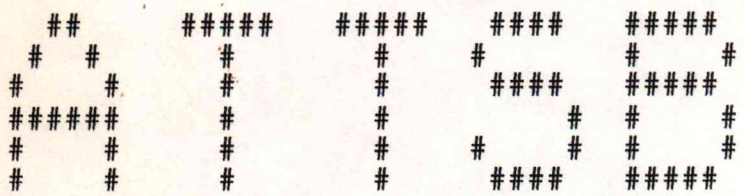
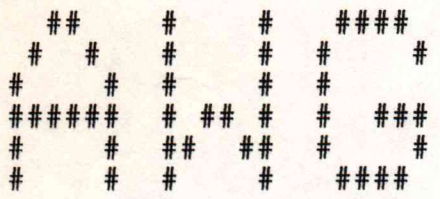
********** remote lp from attsb **********
********** remote lp from attsb **********
********** remote lp from attsb **********
********** remote lp from attsb **********

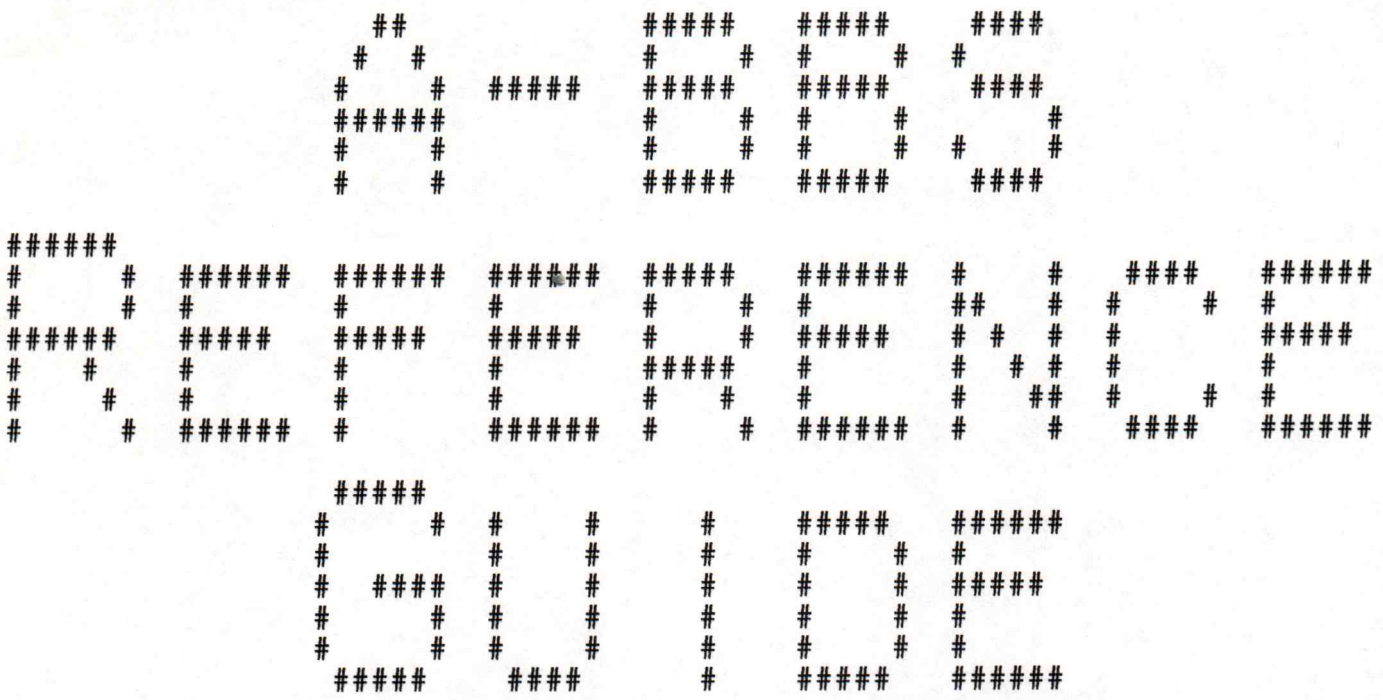




User: a.w.gaunt,V_1j2 x2161 NFMV203140

Request id: attsc_lp-5193 Printer: attsc_lp

Tue Jul 18 16:05:39 EDT 1989



Manual Pages: Prelimnary Manual A-BBS

introduction to a-bbs:

a-bbs is the software for a home security/monitoring and electronic
bulletin board system.
it is written in commodore c128 native basic 7.0 and complied
w/basic 128 by abacus software.

a-bbs requires a commodore 128 with at least one disk drive.
two disk drives (either 1571 or 1581 types) and
a hayes compatible modem (cbm model 1670) are recommended.
the modem may be eliminated if remote operation will not
be used.
an 80 column monitor is recommend as the console screen.
if the console is an configured for 80 column operation
then the rf output may be used as a "background" status
monitor by hooking it up to the homes television circuit.

a-bbs is authored by :
  andrew w. gaunt
  p.o. box 13
  newton-junction, nh
          03859-0013

questions can be answered by sending mail to the above address
or calling the authors a-bbs system at:

New Hampshire (603) 382-3966

sign on the the system as (all upper case ascii)
login: gue
password: *guest*
send mail to user "awg"

brief history:

a-bbs was started around the fall of 1988 and about
ready for release in the spring/summer of 1989.
many hours of programming/debugging and testing have
been put into it.

a-bbs was first meant to be a home security
and monitoring system. soon after it's initial beginning it
was felt that remote access to this system would be desirable.
it was next felt that this remote access could be better used
by adding a means for others to communicate as well. a-bbs grew
into what is is today. it's first 24 hour on-line application was
on april 22 1989.

the author would like to dedicate this system to his wife denise

the two last fields are the date and time that the file last was written.

when a user owns a file the system allows that user to read/write
the file regardless of the read/write levels.

the read level (set with the read command) informs the system
that only users with the set level or above may read the file.
in the above example anyone may read the file. this includes
any operation that requires read access.

the write level (set with the write command) informs the system
that only users with the set level or above may write to the file.
this includes any operation that writes to the file. including
removing it, copying to it or even taking ownership.
in the above example no other user may write to the file, not even the
administrator (with normal means).

note: the read and write commands are available within the shell
      and the editor.

the system supports users from level one to four. level four
is the system administrator.

the read and write levels may be set from zero to five. this allows
the owner of a file to let anyone read/write (read/write level zero)
or no other user to read/write (read/write level five).

this system was implemented to give user control of their files
placed in public directories.

commands description:

- format -
command argument-description
 - command description
 - access level required

-descriptions-

a:, b:, c:, d:, e:
 - change currently logged drive
 - >0

40
 - send output to 40 column screen
 - >3

80
 - send output to 80 column screen
 - >3

breakon
 - set break flag to on, allow a break. default is off, nobreak.
 - >0

cd [directory]
 - change current working directory
 - 1 for level one directories
 - 2 for level one or level two directories
 - >2 for any directory

