;
;
;CPUT "@:SYSTEMDEFS.USER
;### C64 & C128 DEFS ###
;
;SYSTEMDEFS FOR USE WITH THE LT. KERNAL ROUTINES
;
;
;MISC. HOST ADAPTER BUFFERS AND WORK ARES
;
CMDBUF=$8DB6 ;LOCATION OF COMMAND CHANNEL BUFFER
WRTBUF=$8DE0 ;LOCATION OF FILE WRITE BUFFER
WORKAR=$8FE0 ;MISC WORK AREA
HDRBLK=$91E0 ;FILE HEADER BLOCK WORK AREA
MINSUB=$93E0 ;MINI-SUB PROGRAM EXECUTION AREA
DOSOVL=$95E0 ;DOS OVERLAY AREA FOR SYSTEM PROCESSORS AND RUN-TIME MODULES
REDBUF=$9BE0 ;LOCATION OF FILE READ BUFFER
FPTTAB=$9DE0 ;LOCATION OF THE FILE PARAMETER TABLE (FPT)
MESBUF=$9EE0 ;ERROR CHANNEL MESSAGE BUFFER
DIRBUF=$9FE0 ;DIRECTORY ($) PATTERN MATCH BUFFER (32 BYTES LONG)
;
;
;FILE HEADER BLOCK OFFSET DEFINITIONS
;
FILNAM=0 ;FILENAME
NBINFL=$10 ;NUMBER OF BLOCKS IN FILE (INCLUDING HEADER)
NRPBLK=$12 ;NUMBER OF RECORDS PER BLOCK
NBPREC=$14 ;NUMBER OF BYTES PER RECORD
NRINFL=$16 ;NUMBER OF RECORDS IN FILE
FILTYP=$18 ;FILE TYPE CODE (SEE TABLE BELOW
NBLKPC=$19 ;NUMBER OF BLOCKS PER CYLINDER (BITMAP ONLY)
;FOR OTHER TYPE FILES, "NBLKPC" CONTAINS THE HI ORDER BIT OF THE
;# OF BYTES IN LAST BLOCK COUNTER (LO ORDER IS 'NBYTLB').
;THIS BIT IS KEPT IN THE LEAST SIG. BIT
LOADAD=$1A ;LOAD ADDRESS OF THIS FILE
NBYTLB=$1C ;# OF BYTES IN LAST BLOCK (LO ORDER PART)
USRLUN=$1D ;USER/LOGICAL UNIT
HDRFLG=$1E ;HEADER BLOCK ACTIVE FLAG ($AC OR $AF = ACTIVE HEADER)
BLMILO=$20 ;BLOCK LIST (MIDDLE & LOW ORDER BYTES)
BLKAVL=$90 ;START OF A 3 PREC. # OF BLOCKS AVAILABLE COUNTER (MAP ONLY)
BLKUSD=$93 ;START OF A 3 PREC. # OF BLOCKS USED COUNTER (MAP ONLY)
ALTLOD=NRPBLK ;ALTERNATE LOAD ADDRESS (USED BY SPECIAL PROC. ONLY, TYPE 3)
;
;
;THE FOLLOWING ARE FILE TYPE EQUATES
;
;CONTIGUOUS TYPE FILES
;
;
SYSFIL=1 ;CONTIGUOUS DOS SYSTEM FILE (EX. DISCMAP & INDEX)
PRSFIL=2 ;CONTIGUOUS DOS PROCESSOR FILE (EX. SAVE,DIR,DEL ETC.)
SPRFIL=3 ;SPECIAL DOS PROCESSOR, RUNS OUTSIDE DOSOVL AREA (VIA SWAPPER)

```
INDFIL=4 ;MULTIPLE DIRECTORY INDEX FILE
CNTFIL=5 ;CONTIGUOUS DATA FILE
;




;
;RANDOM BLOCK LIST TYPE FILES
;
;
BASFIL=11 ;BASIC LANGUAGE PROGRAM FILE
RNDFIL=12 ;RANDOM M.L. FILE
SEQFIL=13 ;RANDOM SEQUENTIAL FILE (EX. TEXT FILES)
USRFIL=14 ;USER FILE
RELFIL=15 ;RELATIVE FILE
RANFIL=16 ;RANDOM FILE (#) - ONE PER LU ALLOWED
;
;
;MISC. SYSTEM VECTORS AND CONTROL LOCATIONS
;
BNKOUT=$FC4E ;ADDRESS OF THE KERNAL BANK-OUT ROUTINE IN SHADOW RAM
BNKSWT=$FC5F ;KERTRAP BANK CONTROL SWITCH
KEYENB=$FC60 ;NUMERIC KEYPAD ENABLE FLAG ($FF=ENABLED 0=DISABLED)
BANKIN=$FC71 ;KERTRAP BANK IN ROUTINE
BASEXT=$FC74 ;BASIC EXTENSIONS TRAP VECTOR (EX. KEY FILE ROUTINES)
GO64MD=$FC7A ;GO 64 MODE ENTRY POINT (USED BY C128 MODE)
IDX64=$C8 ;INDEX TO CURRENT POSITION IN THE BASIC INPUT BUFFER @ $0200
IDX128=$EA ;INDEX TO CURRENT POSITION IN THE BASIC INPUT BUFFER @ $0200
ERRCHN=$E0 ;ERROR CHANNEL FPT OFFSET
BUF=$0200 ;COMMAND BUFFER USED BY EXEC & PROCESSORS
;
;
;
;THE FOLLOWING ARE SYSTEM VARIABLE DEFINITIONS
;
;
SYSVAR=$8000 ;LOCATION OF SYSTEM VARIABLES
;
ACTIVL=SYSVAR ;CURRENT ACTIVE LOGICAL UNIT
ACTIVU=SYSVAR+1 ;CURRENT ACTIVE USER
ORIGCR=SYSVAR+5 ;ORIGINAL C.R. SAVED ON INITIAL ENTRY FROM A TRAP *
UNUSD1=SYSVAR+6 ;CURRENTLY UNUSED BYTE (RESERVED FOR FUTURE USE)
DRVSWT=SYSVAR+7 ;DISK DRIVER READ/WRITE LOOP CONTROL SWITCH
MODESW=SYSVAR+8 ;CPU MODE SWITCH (0=C64  <>0=C128) *
