commands from the main program. These call other
             routines at lower locations.
```

5.2 C-NET "SYS's"

```
SYS51968 : Set up I/O windows.
SYS52152 : Read disk file. (Unabortable)
SETUP    : Open file #2
SYS52204 : Get character from modem.
RESULTS  : peek(780)=ASCII of character.
SYS52230 : Get a character to an$
SYS52238 : Compare date in an$ to ld$ (last date).
SETUP    : an$=11 digit date string to compare
RESULTS  : peek(254)=0 if not "new".
SYS52395 : Enable error trapping routine.
SYS52470 : Display top screen #1. (Caller on-line)
SYS52488 : Display bottom screen.
SYS52544 : Beep. (chat page)
SYS52564 : Terminal mode for C-term
SYS52572 : Check for device present.
SETUP    : open command channel, poke42480,device number
RESULTS  : peek(144)=status 0=ok
SYS52587 : Save pointers to variables.
SYS52605 : Restore pointers to variables.
SYS52720 : Load a proto file.
SETUP    : Open file #6
SYS52736 : Set up IRQ wedge.
SYS52760 : Set up screen handler.
SYS52896 : Input text from keyboard. (modem)
```

```
    RESULTS  : an$="text entered"
        SYS52904 : Output text to screen. (modem)
    SETUP    : a$="text"
        SYS52907 : Convert 11 digit numeric date to full text.
    SETUP    : an$=11 digit date string
    RESULTS  : an$=text representation of string
        SYS53013 : Read disk directory.
    SETUP    : Open file #2
        SYS53021 : Display top screen #2. (Idle)
        SYS53037 : Disable error trapping routine.
        SYS53082 : Read file from disk until EOF. Clear screen first.
    SETUP    : Open file #2
        SYS53085 : Read file from disk until EOF. Do'nt clear screen
                   first
    SETUP    : Open file #2
        SYS53116 : Enter chat mode.
        SYS53138 : Input a password (Prints X's).
    RESULTS  : an$="password"
        SYS53154 : Input line of text from disk
    SETUP    : Open file #2
    RESULTS  : a$="text"
        SYS53224 : Load a p-file into memory.
    SETUP    : Open file #2
        SYS65481 : define an output channel. (used in setup only
                   along with 65490 to check status of devices on-line)
        SYS65484 : Un-CMD output, so PRINT does not go to modem, but
                   screen only.
        SYS65490 : Output a character
```

5.3 ACCESSIBLE MEMORY LOCATIONS FOR PEEKS AND POKES:

```
    142      : temporary use by copier.
    144      : st (status word variable).
    152      : number of current open files.
    186      : current device number- used in C-Net in configuration
               and setup only.
    254      : temporary use. set to 1 if result of a date compare
               (sys52238) is greater (new bulletins, etc).
    512      : temporary use- punter variable.
    603      : last file number opened.
    650      : flag- which keys will repeat? 128 = all keys. 64 = no
               keys. 0 = cursor keys, spacebar, and delete key only.
    661      : low byte for bit timing- (modem).
    662      : high byte for bit timing- (modem).
    665      : low byte- time required to send bit.
    666      : high byte- time required to send bit.
    667      : end of modem receive buffer.
    668      : start of modem receive buffer.
    780      : (accumulator)- after sys52204, this will hold the ASCII
               value of the character from the get.
    781      : (x register)- upon exiting the editor, this location can
               contain the following values: 0 = .A was hit (Abort).
               1 = .S was hit (Save).        2 = .B was hit (Border).
               3 = .H was hit (Help).        4 = Chat mode was entered
    782      : (y register).
    783      : (p (status) register).
    828      : minutes allowed for this call- (255=infinite).
               During setup (boot) this location holds device number
```

that program was booted from.
```
 829    : calls per day allowed- (0=infinite).
 830    : idle time allowed.
 831    : downloads allowed- (0=infinite).
 832    : number of feedbacks that have been left this call.
 833    : number of downloads made this call.
 836    : temporary use.
1007    : MCI access flag in editor- 0=yes 1=no.
1016    : number of lines minus 1 used in the editor.
1018    : temporary editor use.
1019    : temporary editor use.
1022    : number of lines minus 1 allowed in the editor.
1264    : screen checkmark for "Sys"- 160=off 250=on.
1274    : left side screen checkmark for "Loc"- 160=off 250=on.
1278    : right side screen checkmark for "Loc"- 160=off 250=on.
1289    : screen checkmark for "New"- 160=off 250=on.
1294    : screen checkmark for "Prt"- 160=off 250=on.
1299    : screen checkmark for "U/D"- 160=off 250=on.