<cp,copy> [d:]file [d:]file
 - copy files
 - >0 for seq files (a-bbs uses all seq data files).
 - >3 for prg files, none used by a-bbs, except for executable code.

date [mm dd yyyy]
 - read or set system date
 - >0 to read, >3 to set

<dos,@> [string]
 - read drive error channel or send string to disk drive dos
 - >0 to read , >3 to send string

<disp,dump> [<loo[p],sys[tem],pot[s],sch[edule>]
 - dump variable contents
 - >1

echo strings
 - send character strings to terminal (see log and print)
 - >0

ed [[d:]file]
 - edit a file (refer to editor commands)
 - >0 read any seq file, write headered seq files

```
 - disallow breaking submitted scripts, default condition
   when a script is submit
 - >0

play string
 - play music w/sid chip, refer to cbm basic 7.0 play command
 - >3

port [<rea[d],wri[te],and,or,xor>] [byte] [address] [bank]
 - read/write memory
 - >3

print strings
 - send character strings to printer if online (see echo,log)
 - >"value of printer level variable" see variable named "level".

read level
 - set level required by another user to read
   files written with ed or uload (see write).
   no arguments displays current levels.
 - >0

rerun
 - run copy currently in memory (warm-start)
 - >3

restart
 - reload itself into memory and run (cool-start)
 - >3

<rm,scratch> [d:]pattern
 - remove files
 - >1

<set,clear> variable [index] contents
 - change system/environment variables
 - >3

sleep seconds
 - temporarily pause system
 - >0 for 5 or less seconds,
 - >3 for 0 to 20 minutes

slow
 - set slow mode (1Mhz clock)
 - >0

<submit,<> [d:]file ...
 - run a command script
   if the shell encounters an unknown command it will attempt
   to submit the file of the same name as the unknown command
   on the current drive from the directory 1/sub. the file must
   have a valid header.
 - >0

terse
```

```
 - display less information
 - >0

time [hhmmss <am,pm>]
 - read or set time
 - >0 to read
 - >3 to set

<type,cat> [d:]file ...
 - send contents of file to screen
 - >0

uload [d:]file
 - capture input into file
 - >0

user [user [password]]
 - display user info or become user
 - >0

verbose
 - display more information
 - >0

write level
 - set level required by another user order for writing
   to files written with ed or uload (see read).
   no arguments displays current levels.
 - >0

#xmit [d:]file ...
# - transmit file using a protocol (to remote receiver).
# - >0
#not implemented

#xrcv [d:]file ...
# - receive file using a protocol (from remote transmitter).
# - >0
#not implemented
```

```
exec [d:]file
 - run another program
 - >3

exit
 - logout
 - >0

fast
 - set fast mode (2MHz clock)
 - >0

<find,grep> string [d:]file [[d:]file] ..
 - find a string of characters in a file (not a true "grep")
 - >0

<help,?>
 - display help files
 - >0

input [-p] [-c]
 - get a line of input and make it available in
   percent sequence %!
    -p parse line input immediately after it is read.
    -c case convert line input

 - >0

log strings
 - send character strings to file etc/log (see echo,print)
 - >"value of printer level variable" see variable named "level".

loop [number]
 - check status of loops and pots for number of times
 - >0 allows numbers up to 50, >2 allows larger numbers

<ls,dir,directory> [[d:]pattern] ...
 - list files
 - >0

mail [user subject file]
 - read or send mail
 - >0

mode [read-level write-level]
 - set level required by another user to read
   and write files written with ed or uload (see write, read).
   performs same function as read & write commands at one time.
   no arguments displays current levels.
 - >0

<mv,rename> [d:]file [d:]file
 - move files
 - >0

nobreak
```

---

support files description:

system files:

[a-bbs.xxx] - c128 native basic 7.0 source code for abbs, version xxx
p-a-bbs.* - a-bbs executable, compiled w/basic128 by abacus
etc/sprite - sprite data, used to create interrupt in basic

etc/config - script run at boot time to configure set variables etc.
etc/passwd - user names, passwords, access levels and other data
etc/ulog   - user, date, time stamps of successful logins
etc/log   - file appended to by log command
etc/issue   - initial login message, system name, location etc.
etc/motd   - message of the day
etc/auto   - default scheduled script, for 12:00 am
etc/profile - system profile script, executed at login, set defaults etc.
etc/exit   - script file run when user exits (logs off)

these files are scripts run when a monitored loop is triggered and both the exit and entry delays have elapsed.

alm/loop0 - loop zero
alm/loop1 - loop one
alm/loop2 - loop two
alm/loop3 - loop three

these files are scripts run when the status of a pot is above the maximum or below the minimum and no user is logged in. the script will be submitted until the variable count is equal to quit. count is incremented each time the pot routine submits the script.

alm/pot0 - pot zero
alm/pot1 - pot one
alm/pot2 - pot two
alm/pot3 - pot three

system administrator files:

sys/profile - system administrators profile, executed when "sys" logs in.
sys/mail   - mail file for system administrator
sys/free   - a script to display number of disk blocks free
sys/row   - a script to play the song "row row row your boat"
sys/mary   - a script to play the song "mary had a little lamb"

help files:

hlp/* - text files used by help command, this is one of them.

user files:

???/profile - user profile, executed when user "???" logs in.
???/mail   - mail file for user "???"
???/*   - private files accessible by user "???".

public files:

1/* - level one and above users
2/* - level one and above users

---

system variables:

variables may be changed with either the set/clear commands.
there are two types of variables, indexed arrays and non-indexed
variables. to set or clear an indexed variable the index must be
specified as the second argument on the command line. e.g.
set note 1 "here is a string for indexed var. note, index #1"

the set/clear commands map the english words on/off to the number
0 and 1. on and off are considered logical conditions 1 and 0.
variables that are to be used as logically can be set and cleared
more intuitively. i.e. the command lines "set error off" and
"set error 0" will both set the the error variable to "0".

note, the set/clear commands are the same command.

the contents of many of the variables can be seen with the disp/dump
command.