BASVEC=SYSVAR+9 ;BASIC EXTENSIONS VECTOR
KERRTN=SYSVAR+11 ;RETURN VECTOR USED WHEN CALLING EXTERNAL ROUTINES *
LKWEDG=SYSVAR+13 ;VECTOR TO LT KERNAL BASIN WEDGE *
GOTO64=SYSVAR+35 ;VEXTOR TO THE GO C64 ROUTINE
SAVNAM=SYSVAR+37 ;FILE'S LU# & HDR. BLK. ADR. - USED FOR SAVE & REPLACE
LKRNUM=SYSVAR+40 ;CURRENT KERNAL ROUTINE CONTINUATION NUMBER *
BEEPFL=SYSVAR+41 ;BEEP ON ERROR FLAG (0=NO BEEP)
HRDNUM=SYSVAR+42 ;LOCATION OF HARD DRIVE DEVICE NUMBER
ERTRPF=SYSVAR+43 ;LOCATION OF ERROR TRAP FLAG *
ORIGA=SYSVAR+44 ;SAVE LOCATION OF 'A' FOR ALL 'LK' TRAPS
ORIGX=SYSVAR+45 ;SAVE LOCATION OF 'X' FOR ALL 'LK' TRAPS
ORIGY=SYSVAR+46 ;SAVE LOCATION OF 'Y' FOR ALL 'LK' TRAPS
ORIGP=SYSVAR+47 ;SAVE LOCATION OF 'P' FOR ALL 'LK' TRAPS
AUTBOT=SYSVAR+48 ;AUTO BOOT FLAG *
CRDSOV=SYSVAR+49 ;BLK. ADR. OF CURRENT DOS OVERLAY
CRMINS=SYSVAR+50 ;BLK. ADR. OF CURRENT MINI-SUB
```

```
CNTR1=SYSVAR+51 ;OFFSET COUNTER USED FOR COMMAND TAIL PROCESSING
SVPCRC=SYSVAR+52 ;TEMP. STORAGE FOR PRECONFIGURATION REGISTER 'C' *
SVPCRD=SYSVAR+53 ;TEMP. STORAGE FOR PRECONFIGURATION REGISTER 'D' *
REDCHN=SYSVAR+54 ;CURRENT READ CHANNEL FPT POINTER
WRTCHN=SYSVAR+55 ;CURRENT WRITE CHANNEL FPT POINTER
CPUSPD=SYSVAR+56 ;DEFAULT CPU SPEED (0=1MHZ  1=2MHZ) *
;
;




;* THESE LOCATIONS MAY BE USED FOR REFERENCE PURPOSES ONLY.
;  THEY SHOULD NEVER BE DIRECTLY MODIFIED BY A USER WRITTEN ROUTINE !!
;
;
;THE FOLLOWING ARE THE LKDOS JUMP TABLE SUBROUTINE EQUATES:
;
;
JUMPTB=SYSVAR+57 ;ADDRESS OF THE L.K. JUMP TABLE
;
;
KURSET=JUMPTB ;KERNAL CALL SETUP FOR USE TRAPPED KERNAL ROUTINES
KURST2=KURSET+3 ;KERNAL CALL SETUP FOR USE OF NON-TRAPPED KERNAL ROUTINES
KERCAL=KURST2+3 ;KERNAL CALLING ROUTINE
REDFIL=KERCAL+3 ;READ FILE ENTRY FOR AUTO-BOOT SEQUENCE **
DRIVER=REDFIL+3 ;HARD DRIVE DISC DRIVER ROUTINE (FOR READS & WRITES)
OUTPUT=DRIVER+3 ;CHARACTER OUTPUT ROUTINE
FNFILE=OUTPUT+3 ;FIND FILE ROUTINE
LODRND=FNFILE+3 ;LOAD RANDOM BLOCK LIST TYPE FILE
ERHAND=LODRND+3 ;ERROR HANDLER ROUTINE
CHEKDV=ERHAND+3 ;CHECK FOR HARD DISK DEVICE NUMBER ROUTINE **
SISRT1=CHEKDV+3 ;SYSTEM RETURN - 'RTS' WITH CURRENT REGISTERS AS IS **
SISRT2=SISRT1+3 ;SYSTEM RETURN - 'RTS' WITH ORIGINAL REGISTERS **
SISRT3=SISRT2+3 ;SYSTEM RETURN - 'VIA LKRTNM' WITH REGISTERS AS IS **
SISRT4=SISRT3+3 ;SYSTEM RETURN - 'VIA LKRTNM' WITH REGISTERS **
SISRT5=SISRT4+3 ;SYSTEM RETURN - 'ABS JMP' WITH REGISTERS AS ON ENTRY **
SAVRGS=SISRT5+3 ;REGISTER SAVE ROUTINE **
LODRGS=SAVRGS+3 ;REGISTER LOAD ROUTINE **
CLRHDR=LODRGS+3 ;'HDRBLK' AREA CLEARING ROUTINE
DOSRET=CLRHDR+3 ;DOSWEDGE RETURNS HERE IF ANOTHER DOS OVLY IS CALLED **
MLTPLY=DOSRET+3 ;TRIPLE PRECISION MULTIPLY ROUTINE
KRCAL2=MLTPLY+3 ;A KERCAL FOR USE BY TYPE 3 TRAPPED CALLS **
ALRAND=KRCAL2+3 ;ALOCATE RANDOM BLOCKS
ALCONT=ALRAND+3 ;ALOCATE CONTIGUOUS BLOCKS
APBLOK=ALCONT+3 ;APPEND BLOCK(S) TO FILE
DEALRN=APBLOK+3 ;DEALLOCATE BLOCKS OF A RANDOM TYPE FILE
DEALCN=DEALRN+3 ;DEALLOCATE BLOCKS OF A CONTIGUOUS TYPE FILE
MLRTRN=DEALCN+3 ;MACHINE LANGUAGE RETURN **
LODCON=MLRTRN+3 ;LOAD CONTIGUOUS TYPE FILE
COMCHN=LODCON+3 ;COMMAND CHANNEL PROCESSOR **
DIRECT=COMCHN+3 ;DISK DIRECTORY PROCESSOR **
DOSEXT=DIRECT+3 ;ENTRY POINT FOR CALLING AN EXTENDED DOS OVERLAY
SWPOUT=DOSEXT+3 ;MEMORY/DISK SWAPPER ROUTINE
SETLUN=SWPOUT+3 ;SET AN LU ACTIVE
MNSEXT=SETLUN+3 ;ENTRY POINT FOR CALLING AN EXTENDED MINI-SUB