2040    : temporary editor use.
2043    : file type for upload in term. 1="p" 2="s"
40992   : table for MCI variables.
41328   : table for light bar positions.
41220   : controls speed of cursor spin.
42241   : number of file for ml fileread routine.
42402   : day of week (1-7).
42403   : month (1-12).
42404   : date (1-31.)
42405   : last two digits of year.
42480   : device number for sys52572.
46569   : ascii for "X" for password input.
47802   : flashing chat page- 0=off 1=on.
47903   : clock at idle screen- 0=off 1=on.
50641   : in punter only, should have either 208 or 240
          depending on modem xor value.
52136   : low byte- RS-232 baud rate timer constant.
52137   : high byte- RS-232 baud rate timer constant.
53242   : lines to jump for MCI commands.
53246   : flag- will MCI commands execute on read? 0=yes 1=no.
53248   : flag- local mode - 0=off 1=on.
53249   : flag- upper/lowercase in input routine - 0=upper and
          lowercase allowed 1=uppercase only.
53250   : flag- can graphics characters be entered? 0=yes 10=no.
53251   : time remaining (same as asc(tr$)).
53252   : maximum characters allowed for input routine.
53253   : flag- word wrap for input routine.
53254   : flag- was chat requested? - 2=yes.
53258   : modem xor value (16 or 0).
53262   : flag- C/G mode - 0=off 1=on.
56577   : RS-232 data port register b.
56579   : RS-232 data direction register b.
```

6. SYSTEM FILE INFORMATION
6.1 RELATIVE FILE FORMATS:

u.config: record length: 254 (1 record-23 fields per user).
Each record contains all of the information for a single
user,and is divided into 23 fields, each field seperated by a
carriage return. This table contains a description of each
field, and the variable that is assigned to it when a user logs
onto the system.

Field	Variable	Usage
1	na$	The User's Handle
2	pw$	The User's Password
3	ff$	First Name (Real Name)
4	ll$	Last Name (Real Name)
5	ph$	Area Code/Phone Number
6	ld$	Requested Last Date
7	ac%	Access Group #
8	ct%	Number of Calls Made Today
9	tc%	Total Calls to the System
10	co%	Computer Type (Number)
11	ll%	Line Length/ Number of Columns
12	ul	Upper/Lower Case Flag
13	lf	LineFeeds Flag
14	em	Expert Mode Flag
15	dc	Total Downloads Made
16	uc	Total Uploads Made
17	bd	Total Blocks Downloaded
18	bu	Total Blocks Uploaded
19	cr	Current Credit Points
20	ps	Total Posts Made
21	rp	Total Responses Made
22	d5$	True Last Call Date
23	fl$	User Flags (Various Uses)

u.alpha: record length: 30 (1 record-two fields per user).
The records in this file consist of two fields each, the first
field containing the users handle, (all in alphabetical order),
and the second, his ID number. This file is used to do
alphabetical handle searches only, and uses a binary sort, which
is very fast and efficient. The only drawback to this, is that
it takes quite a while, depending on the size of the file, to
delete a user or change a handle on the system. This is due to
the fact that the file has to be re-structured from that handle
down each time a handle is deleted or added to it. If the handle
begins with an "A", for example, it must shift all handles after
that up or down one record depending on whether that user was
added or deleted.

etc.data: record length: 31
This file contains miscellaneous data for the system.
Record#   Data
    1   : (ca) Total calls to the system. Updated every logoff.
  2-11   : First four bytes=data for access groups 0-9
         Byte #1: minutes per call

(30)

```
                        Byte #2: calls per day
                        Byte #3: idle time
                        Byte #4: downloads per call
                        Bytes#5-xx: access group name.
    12      : (ur) Total users on "u.config" file, including
              deleted accounts. Updated whenever a new user
              logs on.
    13      : (oc$) 30 digit numeric string, designating which
              subboards are open/closed. 0=open 1=closed
    14      : (oc$) 30 digit numeric string, designating which
              UD boards are open/closed. 0=open 1=closed
    15      : Unused.
    16      : (uh) Total users on "u.alpha" file, (does not
              include deleted accounts). Updated whenever a
              new user logs on, or a user is deleted.
    17      : Handle of last caller to the system.
    18      : (pp$) Password for password subboard.
    19      : Date/time of bootup, or last logoff, whichever
              was last.
    20      : (p1%,p2%,p3%) Prime time data. First value is
              number of minutes per call, (0=no prime time),
              second is starting hour (0-23) of prime time,
              third is ending hour of prime time.
    21-30   : (fl$(x)) Default flag strings for access groups.