variable name	useful index	expected contents	- description
nam[e]	0-3	text-string	- loop name
mon[itor]	0-3	on,off,0,1	- flag, monitor loop on/off
exi[t]	0-3	0-32000	- number of ticks before alarm is ready
ent[ry]	0-3	0-32000	- number of ticks before alarm flag is set
mas[k]	0-3	0-255	- byte to write to output port
ala[rm]	0-3	0-32000	- seconds mask byte will be written
pot[name]	0-3	text-string	- pot name
max[imum]	0-3	0-32000	- max pot value tolerated
min[inimum]	0-3	0-32000	- min pot value tolerated
slo[pe]	0-3	0-32000	- slope for pot
bia[s]	0-3	0-32000	- offset for pot
qui[t]		0-32000	- number of times pot script runs
cou[nt]		0-32000	- number of times pot script has run normally incremented by system and cleared by user
dri[ve]	0-4	8,9,10,11	- device numbers for drive a,b,c,d,e
mai[l]		8,9,10,11	- device for mail and user files
sys[tem]		8,9,10,11	- device number for system files
err[or]		0,1	- error flag, normally set and cleared internally when software error occurs. ie. a bug or configuration problem
con[nect]		127	- value of user port when modem online currently this is not used. the idle routine simply looks for certain characters from the port. (cbm model 1670)
dis[connect]		47	- value of user port when modem offline if this value is read from the user

port while logged in the exit flag will be set, logging the user out. (cbm model 1670)

bau[d]			- the next four arguments after the baud option will be interpreted as decimal numbers used to build the file name string when opening the tty port. ie. 32 is a space character. changing this may alter the baud rate, parity, stop bits etc. refer to the c128 programmers reference manual for info on how to build this string. default is 1200,N,7,1.
mod[em]		ate1	- modem init string (cbm model 1670)
han[g]		+++	- modem hang up or disconnect string this string will be sent to the modem twice with a two second delay between the sends when the system attempts to disconnect from the tty.
int[ro]			- this string is the very first thing a user sees when logging in at the tty. when the system detects a user logging in at the tty this string is sent. it's default setting is: #send control-c or delete to login
log[in]		0-32000	- maximum number of milliseconds user must successfully login before timeout
key[board]		0-32000	- maximum number of milliseconds user may remain inactive at keyboard before system times out and logs user off.
lev[el]		1-3	- level required for printer access
pri[nter]		0,1,on,off	- printer enable flag
sch[edule]		<0-12> <0-59> <am,pm> [d:]file	- time for script file(s) to be run
ver[bose]		on,off,1,0	- verbose flag
mes[sage]	0-5	string	- message to be displayed to terminal if alarm is triggered
not[e]	0-5	string	- note to be displayed to auxilliary screen in "note" fields.
ide[ntity]		string	- system identity or name

---

percent sequences description:

1) the following characters have special meaning when following the % character on a command line.
   e.g. - %d will may expand to jan 1, 1989

2) if the sequence is surrounded by double quotes then the will be taken literally.

d - date
t - time
o - month number
a - day number
y - year
h - hour
m - minute
i - system identity string
u - user
n - carriage return, line feed
0 - pot 0 status
1 - pot 1 status
2 - pot 2 status
3 - pot 3 status
p - input port status (loops)
g - control-g, bell
r - carriage return
e - escape
b - backspace
s - space
j - line feed
f - form feed
q - double quote
v - memo text of etc/passwd for currently logged in user
! - string of characters from last input command.

command: 40

abstract: use 40 column screen

usage: 40

description: when 40 is invoked a-bbs will send output to the 40 column vic ii screen.

caveats: if the c128 is in fast mode the 40 column screen is off. slow must be used to see what is sent to the 40 column screen.

examples: 40

files: none

see also: 80, fast, slow

routines: none

command: 80

abstract: use 80 column screen

usage: 80

description: when 80 is invoked a-bbs will send output to the 80 column screen.

caveats: none known

examples: 80

files: none

see also: 80, fast, slow

routines: none

command: breakon

abstract: allow a submitted script to be interrupted

usage: breakon

description: breakon sets the break flag to on so that a submitted script can be stopped by sending a control-c or delete. the default setting is off.

caveats: users can abort scripts such as etc/profile or etc/exit. this may not always be desirable.

examples: breakon

see also: nobreak

command: cd

abstract: change the current pseudo directory

usage: cd [directory]

description: a-bbs uses the standard cbm file system. disk directorys are not supported by cbm disk drives. the 1581 drive does support disk partitions however this feature is not taken advantage of. a-bbs uses it's own system of pseudo directorys. this is done by attaching a prefix and a slash character to every file name.
the suggested convention is three character directory names.

cd operates differently for users with different access levels. blocking unauthorized users from certain directorys is a method used to secure files against accidental or unauthorized removal, reading, writing etc. all users may access files in their own private directory. level one and two users may not change to another users private directory. level one, may access any directories that begin with "1/".
level two user may access any directories that begin with "1/" or "2/".
level three and above users may change to any directory.
it is suggested that level 4 be reserved to system administrators only.

caveats: the longer the directory name is the shorter the file name may be. this is because the directories are not true directories. they are actually a part of the cbm dos file name.

examples:
```
cd 1/cbm  - change directory to "1/cbm"
cd        - change to home directory (level one and two)
          - change to top level directory (level three)
cd 1/???  - change to directory "1/???", if the ls/dir
            command is used here then the ??? will be used
            as a wild card for all three letter directory
            names under "1/". this can be used as a
            technique to see available directories and files.
```

see also: <dir,ls>

command: <cp,copy>

abstract: copy a file

usage: <cp,copy> [d:]source [d:]destination [<[-s],-u,-p>]

description: cp is for copying files. the default copy method is stream (-s option) except if the files are on the same disk drive. for files on the same drive cp sends the cbm copy command to the drive.

the -s option (default) is for sequential (seq) files. the files are stream copied. both files are opened (1 read, 1 write) characters are read and written one by one util eof is reached. the output file will be a sequential file type. this is true even if the input file was non-sequential. this may be useful for crude file type conversions.

the -p option is for program (prg) files. the file is buffer copied. the file is loaded into memory bank 1 then written out to the disk.

the -u option is for user (usr) files. it is the same as the -s option except the output file will be a user file. a-bbs does not use any files of this type.

caveats: the stream copy is slow but will not overwrite memory. program files will not be copied properly. this is not much of a problem because there is normally no need to copy program files. buffer copying works only with program files. the danger here is that no checking is done to see if the memory used as the buffer is needed for variable storage.
these concerns do not apply when copying files on the same disk drive.

cp does not work well in a submitted script when it operates of files of the same drive the script is run from.

examples: cp b:mail b:old-mail
cp a:etc/nulog a:etc/ulog
cp a:etc/sprite b:etc/sprite -p

files: *

see also: mv

command: date

abstract: display or set date

usage: date [mm dd yyyy]

description: with no arguments date displays the current date.
if it is not set date will display the message:
[da$-not-set]. if it is then it will display the
month, day year. level three access is required
to set the date.
valid dates are any day beyond the year 1988.

caveats: date must be set when system is boot. there is
no involitial memory store for the date.
date does not check for months shorter than 31 days.
if the date is set for feb 30 date will accept it.
the interrupt driven clock routine will change it
to mar 1 on for the following day.

examples:

```
date 02 28 1989     <- will set date to feb 28 1989
date                <- will respond with feb, 28 1989
```

see also: time, clock

command: dos,@

abstract: send a dos command or read dos status

usage: dos [[d:]cbm dos command] ...

description: dos reads the disk error channel (via ds$) or sends a dos command. no arguments reads the error channel.

typical dos commands may be:

```
i0:                 <- initialize the disk drive
n0:disk-name,id     <- format a disk (all data will be lost)
```

refer to the disk drive's manual for more complete descriptions.

caveats: dos does not check for invalid cbm disk commands or syntax errors.

examples:

```
dos                          <- read the error channel
dos a:i b:i                  <- initialize drive
dos a:"n0:new-disk,nd"       <- format a disk
```

see also: rm, cp, mv

command: dump,disp

abstract: dump out contents of environment variables

usage: dump [<loo[p], pot[s], sys[tem], sch[edule],
buf[fers], mod[em]>]

description: dump displays the contents of environment
variables and alarm loop status flags.
a report is printed to the screen that displays the
the variable and its contents.
if no arguments are given dump displays all variables.