PERCMD=MNSEXT+3 ;ENTRY POINT FOR COMMAND CHANNEL POSITION COMMAND **
SWPWRB=PERCMD+3 ;ENTRY POINT FOR SWAP 'WRITE' BUFFER ROUTINE **
FATERR=SWPWRB+3 ;ENTRY POINT FOR FATAL ERROR HANDLER ROUTINE
GTPORT=$9F03 ;ADDRESS OF THE GET PORT NUMBER ROUTINE
;
```

```
;
;** THESE ROUTINES ARE USED ONLY BY SPECIAL RUN-TIME MODULES.
;   THEY MUST NEVER BE CALLED BY A USER WRITTEN ROUTINE !!
;
;
;
;
LTABLE=$80A8 ;ADDRESS OF RAM RESIDENT LU PARAMETER TABLE
;
;
;
;



;LU TABLE ENTRIES CONSIST OF 5 BYTES EACH AND ARE DEFINED AS FOLLOWS:
;
;(BYTE 0)
;
;BIT(S)        USE
;--------------------------------
; 7----------ACTIVE LU FLAG (0=ACTIVE)
; 6,5--------PHYSICAL DRIVE NUMBER (0-3)
; 4,3,2------PHYSICAL CONTROLLER NUMBER (0-7)
; 1,0--------BEGINNING CYLINDER ADDRESS (HI ORDER)
;
;(BYTE 1)
;
; BEGINNING CYLINDER ADDRESS (LOW ORDER)
;
;(BYTE 2)
;
;BIT(S)        USE
;--------------------------------
; 7,6,5,4----NUMBER OF PHYSICAL HEADS (0-15)
; 3----------DOS IMAGE FILE FLAG (1=DOS IMAGE DOES EXIST ON THIS LU)
; 2,1,0------NUMBER OF CYLINDERS (HI ORDER)
;
;(BYTE 3)
;
; NUMBER OF CYLINDERS (LOW ORDER)
;
;(BYTE 4)
;
; NUMBER OF SECTORS PER TRACK
;
;
;
;THE FOLLOWING ARE KERNAL ROUTINE EQUATES
;THEY ARE USED TO GET TO NORMAL KERNAL ROUTINES VIA A CONTINUATION
;VECTOR.  THIS IS NECESSARY FOR USE OF THOSE KERNAL ROUTINES TRAPED
;BY THE HOST ADAPTER LOGIC.
;REFER TO "KURSET & KERCAL" CALLING PROCEDURES
;
;
LKOPEN=0 ;KERNAL OPEN ROUTINE
LKCLOZ=LKOPEN+2 ;KERNAL CLOSE ROUTINE
LKCHIN=LKCLOZ+2 ;KERNAL OPEN CHANNEL FOR INPUT ROUTINE
LKCHOT=LKCHIN+2 ;KERNAL OPEN CHANNEL FOR OUTPUT ROUTINE
```

```
LKCLCH=LKCHOT+2 ;KERNAL CLOSE INPUT & OUTPUT CHANNELS ROUTINE
LKCHRI=LKCLCH+2 ;KERNAL INPUT CHARACTER FROM CHANNEL ROUTINE
LKCHRO=LKCHRI+2 ;KERNAL OUTPUT CHARACTER TO CHANNEL ROUTINE
LKGTIN=LKCHRO+4 ;KERNAL GET CHARACTER ROUTINE
LKCLAL=LKGTIN+2 ;KERNAL CLOSE ALL FILES & CHANNELS ROUTINE
LKLOAD=LKCLAL+2 ;KERNAL LOAD RAM ROUTINE
LKSAVE=LKLOAD+2 ;KERNAL SAVE RAM ROUTINE
;
;
;
CATLOG=$24 ;BLOCK CONTAINING INDEX CATALOG ROUTINE
;
;THE FOLLOWING ARE ERROR CHANNEL MESSAGE FILE DEFINITIONS:
;REFER TO THE "ERROR HANDLER" CALLING PROCEDURES
;
;
DOSM00=00 ;00,OK,00,00
DOSM01=01 ;01,FILES SCRATCHED,00,00




DOSM02=02 ;01,FILES SCRATCHED,01,00
DOSM30=30 ;30,SYNTAX ERROR,00,00
DOSM31=31 ;31,SYNTAX ERROR,00,00
DOSM33=33 ;33,ILLEGAL FILENAME,00,00
DOSM50=50 ;50,RECORD NOT PRESENT,00,00
DOSM51=51 ;51,OVERFLOW IN RECORD
DOSM52=52 ;52,FILE TOO LARGE,00,00
DOSM60=60 ;60,FILE OPEN FOR WRITE,00,00
DOSM61=61 ;61,FILE NOT OPEN,00,00
DOSM62=62 ;62,FILE NOT FOUND,00,00
DOSM63=63 ;63,FILE EXISTS,00,00
DOSM64=64 ;64,FILE TYPE MISMATCH,00,00
DOSM65=65 ;65,NO BLOCK,00,00
DOSM70=70 ;70,NO CHANNEL AVAILABLE,00,00
DOSM72=72 ;72,DISK/DIRECTORY FULL,00,00
DOSM73=73 ;73,POWER UP MESSAGE
;
;
;
;
;
;
;
;THE FOLLOWING PARMS. DEFINE THE LOCATIONS OF SYSTEM FILES
;
PARMFL=$029E ;ADDRESS OF THE PORT PARAMETERS FILE(1 BLOCK PER PORT)
SWAPFL=$02AE ;ADDRESS OF THE PORT SWAPFILE (8 BLOCKS PER PORT)
;
;
;
;THE FOLOWING ARE OFFSET DEFS. FOR EACH PORT'S PARAMETER BLOCK OF THE
;SYSTEM CONFIG FILE :
;
;
PBC064=0 ;BORDER COLOR (C64)
PSC064=PBC064+1 ;SCREEN COLOR (C64)
PCC064=PSC064+1 ;CURSOR COLOR (C64)
PBC428=PCC064+1 ;BORDER COLOR (C128/40COL)
```

```