```

etc.stats: record length: 9

The e.stats file contains information about the current status of the board. Such information as total number of posts, and total files in the U/Ds are held. Also many other useful bits of information are in here. The information in this file is held in the st() array. This array is not an integer array for the reason that some of the information may need a greater range than integer variables can handle.

The numbers in this table pertain to the subscript values for st() and the position on the screen that they are put on the B.A.R. screen:

B.A.R. Stats	Last Call	Log Total	System Total	Current Total
Feedback	1	12	23	30
Sysop Mail	2	13	24	31
User Mail	3	14	25	32
Posts	4	15	26	33
Responses	5	16	27	34
Uploads	6	17	28	35
Downloads	7	18		36
New Users	8	19	29	
Calls/Log	9	20	Log % Tot	
Time Used	10	21	U:	
Time Idle	11	22	I:	

numbers 37 and 38 are not displayed on the chart, but are used to determine the percentages of time and idle since the system started. They contain total number of minutes used, and total number of minutes idle since the system was first configured.

sys.say: record length: 164
This file contains all of the "fortunes" that appear at logon to
the users. The first record in the file simply contains the
number of four line "sayings" in the file, the second record up
to the end of the file contain the actual sayings. Each record
is split into four fields of 38 characters, each seperated by a
carriage return, one field for each line of the saying.

6.2 SEQUENTIAL FILE FORMATS:
Configuration files:

bd.data
This file contains configuration data for the BBS. It is written
on your boot disk upon configuration, and never changed unless
you run the "p.reconfig" to change data.
Entries #1-12: Device and drive numbers for your "system"
                disks, 1 through 6.
    Entry #13: The two letter system identifier.
    Entry #14: Modem type.
    Entry #15: Number of credit points to start for a new
                user.

sys.Sub
This file contains data for your subboards.
Entry #1 contains the number of subboards that the system is
configured for. Then the entries, in groups of five, consist of
the following:
Entry #1: Subboard name
Entry #2: Access code
Entry #3: Subop ID number
Entry #4: Device number for this subboard
Entry #5: Drive number for this subboard

sys.U/D
This file contains data for your UD boards.
Entry #1 contains the number of UD boards that the system is
configured for. Then the entries, in groups of five, consist of
the following:
Entry #1: UD board name
Entry #2: Access code
Entry #3: Subop ID number
Entry #4: Device number for this subboard
Entry #5: Drive number for this subboard

Directory files:

dir.sub xx (xx=number of subboard)
Contains information about bulletins on each individual
subboard.
Entry #1 contains the number of bulletins on the respective
subboard. Then the entries, in groups of five, consist of the
following:
Entry #1: Title of bulletin
Entry #2: Handle of user who posted it
Entry #3: 22 digit string, first 11 digits= date of
            bulletin, second 11 digits= date of last

response.
Entry #4: ID number of user who posted the bulletin
Entry #5: Number of responses

dir.u/d xx (xx=number of UD board)
Contains information about files on each individual UD board.
Entry #1 contains the number of files on the respective UD
board. Then the entries, in groups of six, consist of the
following:
Entry #1: How many blocks the file is
Entry #2: Handle and ID number of uploader
Entry #3: First 11 digits, date of upload. Bytes #12-
          filename and filetype.
Entry #4: Description of the file
Entry #5: Computer type. (number)
Entry #6: Number of times downloade

dn-main
Directory for system news files.
Entry #1 contains the number of files on the directory. Then the
entries, in groups of three, consist of the following:
Entry #1: Title
Entry #2: Date of news
Entry #3: Access coded-title

Directories for system P and G files.
Entry #1 contains the number of files on the directory. Then the
entries, in groups of three, consist of the following:
Entry #1: Title (preceded by a "d-" if a sub-directory)
Entry #2: Source
Entry #3: Access code

7.   C-NET PROGRAMMING TIPS AND HINTS:
This section will give you some ideas and neat tricks to use to
help your p-files run better, and faster.First, we would like to
document and correct some of the most common mistakes found in
programming p-files.
        1> The infamous "Help! The system won't hang up!" trick...As
mentioned in chapter two, one of the most common problems is not
checking in the proper places for carrier loss/time up. If you
simply remember that the system is sending carriage returns if
there is no carrier, you can make sure that the program gets back
to somewhere that will catch this, and log the "user" off. One
thing that I do is try at most prompts to make a carriage return
act just like the user typed a "Q" to main. This is usually ok to
do, unless you want a carriage return to mean something else. The
results: it will send him back the the "main" prompt, which has
the check for time up, and log him off. no problem!