- loop
  variables associated with alarm loops.

- pots
  variables associated with analog loops.

- system
  system variables, flags and status

- schedule
  display status of scheduled scripts; if
  scheduled, when and which ones. if they
  have been run or waiting to be run.

- buffers
  show contents of alarm loop message buffer
  and note buffer

- modem
  baud rate, parity etc. (as described in c128
  reference manual). on/off line detect bytes.
  login prompt introduction string.
  modem initialise and hangup string

caveats: none known

examples: dump
dump modem
dump sys

see also: set,clear

command: echo

abstract: echo command line arguments to screen

usage: echo string ...

description: echo takes the command line arguments passed to it
and sends the to the screen.
if % sequences are used on the command line then echo
echo them after they have been translated.

caveats: echo does not print spaces between arguments. if spaces
are desired in echo's output then enclose the string in
double quotes or use the %s sequence. echo does not print
a carriage return unless it is passed a carriage return.
to do this use the percent sequences %r, %n.

examples: echo "here is a line of text"%n
echo "i am user "%u%n
echo these words will all be run together

see also: print, log

command: ed

abstract: edit the contents of a file in the edit buffer

usage: ed [[d:]file]

description: ed is an interactive text editor. once the editor is invoked the edit prompt is displayed and the editor commands are available. shell commands are not available while in the editor. the commands available may be displayed by typing ? <cr> while in the editor.

the following is a sample of what will be printed:

```
? - help
prompt, x/y  x=total lines, y=current line
print and list mark current line with a *
p            - print +- 6 lines to terminal
l <n1 <n2>> - list all lines or lines n1 to n2
w <d:><file>- write buffer to disk
r <d:><file>- read file into buffer
d <n>        - delete one or n line
i <n>        - insert one or n blank lines
e            - edit current line
a            - append text into buffer
$            - goto end of buffer
z            - zap buffer (no warning)
read [n]     - set read level to n
write [n]    - set write level to n
x,q          - exit editor
nn           - point to line nn

maximum buffer size is 1000 lines  0 % full

-------------------------------------------------------
print and list commands mark current line with a ->

editor prompt description:
x/y d:dir/*
x=total lines, y=current line d:=drive, dir=current dir
```

caveats: the shell and edit prompts are similar in appearance. sometimes one may mistakenly try to use a shell command while in the editor.

when writing a file the name must always be specified. if ed is passed a filename for reading at the shell command line it is forgotten after it is read.

ed never clears (zaps) the buffer when it is exited. if a file is read into the buffer it is always inserted into the buffer. this may produce unwanted results when editing a file after a previous edit has taken place. the buffer must be explicitly cleared.

ed does not edit lines, they must be re-typed.

ed commands cannot be submitted.

examples: ed myfile

files: *

see also:

routines: ed

command: exec

abstract: execute another program

usage: exec [d:]program

description: exec will load another program into memory and run it.

caveats: the user of exec may not be able to re-gain control from the tty.

examples: exec "my-program"

files: *

see also: restart, rerun

command: exit

abstract: exit the system

usage: exit

description: exit sets the exit flag. when this is set
the shell will end. the user will be logged out.
if there is a connection at the modem, it will be
terminated. exit will submit the script etc/exit.

caveats: none known

examples: exit

command: fast

abstract: set the cpu clock to it's fastest speed of 2 mhz.

usage: fast

description: fast sets the c128 to it's fast mode of operation.
if the 40 column vic screen is not being used then this
mode may be selected for greater processing speed.

caveats: if fast invoked while the user is at the console and
using the 40 column screen, the screen will go blank.

examples: fast

files: none

see also: slow, 40, 80

routines: none

command: find,grep

abstract: find a string of characters in a file

usage: find pattern [d:]file ...

description: find looks for a string of characters in a file
and prints out the line # and the line itself if
it is found. it will do this for all lines that
contain the pattern. find expects the lines of the file
to be delimited by carriage returns (normal abbs format).

caveats: if the pattern is split between line it will not be found.
if lines of the file are too long for find to handle then it
will split the line in two rather than abort.

examples: find "here is a text string" testfile
find "subject" b:mail

files: *

command: help,?

abstract: display help files for selected subjects

usage: help subject ...

description: the help command displays the contents of a help file
to the terminal. the help files used are selected
by what subjects are specified on the command line.
help may be obtained for commands, support files,
script files, routines and miscellaneous subjects.
for a list of subjects available select subjects as the
subject for help. refer to the example.

caveats: none known

examples:

```
help subjects
help quick
? help
? mail
help q*
```

files: hlp/*

see also: *

routines: help, type

command: input

abstract: input a line from terminal/keyboard

usage: input [-p] [-c]

description: input will read one line (terminated by a <cr>) from
the keyboard or terminal and make the text of that
line available with the percent sequence %!.
if no options are specified then no conversion is done
to the text entered.
the -p causes input to parse the line immediately as is
the command line. ie: percent sequences are expanded.
the -c option causes input to convert the text to single
case, upper case ascii.

caveats: if -p is used without -c then percent sequences entered
must be in upper case ascii.

examples: input
input -p
input -c -p

see also: percent

command: log

abstract: send command line arguments to log file

usage: log [string] ...

description: log is the same as the echo command but instead
of sending output to the screen it appends it's output
to the file etc/log.
refer to the echo command

caveats: the file etc/log must exist
refer to the echo command

examples: log "the value of the loops is: "%p%r
log "pot 3 is: "%3%r

files: etc/log

see also: echo, print

command: loop

abstract: show loop and pot status for number of times

usage: loop [number]

description: loop is a command intended for use in debugging
alarm loop/pot hardware. it will continuously print
out the status of the alarm loops and pots for the
number of times specified. the maximum number of times
is limited depended on a users access level.

caveats: loop does not stop the keyboard timeout timer.
if loop is given a very large number then the system
may timeout due to keyboard inactivity when it completes.

examples: loop 1000

command: dir,ls

abstract: list file names

usage: <dir,ls> [d:]pattern ...

description: dir will show the file names of the current directory.
wild card characters are passed to the cbm dos.
if the directory contains wild card characters then
the cbm dos will use them.

caveats: if dir is used in a submitted script, the script will
abort when ls is done.