PSC428=PBC428+1 ;SCREEN COLOR (C128/40COL)
PCC428=PSC428+1 ;CURSOR COLOR (C128/40COL)
PSC828=PCC428+1 ;SCREEN COLOR (C128/80COL)
PCC828=PSC828+1 ;CURSOR COLOR (C128/80COL)
PDN064=PCC828+1 ;DRIVE NUMBER (C64)
PDN128=PDN064+1 ;DRIVE NUMBER (C128)
PLU064=PDN128+1 ;LOGICAL UNIT (C64)
PLU128=PLU064+1 ;LOGICAL UNIT (C128)
PUS064=PLU128+1 ;USER NUMBER (C64)
PUS128=PUS064+1 ;USER NUMBER (C128)
PBF064=PUS128+1 ;BEEP FLAG (C64)
PBF128=PBF064+1 ;BEEP FLAG (C128)
PPD064=PBF128+1 ;PRINTER DEVICE # (C64)
PPD128=PPD064+1 ;PRINTER DEVICE # (C128)
PPS064=PPD128+1 ;PRINTER SECONDARY ADDRESS (C64)
PPS128=PPS064+1 ;PRINTER SECONDARY ADDRESS (C128)
PAL064=PPS128+1 ;AUTO SERIAL LOAD FLAG (C64)
PAL128=PAL064+1 ;AUTO SERIAL LOAD FLAG (C128)
PSF064=PAL128+1 ;PATTERN MATCH SCRATCH FLAG (C64)
PSF128=PSF064+1 ;PATTERN MATCH SCRATCH FLAG (C128)
PCPUMD=PSF128+1 ;CPU MODE
PSP064=PCPUMD+1 ;CPU SPEED (C64)
PSP128=PSP064+1 ;CPU SPEED (C128)



PKPDEN=PSP128+1 ;KEYPAD ENABLE FLAG (C64 ONLY)
PIR064=PKPDEN+1 ;IRQ TRAP (C64)
PIR128=PIR064+1 ;IRQ TRAP (C128)
PNM064=PIR128+1 ;NMI TRAP (C64)
PNM128=PNM064+1 ;NMI TRAP (C128)
PLR064=PNM128+1 ;PORT LOCK RETRY COUNT (C64)
PLR128=PLR064+1 ;PORT LOCK RETRY COUNT (C128)
PCPMLU=PLR128+1 ;THIS PORT'S CP/M LOGICAL UNIT
;
;
;
;
;THE FOLLOWING ARE SYSTEM INDEX BLOCK EQUATES:
;
INDNAM=$00 ;FILE'S NAME (16 CHARACTERS)
INDNBL=$10 ;NUMBER OF BLOCKS IN FILE (INCL. HEADER) HI,LO ORDER
INDBPR=$12 ;NUMBER OF BYTES PER RECORD  HI,LO ORDER
INDRIN=$14 ;NUMBER OF RECORDS IN FILE HI,LO ORDER
INDTYP=$16 ;FILE'S TYPE CODE
INDLOD=$17 ;FILE'S LOAD ADDRESS HI,LO ORDER
INDUSR=$19 ;FILE'S USER/LU INDICATOR
INDFLG=$1A ;FILE'S STATUS FLAGS (SUCH AS: CHANGED SINCE LAST BACKUP BIT)
INDRS1=$1B ;RESERVED FOR FUTURE USE
ENTCNT=$1C ;NUMBER OF ACTIVE ENTRIES IN THIS INDEX BLOCK (1ST SLOT ONLY)
INDACF=$1D ;ACTIVE SLOT INDICATOR FOR THIS INDEX ENTRY (***)
INDHBA=$1E ;FILE'S HEADER BLOCK ADDRESS  HI,LO ORDER
;
;
; *** 0=CURRENTLY ACTIVE ENTRY,$FF=NEVER USED ENTRY,$80=DELETED FILE
;
;
;
;
;
```

```
;THE FOLLOWING ARE THE FILE PARAMETER TABLE (FPT) EQUATES.  EACH TABLE
;CONSISTS OF 32 DECIMAL BYTES.
;
;
LOGFLN=$00 ;LOGICAL FILE NUMBER(1 BYTE - $FF=UNUSED FPT)
FILOAD=$01 ;FILE'S LOAD ADDRESS(2 BYTES - HI,LO ORDER)
RESRVD=$03 ;BYTES 3,4 AND 5 ARE RESERVED FOR INTERNAL USE
USERLU=$06 ;FILE'S USER/LU INDICATOR
FILHDR=$07 ;FILE'S HEADER BLOCK ADDRESS(2 BYTES - HI,LO ORDER)
CRBLOK=$09 ;CURRENT BLOCK DISPLACEMENT FROM FILE'S HDR. OR FROM BLK LIST
ABSBLK=$0B ;ABSOLUTE BLK. ADR. OF CURRENT BLOCK (2 BYTES)
CURBYT=$0D ;CURRENT BYTE OF FILE(3 BYTES I.E. NEXT TO BE ACCESSED)
NUMBLK=$10 ;NUMBER OF BLOCKS IN FILE(INCL. HDR.(2 BYTES - HI,LO ORDER)
NUMREC=$12 ;NUMBER OF RECORDS IN FILE(2 BYTES - HI,LO ORDER)
NUMBYT=$14 ;NUMBER OF BYTES IN FILE(EXCL. HDR. 3 BYTES - HI,MI,LO)
NUMBPR=$17 ;NUMBER OF BYTES PER RECORD(2 BYTES - HI,LO ORDER)
TYPFIL=$19 ;TYPE OF FILE(1 BYTE - SEE FILE TYPE EQUATES ABOVE)
FILSTS=$1A ;FILE'S STATUS(1 BYTE - HI ORDER BIT OF # OF BYTES IN
;LAST BLOCK IS THE LEAST SIG. BIT)
NUMBLB=$1B ;NUMBER OF BYTES IN LAST BLOCK(1 BYTE - LO ORDER PART)
EOFFLG=$1C ;END OF FILE FLAG
IDXOFS=$1D ;OFFSET TO INDEX BLOCK CONTAINING FILE'S ENTRY
LODFLG=$1E ;USED AS FLAG WHEN PASSING A FILES LOAD ADDRESS
SECADR=$1F ;FILE'S SECONDARY ADDRESS
;
.END
```