        2> It must be garbage day! The computer is collecting
again!One of the worst things about the Commodore 64 is the fact
that it has to take a ten second break every so often to perform
a garbage collection. Argh! We all know how that is! One way to
at least make these occurances happen less often, is to not
concatenate your strings as often. (This means add them together,
if you didn't know). A nifty trick I like to use takes advantage
of the MCI variables. For example, instead of:
        10 a$="You now have"+str$(cr)+" credits, and"
        20 a$=a$+str$(asc(tr$))+" minutes left.":syso
            I would do something like this:
        10 b$=str$(cr):an$=str$(asc(tr$))
        20 a$="You now have\v6 credits, and\v7 minutes left.":syso
            Note: "\" is the british pound key.
This avoids concatenating strings, and speeds things up quite a
bit.
        3> Although we have said it before, it's worth repeating.
PLEASE check the variables you are using. One of the most elusive
bugs that we had reported with version 12 is that people were
getting a redim problem whenever they tried to edit a bulletin in
the sub-boards. It turned out that one of the popular p-files
used the array sh$(), which is used in subs for the edit routine.
Thus, problems! Some of these may show up in really awful ways,
like deleting accounts, passwords, etc., so BE CAREFUL!
        4> Use the sys52605 to clear out all the added variables as
much as you can. This keeps the system from running low or even
out of memory, which can cause a crash. Another neat trick you
can do is clear out all the system string arrays that you have
used, again clearing memory when the pfile is done. Example: If
you use the tt$() array to store a list of something in your
p-file, when you exit, do something like this:
            for i=0to100:tt$(i)="":next
This will clear it out, and free up to 8k of memory that normally
would not be cleared until the editor was used again.
        5> A nifty trick to make sure you remembered to do all that
stuff, and something I do now with almost all of my p-files, is
use a line like this: 900 forx=0to100:tt$(i)="":next:sys52605:
            on-(tr$<"ctrl-a")goto1080:gosub1018:goto1812
Then instead of referring to line 1080 AND 1812 in the p-file,
you can just use 900, and it will do all the housework for you
upon exit.

6> Another trick is to put all your common subroutines at the beginning of the p-file, starting at line 2, and make line 1 "goto" the actual start of your program. This also speeds up calls to the subroutines, since it does not have to look as far into the program for them. And by all means, use the subroutines in the main CNet program whenever you can. Study them, and use them.

7> Try not to "hard-wire" the p-files to your system. We get many p-files uploaded that start with something like:
            open2,8,2,"sys.instructions",etc.
Many sysops do not use those particular devices like you do! And although it may seem easy to change for you, many do not even know how to change it, so use the regular routines for it. dr=1:a$="sys.instructions":gosub1011 will do the same thing, and use less space, and run on ALL systems with no problem.

8> Whenever you want to seperate text with a horizonital line, use MCI variable j. (ak$). It works fine, and saves memory. If you want a shorter line, just use left$(ak$,12) for a 12-character line, etc.

9> I see many lines of code that end with something like gosubxxx:return. This can always be replaced with gotoxxx (The return from this subroutine call will act as the "final" return here, without hurting the stack!

10> Instead of using "ifa<>0then...", use "ifathen". It will speed things a bit, and perform the same thing. Likewise, any comparison or assignment to 0 can be changed to a period. (a=. or ifa<. rather than a=0 or ifa<0) This also speeds things up a bit.

11> Get to know how to use boolean logic. (One of my favorite pastimes). You can eliminate many "if" statements, and combine more lines of code this way.
            Instead of: 10 ifa=1thena=0 20 ifa=0thena=1
            Use: 10 a=1-a
            Instead of: 10 ifa=0thena$="zero" 20 ifa=1thena$="one"
            Use: 10 a$=mid$("zeroone",a*4+1,4) Try it!!!

12> On input commands, (GOSUB 1005,1006), if you are expecting numeric input, always use 1006 rather than 1005. Since 1005 allows lower case, the typical board crasher will try to enter something like 1e99 to see if the p-file can handle it! Also make sure and use the abs() and int() functions if necessary.

13> C-Net's INPUT and GET routines translate certain characters to avoid DOS problems. They are documented here, so you will know what to check for if you are checking input for any of them.

from	to	key	reverse capital
,	chr$(133)	f1	E
:	chr$(134)	f3	F
"	chr$(135)	f5	G
*	chr$(136)	f7	H
?	chr$(137)	f2	I
=	chr$(138)	f4	J
(CR)	chr$(139)	f6	K
^	chr$(140)	f8	L

Output routines will also recognize these characters and automatically convert them before sending. This is the reason you see many reverse "K"'s instead of chr$(13)'s in the text. They save memory and time in programming.