examples: dir a: b:
ls xx*

see also: cd

command: mail

abstract: read own mail or send mail to a valid user

usage: mail [user subject file]

description: if no arguments are specified mail invokes the editor and reads the users mail file into the editors buffer.

to send mail a valid user name must be specified as the first argument, an arbitrary string of characters must be specified as the subject and lastly, the file to be sent to the user specified.
the users person mail file must have an appropriate write level in order to be able to receive mail from others.

caveats: the user must in his/her private directory of the systems mail disk drive in order to read or send mail.
it is faster to read ones mail by using the cat command to send the contents of the file to the terminal.

examples: mail sys how-do-i myfile
mail sys "login request" b:my-info

files: ???/mail

see also: ed, user, read, write,

command: mode

abstract: set read and write level

usage: mode [read-level write-level]

description: mode sets both the read and write level at the same time. refer to the manual pages of the read and write commands for their descriptions

caveats: this is a redundant command.

examples:
```
mode 3 4   - set read to 3, write to 4
mode       - display current modes
```

see also: read,write, ed, uload

command: <mv,rename>

abstract: move a file

usage: <mv,rename> [d:]source [d:]destination [<[-s],-u,-p>]

description: mv is for moving files. mv is the same as cp except the source file is removed after the file is copied to it's destination. see the description for copy for more information on mv.

caveats: mv does not work well in a submitted script when it operates of files of the same drive the script is run from.

examples:
```
mv b:mail b:old-mail
mv a:etc/ulog a:etc/o-ulog
mv a:etc/sprite b:etc/sprite -p
```

files: *

see also: cp

command: nobreak

abstract: do not allow a submitted script to be interrupted

usage: nobreak

description: nobreak set the break flag to off so that a submitted script can not be stopped by sending a control-c or delete.

caveats: individual commands can still be stopped. if a command such is type used in a script with nobreak then sending control-c or delete will stop the type command and not the script.

examples: nobreak

see also: breakon

command: play

abstract: play a tune

usage: play notes

description: play checks each character of its arguments and
checks to see if they are valid characters for the
c128 basic 7.0 play command. if an invalid character is
found it is removed. the characters are then passed to
the c128 basic play command.
valid characters are:

0123456789 abcdefg #$rm vut ??????????????????

caveats: the interrupt routine is disable when the system is
playing music. alarm loops and pots will not be
monitored until play is done.

examples: play abcdefg
play "v1av2cv3e"

files: sys/mary
sys/row

see also: cbm basic 7.0 (C128) manual page for play command

command: port

abstract: read or write port or memory location.

usage: port [<rea[d],wri[te],and,or,xor>] [byte] [address] [bank]

description:

port reads or writes to memory locations of different banks. the byte argument is not used during a read but must be specified if an address is specified.
the default bank is bank 15. the default address is the location for joy port two.

read display the contents of the memory location.
write writes the value of byte to the memory location.
and writes the byte to the location after it is logically "anded" with the byte read from the location.
or writes the byte to the location after it is logically "ored" with the byte read from the location.
xor writes the byte to the location after it is logically "xored" with the byte read from the location.

caveats: writes and bank switches may crash the system.

examples: port read
port write 255

command: print

abstract: send command line arguments to printer.

usage: print [string] ...

description: print is the same as the echo command but instead
of sending output to the screen it attempts to send
it to the printer. if the printer is not on-line then
it's output may go to the console screen.

caveats: refer to the echo command

examples: print "the value of the loops is: "%p%n
print "pot 3 is: "%3%n

see also: echo, log

command: read

abstract: set read level

usage: read [level]

description: read set the read level that will be used when writing
a file. this number (from 0 to 4) will be used when
ever a file is written with a-bbs commands such as ed
and uload. when another user attempts to read the file
(assuming it is placed in a public directory) then this
number will be compared to the users access level. if the
other users access level is less than the files read level
then that user will be denied read access. level 4 read
access denies read access to all users, including the
system administrator.

caveats: a clever enough system administrator can find a way to read
files with level 4 read access. a non a-bbs program will
probably not deny access.
the dos command does not check read/write access levels.
once the level has been set and the file written. the only
way to change it is to edit the file and re-write it or use
another program that allows disk editing.
the owner of a file is never denied read access.

examples: read <0,1> <- anyone may read.
read 2 <- level two and above users may read.
read 3 <- level three and above users may read.
read 4 <- only system administrators may read.
read 5 <- only the owner of the file may read.

files: *

see also: write, mode, ed, uload

command: rerun

abstract: rerun the a-bbs system

usage: rerun

description: rerun will re-run the copy of the executable currently in memory.

caveats: the user of rerun will be logged off or disconnected from the tty

examples: rerun

see also: exec, restart

command: restart

abstract: restart the a-bbs system

usage: restart

description: restart will load the memory with a fresh copy of the a-bbs system executable. (p-a-bbs*) and run it.

caveats: the user of restart will be logged off or disconnected from the tty

examples: restart

see also: exec, rerun

command: set, clear

abstract: set environment variables

usage: set variable [index] contents ...

description: set allows the system administrator to configure the system environment according to personal preference. it may be desirable to use scheduled scripts to reconfigure the system for different times of the day.

because many of the variables are intended to used as logical 1 or 0 values, set maps the english words "on" and "off" to the numbers "1" and "0".

for this system set and clear are synonyms. the reason is that set will set a variable to zero or null if it is not specified. this can be thought of as setting it to zero or null. clear is just the same. the reason this is done is not for the system but the user. sometimes it is easier to think of something being set to zero as being cleared. e.g. set verbose 1 will set the verbose flag to one. set verbose will set the verbose flag to zero. the second example may be confusing. clear verbose will set the verbose flag to zero which may be less confusing for some.

refer to the general description for variable names and their purpose.

caveats: when the exit delay is set for a particular loop, it is set for all loops.
if set is given an incomplete set of arguments the variables will most likely be set to zero by default. this may produce undesirable results ex. if a disk drive is set to 0 errors will occur.

examples:
```
set alarm 600        <- 600 seconds = 10 minutes
set monitor 0 on
set exit 60
set entry 60
set schedule 1 0 am a:sys/auto1 a:sys/auto2 b:tmp/junk2
```

see also: dump, general, monitor, alarm, verbose, terse, potmon

command: sleep

abstract: sleep for some seconds

usage: sleep seconds

description: sleep will cause the system to pause for
the number of seconds specified. the maximum
number for non level 3 users is 5 seconds.
numbers over the maximum will be taken as the maximum.

caveats: the interrupt pauses too, the alarm loops
will not be monitored while sleeping.

examples: sleep 5

routines: sleep

command: slow

abstract: set the cpu clock to it's slower speed of 1 mhz.

usage: slow

description: slow sets the c128 to it's slow mode of operation.
if the 40 column vic screen is to be used then this
mode must be selected in order for it to be seen.

caveats: the response time is degraded.