@@:13:7.2 Update Docs is a seq. file

Lt. Kernal DOS 7.2
11/26/90

PLEASE READ THIS ENTIRE FILE before you use this SYSGEN diskette. Quite a number of changes have been made, and some REQUIRE that you be aware of them.

This file will display properly in either the 40 or 80 column modes, and will print correctly using TYPE, or the "RUN THIS README" program on the disk.

This is the Lt. Kernal DOS you've all been waiting for. Thank you for your patience. We're sure you'll think the wait was worth while.

This new DOS incorporates a number of enhancements over DOS 7.1. This file supercedes and forms an addendum to the manual you received with your Lt. Kernal disk drive.

Some of the enhancements to DOS 7.1 were made available in the form of 'patch' files on the Xetec and Fiscal Information, Inc. BBS's. The DOS on this diskette contains all of those prior enhancements and several new features, in addition to a few minor 'bug' fixes.

This release fixes three bugs:

1) MERGE is no longer hard-coded to access device #8, but will default to the device number presently assigned to the Lt. Kernal.

2) The directory-generator for the '$' channel now properly lists the file names of files which are greater than 999 blocks in length. Previously, the file-size was listed properly, but the filename was left blank.

This bug only affected the directory listing obtained by reading the '$' channel, and not the listing obtained via the DIR command.

3) A half-stepping failure during error-recoveries in FASTCOPY has been fixed. It is now possible to recover from certain read errors on the floppy.

Many NEW features have been added to the Lt. Kernal. Those previously released as patch-files include:

****

Lt. Kernal Dos 7.1 patch 10/11/88

This fix enables CBM's new C-128 assembler package to run on the Lt. Kernal, and may allow other software to work which was previously not Lt. Kernal compatible.

****

CP/M patch 12/08/88

1) Implements the PROFILE function

2) Redefines the A and L drives so that the Lt. Kernal becomes A: and the first floppy becomes L: (the first VIRTUAL floppy is still E:)

These first two functions are options. If you wish to implement these two features under DOS 7.2, you must run the stand-alone program called "SWAP CPM UNITS". This program is not automatically copied to your Lt. Kernal during SYSGEN. You must manually copy it, along with several other programs we will discuss later.

3) Enables two undocumented features which have always been hidden in the Lt. Kernal CP/M system.

GO64 and GO128

Those two processors are hidden in the BUILDCPM command, and you can have them without difficulty. To get them, follow the procedure below.

2) Make sure you have CONFIGed a CP/M LU (already captured, or not).
3) Invoke BUILDCPM

   Just as 'T.COM' is contained in ram from $4000-$40FF, so are GO64.COM and GO128.COM held in other areas!

4) IF you haven't captured a CP/M image yet, SAVE the T.COM processor by the method detailed in the Lt. Kernal manual, but DON'T INVOKE IT!

5) Next: type SAVE<ret>
         type CCP<ret>
         type ↑C (control C)

   SAVE will ask the filename:
      type GO64.COM<ret>
   SAVE will ask the starting
     address (in HEX):
      type 4700<ret>
   SAVE will ask the ending
     address (in HEX):
      type 47ff<ret>

6) Next: type SAVE<ret> (again)
         type CCP<ret> (again)
         type ↑C (control C)

   SAVE will ask the filename:
      type GO128.COM<ret>
   SAVE will ask the starting
     address (in HEX):
      type 4800<ret>
   SAVE will ask the ending
     address (in HEX):
      type 48ff<ret>

7) NOW if you haven't captured CP/M yet, this is the time to invoke the 'T' processor, as outlined in the Lt. Kernal manual.

8 If you HAVE already captured CP/M, DON'T invoke 'T' now. Instead, re-boot to CP/M.

9) Last: PIP those two files to the Lt. Kernal drive (L or A depending upon whether or not you have run PATCH yet).