examples: slow

see also: fast, 40, 80

command: submit

abstract: submit a command file to be run as typed into the shell.

usage: submit [d:]file ...

description: submit reads the filenames specified on the command line and interprets the text as if it were commands typed into the shell. submit calls the same routines as the shell. submit checks too see if a script tries to call submit and blocks it. it also checks the keyboard or tty in case the user wishes to abort the script.

caveats: submit cannot submit itself
commands that use the disk drive error channel may cause submitted scripts to abort early. these commands include

ls
mv,cp (single drive copy/move)
dos

examples: submit myscript

files: etc/config
etc/profile
sys/profile
???/profile
etc/loop?
sys/free
other scripts

see also: config, profile, shell, set

command: terse

abstract: set the verbose mode to off

usage: terse

description: terse sets the verbose mode to off. the system will display a less verbose prompt and submitted files will not displayed as they are read.

caveats: none known

examples: terse

see also: verbose

command: time

abstract: set or display the time of day (am/pm)

usage: time [hhmmss <am,pm>]

description: with no arguments time displays the current time.
if it is not set date will display the message:
[tm$-not-set]. if it is then it will display the
hour minute sec tenth of second and whether it is
am or pm. the format is hh:mm:ss.tt ?m.

level three access is required to set the date.

time sets the tod clock of the c128's vic ii chip.
this hardware clock is more accurate than the software
clock and is not affected by the operating system.

caveats: although the tod clock is not affected by a system reset
the time must be set when system is boot. there is
no involitial memory store for the date.
the time reflects what the time was the last time
the clock routine was called. if time is entered
repeatedly the time may appear to remain unchanged.
the clock is actually updating in real time, it is
the string used to display the time that is not in
real time.

examples: time 034600 am <- will set the time of day

time
03:46:00.03 am <- will be displayed

see also: date, clock

command: <cat,type>

abstract: send the contents of a file to the screen

usage: <cat,type> [d:]file ...

description: cat opens the disk file(s) specified and sends the contents to the terminals screen. more than one file may be specified. the contents of the file is unaffected.

caveats: cat erroneously sends extra blank lines when its reaches the end of a file. this may be fixed in a later version.

files: *

examples: cat b:profile
type mail

command: uload

abstract: up-load a file

usage: uload [d:]file

description: uload opens a file for write (with header) and simply reads characters from the keyboard or tty until it sees a EOT character (control-d). each character read will be written to the file. uload also echos the the characters back.

caveats: if uload does not see any characters for the time specified as the login time (milliseconds) it will timeout, close the file and exit.

examples: uload mytext

files: *

see also: read, write

command: user

abstract: check for valid user password and/or display user data

description: the user routine searches for valid users and depending on the present access level, user name and password supplied it will do different things.

if access level is 0 then the user routine will only allow a user to login as a valid user. this is done at login or if the access level is set to zero after login.

if the access level is 1 or more the routine will allow the user to print data about all or any single user.
if a valid user name is supplied it will print data specific to that user. if none is supplied it will print data specific to all users.
if a valid user name and password combination is supplied it will allow the user to login as a different user.

caveats: when the user routine is used to login as a valid user and it is successful, it appends data into the file etc/ulog. no checking is done to see how large this file gets. refer to the manual page for ulog.

files: what files are read/write affected by this routine.

see also: ulog

command: verbose

abstract: set the verbose mode to on

usage: verbose

description: verbose set the verbose mode to on. the system will display a more verbose prompt and submitted files will displayed as they are read.

caveats: none known

examples: verbose

see also: verbose, set

command: write

abstract: set write level

usage: write [level]

description: write set the write level that will be used when writing a file. this number (from 0 to 4) will be used when ever a file is written with a-bbs commands such as ed and uload. when another user attempts to write the file (assuming it is placed in a public directory) then this number will be compared to the users access level. if the other users access level is less than the files write level then that user will be denied write access. level 4 write access denies write access to all users, including the system administrator.

caveats: a clever enough system administrator can find a way to write files with level 4 write access. a non a-bbs program will probably not deny access.
the dos command does not check read/write access levels.
once the level has been set and the file written. the only way to change it is to edit the file and re-write it or use another program that allows disk editing.
the owner of a file is never denied write access.
if another user is allowed write access then that user can take ownership of the file writing to it.

examples:
```
write <0,1> <- anyone may write.
write 2     <- level two and above users may write.
write 3     <- level three and above users may write.
write 4     <- only system administrators may write.
write 5     <- only the owner of the file may write.
```

files: *

see also: read, mode, ed, uload

command: xmit

abstract: xmodem download

usage: xmit [d:]file ...

description: xmit starts a standard xmodem transmit.
xmit will attempt to send a file to a remote
receiver using the xmodem checksum protocol.
if xmit recieves a CAN (control-X) it will abort.

caveats: not implemented yet

examples: xmit myprog

files: *

see also: xrcv, uload

command: xrcv

abstract: xmodem upload

usage: xrcv [d:]file ...

description: xrcv starts a standard xmodem receive. xrcv will attempt to recieve a file using the xmodem checksum protocol.

caveats: not-available yet

examples: xrcv myprog

files: *

see also: xmit, uload

routine: alarm

abstract: sound the alarms

description: alarm takes control of the system if the alarm
is armed, and the exit/entry delay ticks have counted
for a monitored loop that has been closed.
alarm will submit an alarm script for the loop that
has been closed. the scripts are:

loop	script
0	etc/loop0
1	etc/loop1
2	etc/loop2
3	etc/loop3

the system administrator may re-write these scripts
to suit the needs of the systems application.
after the script is submitted the alarm pokes the mask
byte to the output port.
this is done so that even if the disk drive fails
the output port can be affected. this byte will determine
which bits of the output port go to zero volts or
do not change.
alarm will then sleep for the number of seconds specified
by "set alarm". the port will then be restored to what
it was before.

caveats: if the scripts are non-existant an error will occur.
the scripts should not be removed. if they are not
needed then make them empty files.

files: etc/loop0
etc/loop1
etc/loop2
etc/loop3

see also: set, dump, submit

routine: clock

abstract: maintain systems time/day/month/year

description: the clock routine maintains the systems record of time of day, the day number, month name and current year (greater than 1989). clock uses the C128's hardware TOD clock as it's time base. this enables it to be unaffected by the C128's operating system and maintains reasonably accurate time. clock accounts for short months. the time must be set after the system is booted with the time command. clock will wish you a happy new year.

caveats: clock does not account for leap years.

see also: time, date

routine: monitor

abstract: check monitored loops for activity

level: none

description: the monitor routine runs is called by interrupt and
runs in the background if the alarm is armed.
it checks the status of the loops and sets the alarm flag
if a monitored loop is closed.

caveats: monitor will trigger alarm sequence for one loop only.
the priority is: 0=highest to 3=lowest

see also: set, alarm

routine: potmon

abstract: check pots for out of tolerance condition

description: the potmon routine runs is called by interrupt and
runs in the background.
it checks the status of the loops pots and submits a pot
script if the status goes above or below the maximum/minimum
values specified. if a user is logged in it will wait for that
user to exit. potmon will continue to re-submit the script if
the pot remains out of tolerance. it will do this until the
variable count (which is incremented by potmon) is equal to
or greater than the variable quit.

caveats: the alarm system will override potmon, if a user is
logged in potmon will not submit the pot script.
if a scheduled script is running potmon will not run
a pot script.

files: alm/pot0 <- script for pot zero
alm/pot1 <- script for pot one
alm/pot2 <- script for pot two
alm/pot3 <- script for pot three

see also: set, dump, submit

routine: shell

abstract: the command line interface

description: the shell is the routine that prints the prompt,
then calls the following routines:
input - get a line of input from the keyboard or terminal
find-args - parse line to find arguments, expand percent sequences.
find-dev - parse arguments to find associated devices if any.
next it attempts to call the command specified by the first argument, argument number zero. when the command completes the process starts over.

in a nutshell, no pun intended, the shell is doesn't do much by itself. it call other routines to do the work. it is what the user "sees", "the executive routine".

caveats: all commands are memory resident. at this time there are no transient commands.

see also: input, submit

file: a-bbs

abstract: the main executable for a-bbs system

description: a-bbs is an acronym for "alarm bulletin board system".
a-bbs and it's support files provide the owner with software for a bulletin board program and a home security system.

the alarm system runs in the background while the bulletin board system acts as the interface to the user.
the alarm is configured via the bulletin board system.
it is armed with a toggle switch that the operator must connect to joystick port #1. this port is where all four alarm loops must be connected.
refer to the hardware description for more detail.

the bulletin board system is basically a command line driven shell where the user types in a command line which is parsed and causes the system to perform a function.
the bulletin board system support multiple users (not at the same time) multi-level access and a simple mechanism for file security.

the user may access the bulletin board system in one of two ways. the keyboard/screen or the tty (rs232 port).
access through the tty is expected to be through a commodore 1670 1200 baud modem. a-bbs may work with other modems (including a null modem) but as of this writing such configurations are untested.

caveats: the home security system routines run in the back-ground and don't interfere with the bulletin board system during normal operation however, when the security system sounds an alarm the bulletin board system is disabled for some time until it is done. this is by design since it is felt that an alarm condition should take priority over bulletin board operation.

hopefully this will never happen since it will probably be caused by a false alarm (annoying the neighbors) or a burglary/fire in your home.

examples: while in native commodore 128 mode:
run "p-a-bbs*" :rem refer to hardware description

file: config

abstract: set system configuration a boot time

description: config is a script that is submitted when the system
is booted. the system administrator may put any commands
in that can be submitted. one use might be to configure the
alarm system and the systems general hardware setup.
this will allow the system to recover itself in the event
of an unexpected power failure or temporary shut down.

caveats: if this file does not exist an erro will occur

examples: A sample configuration script may look like this:

```
#@(1)#
# etc/config, sys, 2 , 2 ,jan: 1: 1989, 12:01:00.00 AM
terse
echo "etc/config: time and date may not be set"%n
echo "current time: "%t%n
echo "current date: "%d%n
set printer off
set mon 0 on
set exit 0 5
set entry 0 10
#eof
```

files: etc/config

see also: submit, profile

file: passwd

abstract: password file

description: the passwd file "etc/passwd" is the file where all valid user names, passwords, access levels and other data are contained.

it is plain sequential file that can be modified easily with the editor. the system administrator should be aware that entrys made here should be consistent reguarding upper/lower case. the convention suggested is all lower case (commodore ascii). or all upper case, real ascii. this will make it necessary for a user logging in via the tty to use upper case characters when typing in user/password.

user names should be kept to three characters in length. the login routine truncates user names longer than three characters. the suggested convention here is the persons initials. passwords may be any length that is not ridiculous.

the structure for the data is (fields delineated by commas):

user,password,level,street,city,state,zip,voice-phone, data-phone,comments/memo

the data for each user must be on one line.

caveats: the larger this file gets the longer it will take those at the end of the file to log-on. this is due to it being a sequential verses random-access file.
remember to remove old file headers before re-writing this file. more than one header may cause the user routine to skip the first user entry.

examples:

```
#@(1)#
# etc/passwd, sys, 4 , 4 ,jan: 1: 1989,12:01:00.00 am
sys,system,3,no-street,no-city,no-state,no-zip,phone1,phone2,memo
awg,the-author,3,p.o.box-13,newton-jct,nh,03859,555-1212,603-382-3966,me
pam,peter-a-moss,2,main-st,anytown,usa,00001,555-1212,555-2121,a-friend
```

see also: user, ed, type

file: profile

abstract: set system/user environment

description: there are two types of profiles, system and user.
the system profile is the file called etc/profile
and the users profile will be the file named profile
in a users private directory.
these files are scripts that will be submitted when a
user logs in. the system profile has an access level of
three and the users profile an access level of whatever
is specified in etc/passwd. these files may contain any
commands that can be submitted.

caveats: if this file does not exist an erro will occur

examples: A sample profile may look like this:

```
#@(1)#
# sys/profile, sys, 2 , 2 ,jan: 1: 1989, 12:01:00.00 AM
terse
read 1
write 3
echo "today is "%d" the time is "%t%n
echo "sys/profile complete !"%n
#eof
```

files: etc/profile
???/profile

see also: submit, config

file: ulog

abstract: log of successful login attempts

description: ulog is a data file of successful login attempts.
each time the user routine accepts a valid user/password
combination the file is appended with user, level, date
and time information.