That's all. Now you have a DIRECT way to go from CP/M to the other two modes of your computer WITHOUT resetting. If the computer does hang up (it happens sometimes), just press the reset button and the process will continue as if it had gone correctly the first time.

****

Lt. Kernal Dos 7.1 patch 01/17/89

This patch updates Lt. Kernal Dos 7.1 with a some improvements to:

AUTOCOPY
AUTODEL
AUTOMOVE

The changes enable a new file type (TYPE 5) to be managed. Type 5 files are used ONLY by custom user-written Lt. Kernal software.

****

Lt. Kernal Dos 7.1 patch 02/14/89

This patch enhances the following modules:

EXEC
LKREV
Direct Mode Command Processor

1.) EXEC has been TOTALLY rewritten making it safe to run in harmony with other LK DOS commands; this includes VALIDATE. Other enhancements to EXEC are:

(a) EXEC files are no longer limited to 4 blocks. They may now be as long as 127 blocks.

(b) A PI sign followed by any byte OTHER than a pi sign will cause up to a 4 second pause of the EXEC file playback.

This is useful when you're trying to follow what's happening on your screen. If you need to send a pi sign out as a single data byte, then simply put two pi signs back to back in the EXEC file.

The byte after the pi sign determines the number of jiffies to delay. In addition:

(c) A similar feature also exists using the English Pound sign. A pound sign followed by any byte OTHER than another Pound sign will cause a fixed delay to be issued BETWEEN every EXEC file data byte. The delay may be varied, or switched ON/OFF as often as desired within the EXEC file. This is useful when using an EXEC file to DEMO a piece of software where you'd like to emulate REAL typing.

As with the PI sign, simply put two back to back Pound signs in your EXEC file to output one. The byte after the Pound sign determines the number of jiffies to wait between data bytes.

3.) LKREV now displays your drive's serial number in addition to your DOS revision information.

4.) A direct mode "LAST COMMAND" playback feature has been added. Hitting the equal sign (=) and return will redisplay your LAST direct mode input.

This is useful if you're doing a lot of similar back to back direct mode commands.

****

Lt. Kernal Dos 7.1 patch 05/01/89

This patch updates Lt. Kernal Dos 7.1 to implement a couple of user friendly enhancements.

1.) A slight change has been made to the serial bus "AUTOLOAD" feature. Up till now, if you had "AUTOLOAD" enabled in CONFIG and a load request was passed ond to a floppy that was turned OFF, your program would get back a "DEVICE NOT PRESENT" error. This patch causes the "FILE NOT FOUND" error to be reported instead. This was done because there are many programs that do a TRIAL LOAD to check for the presence of a file, and do NOT expect to get a device error. Now, when you don't need your floppy, you can leave it turned OFF instead of having to repeatedly change your "AUTOLOAD" option in CONFIG.

2.) This next feature is one we have had many requests for over the years. For lack of a better description, it is a user definable ALTERNATE DIRECT MODE command processor. Example:

Let's say that instead of typing "FASTCOPY" you'd like to type "FC", or perhaps "U1" instead of "USER 1". This feature permits you to define your OWN commands, and the DOS actions they will represent.

The feature uses a KEY FILE for fast translation, and allows you to define up to 225 of your own custom DIRECT MODE commands and/or LK DOS command equivalents. The KEYS can be from 1 to 4 characters long, which may be linked to any corresponding ACTION string from 1 to 26 characters long.

In addition, each ALTERNATE command carries with it an ACTION FLAG which controls whether you want the alternate command to be performed as soon as you hit RETURN or to have it DISPLAYED on the screen and to pause until you press RETURN again. This feature gives you a chance to see and modify the command before you actually execute it.

understand how the Lt. Kernal performs a DIRECT MODE search. When a string of characters (without leading spaces) is entered, the following ORDER of searches may take place:

a. First, a search is done on LU 10 USER 0. If found and it is an executable file then control is given to it.
b. If file is NOT found on LU 10 USER 0 AND your ALTERNATE commands are ENABLED, Then a search of your ALTERNATES is performed and executed if one exists.
c. Finally, if command still NOT found, then your current LU and USER is searched.

This concept is important to keep in mind so that you DON'T enter an ALTERNATE KEY with the SAME name as a file or DOS command that ALREADY exists on LU 10 USER 0. If you do duplicate an LU 10 command, the ALTERNATE will NOT be acted upon.

Having understood all that (we hope), let's talk about how to set up and use this feature.

a. COPY: You must copy the file called "ALTED" to your hard drive (use FASTCOPY or COPY-ALL). Do this while in C64 mode to insure its file type staying as TYPE 11.

This is your ALTERNATES editor which allows you to define and maintain your ALTERNATE commands. You may put this editor on ANY LU/USER. However, it will be easier to use if it's on LU 10, and USER 0.

Note: This is a BASIC program with a MACHINE LANGUAGE tail. DO NOT modify or resave it or you may render it useless. ALSO, the editor ONLY runs in the C64 mode. Your ALTERNATES WILL be usable in BOTH the C64 & C128 modes.

d. NOW, invoke the editor by typing "ALTED" and return. The editor is a menu-driven self explainitory program. It gives you the ability to ADD, MODIFY, DELETE, or PRINT your ALTERNATES file.

When you enter the "ADD" option, the editor checks for the existance of your KEY file. If it does NOT exist, it is automatically built for you. As we mentioned, this is a KEY FILE and it will be built on LU 10 USER 0 under the name "USER.ALTERNATES". This file can be copied and/or moved to other LUs for back-up purposes. However, it MUST reside on LU 10 USER 0 to use the ALTERNATE commands defined in it.

IMPORTANT - following your FIRST TIME use of the ADD option which as stated initially CREATES your KEYFILE, you MUST re-boot your system to make the ALTERNATES feature active. AFTER that, it is NOT necessary to reboot when additions or modifies are made in your KEYFILE.

There is a NEW indicator in the READY STATUS line that tells you if your ALTERNATES are ACTIVE. The space immediately following the "C64/C128" indicator will have an asterisk (*) in it IF your ALTERNATES ARE ACTIVE.

Also, you may turn on and turn off your ALTERNATES feature with the following DIRECT MODE commands:

"ALTD" and return to DISABLE
or
"ALTE" and return to ENABLE

Your CURRENT ON/OFF status will be reflected by the asterisk indicator at all times. Also, this setting will be remembered by the DOS even if you reboot your system.

3.) There has been a slight change made to the "LAST COMMAND PLAYBACK" feature (=) that was implemented in the 02/14/89 patch. The cursor is now left left positioned at the END of your playback text instead of the beginning. This makes it easier to quickly modify command tails.

****

That summarizes all the prior 'patch' changes:

Now for THIS NEW DOS's features.

In this DOS, the DIR processor has been enhanced so that the listing produced in the C-128 80-column mode occurs much faster. All the patches released earlier have been incorporated into 'linear code' in this DOS.

The most noticable improvement is in FASTCOPY. In fact, it's been entirely rewritten. There are sufficient new features in FASTCOPY, alone to make up an entire DOS update.

Immediately upon invoking FASTCOPY, you will notice a difference: You may now SPECIFY THE DEVICE NUMBER of the floppy drive on which FASTCOPY will run. If a diskette drive is present at the device address you specify, it will be identified by type. That brings us to the next, very dramatic improvement to FASTCOPY; drives supported.

The FASTCOPY now supports:

1). 1541's.

2). 1571's in the 1541 mode.

3). 1571's in the native, double-sided mode.

4). 1581's.

That's right: FASTCOPY now supports ALL THREE standard Commodore serial drives. Using a 1581 on a C-128, it's incredibly fast. In fact, FASTCOPY can fill an already-formatted 1581 diskette (3160 blocks PLUS 40 blocks of BAM and index) in under three minutes, twenty-five seconds.

FASTCOPY has been heavily tested on the following drives/configurations:

1) NTSC and PAL C-64's

2) C-64's, flat C-128's and C-128 'D' models.

3) VIC 1541, 1541, 1541-C

4) 1571 and 1571 'D'

5) 1581 with the newer WD 1772 disk controller.

Once you've specified a drive, and the floppy-dos has been downloaded, you are presented with a familiar FASTCOPY menu, with a couple of visible changes.

1) The selected drive is identified.

2) TWO NEW MENU ITEMS appear.

The new menu items are:

A) F3 - DISKETTE DUPLICATOR

B) F4 - RESELECT SERIAL DRIVE

The F4 option allows you to specify a new floppy drive without leaving the FASTCOPY menu.

The F3 option is a powerful tool for software distributers and for diskette maintenance. It allows you to create a copy of a single diskette surface in ONE PASS, using a work-file on the Lt. Kernal as a holding buffer.

You are allowed to verify or not verify your copies. You are allowed to specify the file name of the holding file. If the file does not exist, FASTCOPY will create it, and proceed to copy the source diskette into the file.

After the source diskette is copied, you are instructed to insert a destination disk into the floppy drive, and the entire diskette surface is created in a single pass. After the destination copy has finished, you're asked if you'd like to make ANOTHER COPY of the same diskette.

This is a perfect tool for software distributers who have many identical copies to produce. But there is a more powerful feature here waiting to be exploited.

After a diskette has been copied with the DUPLICATE function of FASTCOPY, the holding file is RETAINED. Later, when you specify a holding-file name for a DUPLICATE operation, you may RE-USE the same file. If you do, you have the choices of reading a NEW diskette into the file, or GOING DIRECTLY TO THE DESTINATION PHASE of the copy. This allows you to create identical copies of a diskette again and again, and only requires you to read the original disk ONCE.

The duplicator mode was intended to function properly only on unprotected diskettes; that is, diskettes with no intentional errors. FASTCOPY can proceed past some errors, but they will NOT be duplicated on the destination diskette. The duplicator mode was intended to be a FAST, EASY mode for mass duplication of diskettes.

The remaining functions of FASTCOPY will be familiar, but they have been significantly 'cleaned up', and have a number of changes in their flow. You need to be aware of the changes, since they affect your perception of how the copying proceeds, and REQUIRE CHANGES to any EXEC files used to drive the FASTCOPY process. To that end, we've tested to ensure that FASTCOPY is EXEC-able on both the C-128 and C-64. One important feature has been added to the RESTORE mode to better support EXEC driven copies.

Here are the additions to the FASTCOPY BACKUP and RESTORE modes:

1) In both BACKUP and RESTORE, the LU and USER responses will default to the CURRENT LU and USER. If you change them, the changes will be retained, so that you do not need to re-type them the next time that prompt is encountered.

2) In the RESTORE mode, you are permitted to specify O(riginal) or D(ifferent) destination LU/USER than that from which the diskette was created. The Original entry automatically routes files to the same LU and USER from which they came.

3) The ALL USERS option is now permissable during BACKUP. This feature allows you to back up an entire LU in one copy session, without having to manually specify each USER partiion. The files are automatically tagged on the floppy with their source LU and USER so that they may be RESTOREd to the same place later. IDENTICAL FILE-NAMES from multiple USERs ARE ALLOWED on the same destination diskette.

4) More file-type selections are available at the type-match option. You may now specify the file-types by NUMBER or LETTER type.

   The types supported now include BASIC (B) or M/L (M) types, and EITHER (P) in the RESTORE mode. Additionally, FASTCOPY now allows the BACKUP and RESTOREing of KEY

type 5 files).

KEY and CONTIGUOUS types are backed up as USR files on the floppy drive to retain Commodore compatibility. They are tagged with their characteristics so they may be properly RESTORED.

Immediately upon entering the RESTORE option, you are greeted with a new prompt: The FILE-EXISTS INTERVENTION option. It's not as complex as it may sound.

This feature was added to enhance the ability to drive FASTCOPY from EXEC files. This item allows you to specify what action to take automatically if FASTCOPY detects that the file being RESTOREd already exists on the destination LU/USER. You may specify that FASTCOPY S(kip) or R(eplace) the files on the destination. No prompts will be issued for those choices during the copy, so that EXEC may proceed even when unanticipated files are found during RESTOREs.

You may select the MANUAL mode of intervention instead of the automatic mode. Then FASTCOPY will prompt you for a choice of actions when pre-existing files are encountered.

You are given a much richer array of opportunities to alter the flow of FASTCOPY during both BACKUP and RESTORE operations. Pay close attention to all the new prompts, and the places where you now have the chance to back up a step, or quit entirely. The possible combinations are much greater in number and much friendlier to the user.

Please take note of the following:

During DOUBLE SIDED 1571 and 1581 format passes, the format appears much slower than before. There's a LOT MORE SPACE to format! The 1581 has over 4 and one-half TIMES the capacity of the 1541. Of course, the 1571 has twice the capacity of a 1541, since it uses the same diskettes, but in a double-sided mode of operation.

During 40-column copies with the 1581 drive, the familiar flicker-line has been replaced with a flashing screen. This was done in the interest of speed, and is not a bug. Displaying the line just took too much copy time away from the VERY FAST 1581.

The 1581 is inherently much faster than either the 1571 or 1541. Since it is faster, it is able to fully exploit the 2 MHz mode of the C-128.

In that interest, the 1581 ALWAYS uses the two-megahertz mode of a C-128, even in the 40-column, C-64 mode. This is totally transparent to the user, but will explain why c-64 mode FASTCOPYs on a 1581 go faster on a C-128 than on a C-64.

Now some goodies we've included with this SYSGEN diskette, and then the instructions on how to customize this disk to work with your system.

Included on the diskette are files which will NOT be automatically copied to your hard-disk during the SYSGEN. Among them is the ALTED editor program, discussed earlier.

We've included a 1581 diagnostic pack graciously provided by Fred Bowen of Commodore Business Machines. Just to reiterate: We have ONLY tested the new FASTCOPY 1581 routines on 1581's which have the new WD1772 chips installed. We recommend you have your 1581 updated if you do not already have this chip. The diagnostic will tell you which chip is presently installed.

We also have provided a powerful new tool to you 'serious' Lt. Kernal users:

LKMON

LKMON is a complete Lt. Kernal debugging monitor. It provides disk-editing of Lt. Kernal host-adaptor virtual file editing, memory editing, 'SHADOW RAM', and your first 'official' tool for working DIRECTLY with the Lt. Kernal's internal routine.

LKMON is essentially self-documenting with a large internal help menu. One VERY critical point needs mention here.

Any time you are editing a file with LKMON, ANY CHANGES you make in the

to save your changes.

This is a great plus, but can be very dangerous, if you aren't careful.

NEVER EDIT A LT. KERNAL DOS

component unless you FULLY understand how to use LKMON, and unless you are COMPLETELY BACKED UP, and have a KNOWN WORKING SYSGEN diskette handy.

Be especially careful, also, of any work you do with LKMON within a system file like SYSTEMINDEX or DISCBITMAP. Any unintentional changes there may cause files to be lost.

With the power comes responsibility. Use LKMON carefully, and it will serve you well.

******* HELPFUL HINT **********

Direct block-reads and block-writes with LKMON are done RELATIVE to the beginning of any Lt. Kernal NATIVE LU. You may wish to explore regions of your hard-disk which do not fall within the bounds of a 'native' LU, like, for instance, a CP/M LU. In that case, you need only compute the ABSOLUTE block where the region lies (relative to the beginning of the drive), and use LU 10 as the reference LU. As you remember, LU 10 begins on the very first block of the hard-drive. LKMON will allow you to access blocks BEYOND THE END of the LU used as a starting reference.

Last, we come to HOW TO USE this disk.

In order to customize this diskette to work with your Lt. Kernal, you MUST have a working DOS 7.1 diskette which EXACTLY MATCHES the characteristics of the system you are now running.

This needs emphasis. Many users have upgraded their own Lt. Kernal systems with larger or faster drives than those which were installed when they first bought the systems. The 7.1 SYSGEN you use for a reference disk during this customization MUST be one which EXACTLY reflects your CURRENT system.

Additionally, if you have not done it already, VERIFY that the old 7.1 SYSGEN actually works, before you begin to customize this new diskette. Nothing could be more frustrating than to think that your new 7.2 SYSGEN is bad, when the fault is actually with the OLD one.

To customize your 7.2 sysgen diskette, FIRST make an exact, block-for-block copy of BOTH SIDES of the original disk we sent you. Use a program like the DISECTOR 3-Minute copier, or CLONE MACHINE. We've had bad luck with the FAST HACK'EM package, at least when copying SYSGENs. Put the original away in a safe place, and ONLY work with the copy.

Next, load the program 'CUSTOMIZE' from the back side of the new disk, RUN it, and follow the instructions.

After you've received the notice that the customization is complete, you may proceed to use the SYSGEN diskette as you would normally.

warning!!

Remember to RECORD your LU parameters before you begin the SYSGEN. Use the CONFIG processor to display and print them. Also remember to AUTOCOPY any utility programs and alternates files you may have on LU 10 to another LU. SYSGEN completely clears and re-creates LU 10.

This new DOS has been a wonderful opportunity to better serve you, our customers. You've made the Lt. Kernal a great success. This is our way of saying THANK YOU.