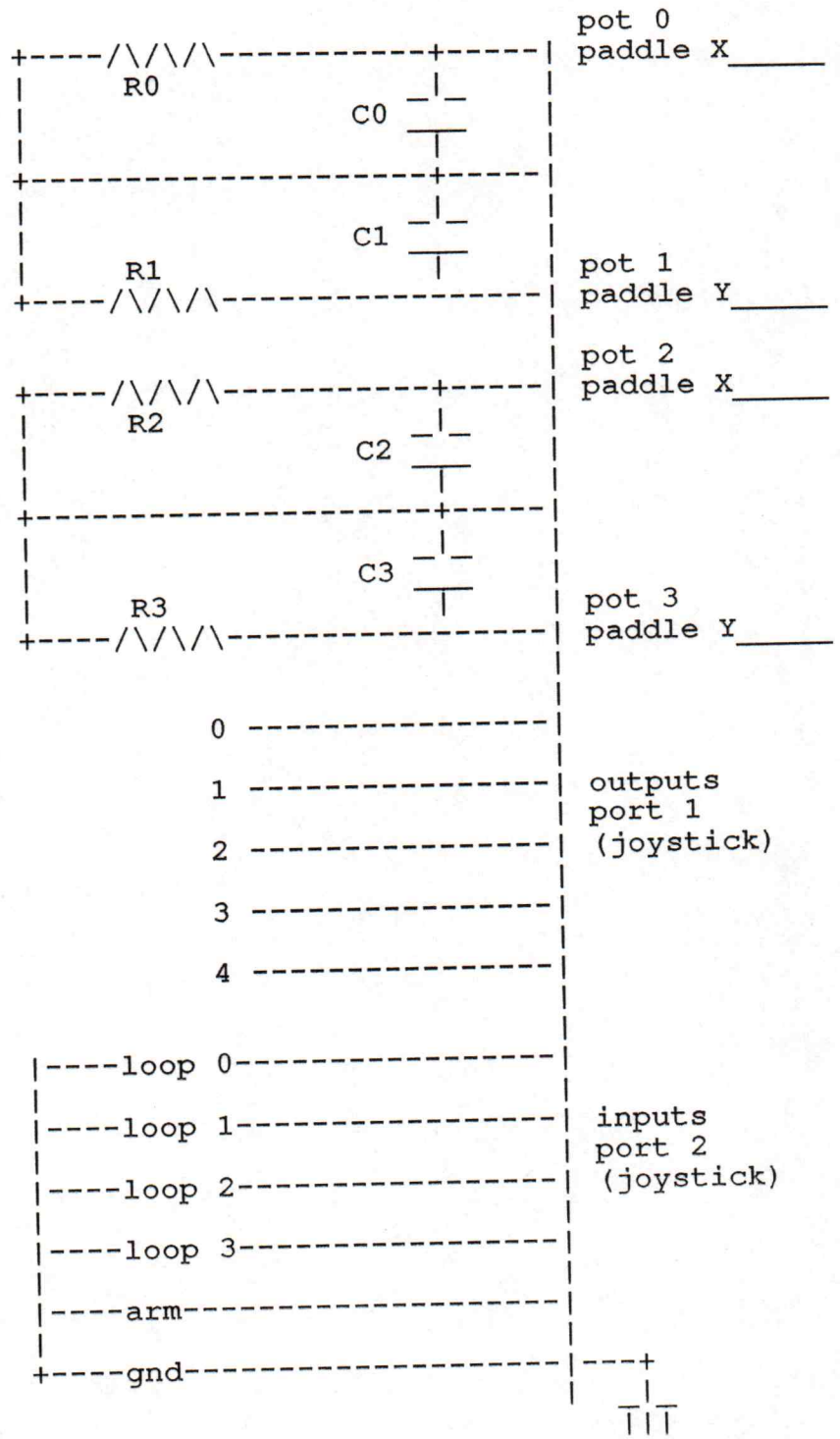
caveats: the user routine simply appends the file. no checking
is done to see how large it gets. if left unchecked it
will grow as large as the disk drive can handle.
this is probably when the diskette is full. it is up to
system administrator to periodically downsize it by editing
it, moving or removing it.
one suggestion would be to use a scheduled script to
move it to a file named o-ulog on a daily basis.
this will clear it out yet allow the system administrator
to monitor current daily system usage.
this file must exist.

examples:
a typical entry may be similar to the text below.

user	level	date	time
sys	3	jan: 1: 1989	03:46:55.03 pm

see also: user, ed, type

Hardware:
This page shows a hardware configuration that can be used for monitoring external devices.



loop 0-3 and arm are normally open. They can be closed with

switches, relays or by some other means.

to use the analog "pot" inputs as tempature the schematic above and the following values are suggested.

R0-R3 _____

C0-C3 _____

note it will be neccessary to adjust set each pots slope and bias values. the following values are suggested.

slope:_____

bias :_____

A-BBS
Version 1.00 r1
general reference:

---

documentation notation:
1) items enclosed by [] are optional

2) items enclosed by <> and separated by commas are possible choices,
   one of which may be chosen

3) items not enclosed by [] or <> are required

4) "..." indicates that previous argument type may be repeated.

---

command line prompt description:
d:dir/>
d:=current drive, dir=current directory

---

command line input:

1) printable ascii characters will be translated to upper case.

2) cbm lower case will appear as ascii upper case on on the tty.
   ascii lower case will be converted to upper case.

3) control-c or delete/rubout will abort command line input and most
   commands. this is ignored if the break flag has not been set. see
   breakon.

4) backspace will delete characters input to the left of the cursor.

5) carriage return signals the end of input, the line will then be parsed.

6) the percent character "%" tells the parser to interpret the next
   character differently, refer to section on "percent sequences".

7) text enclosed by double quotes will be parsed literally, i.e.
   % sequences will not be expanded, spaces will not delineate
   arguments, there will be no case conversion.

---

file header description:
when a file is created with ed or uload, the first two lines of the file
is a header. this header contains information that the system will
read when other commands are used to operate on that file.
following is an example:

```
#@(1)#
# gue/sample,gue, 0 , 5 ,jan: 1: 1989,04:43:03.06:pm
```

the first line is the "signature", this tells the system that the file
has a recognizable header.
the second line is where the files attributes are stored.
first is the file name. second is the name of the user that "owns"
the file. third is the read level, fourth is the write level.

W E L C O M E W E L C O M E W E L C O M E W E L C O M E

I would like to invite you to log in to my BBS, the ==info=rail=.
it is located in newton-junction new hampshire and is available
24 hours per day (except when updating). New users may log in as
a guest. The guest login will run a script which that will ask you
some questions in order to obtain a private login. The questions are
not too personal and will be used to keep track of who you are and what
city you live in. The ==info=rail== is currently a no charge bbs for
the purpose of discussion about any topic. There is no file
upload/download capabilty except for simple ascii file transfers.
Users may send private mail to one another or post articles (as files)
in the bulletin areas. There is a simple file security mechanism
implemented so users may protect their files from un authorized reading
or writing.

To login, use your computer or terminal w/modem to call:

(603) 382-3966

At the LOGIN: prompt type: GUE
At the PASSWORD: prompt type: *GUEST*
(all upper case for login/password)

Wait for the introductory boiler-plate and answer the questions
when prompted. The user name convention at the ==info=rail= is
to use a persons inititals (2-3). If there already is a user with
the same initials then the convention is to use your first and last
initials with a single digit number suffixed to them. For example.

Let's say three users apply, their names are:

Bob R. Jones
Herman Y. Xylogathon
Bill R. James

Since Bob is the first one to apply with the initials brj his user name
will be brj. Since Hermans is the first (and probably last) with the
initials hyx, his user name is hyx. Because Bill has the same initials
as Bob, his user name can't be brj, it will be bj1. Now if another
user named Bo R. Joy applys, his user name will be bj2 and so on...

You may choose any string of upper case ascii (or symbols) four to eight
characters in length for your password. The longer the better. This will
make it more difficult for someone to break in to the system as you.

W E L C O M E W E L C O M E W E L C O M E W E L C O M E

command:

abstract:

usage:

description:

caveats:

examples:

files:

see also: