

**SUPER UTILITY PLUS 3.0  
TECHNICAL MANUAL**

by

**Kim Watt**

**DOS NOTES by Pete Carr**

(Place these pages in your Super Utility Plus  
3.0 Manual binder.)

Published by Breeze/QSD Inc., 11500 Stemmons  
Expressway, Suite 125, Dallas TX 75229  
(214) 484-9428



**SUPER UTILITY PLUS 3.0  
TECHNICAL MANUAL**

by

**Kim Watt**

**DOS NOTES by Pete Carr**

**Published by Breeze/QSD, Inc., 11500 Stemmons  
Expressway, Suite 125, Dallas TX 75229  
(214) 484-9428**

**Copyright (c) 1983 by Breeze/QSD, Inc.**

No part of this publication may be reproduced, either manually or automatically, by any means including the use of electronic, electromagnetic, xerographic or optical information storage media without the express written consent of Breeze/QSD Inc.

**First Printing, April 1983**

## PREFACE

The contents of this manual have been revised to reflect the changes in the new Super Utility Plus Version 3.0. Several new routines have been added, and others have been recoded for greater efficiency or to eliminate problems. This manual should therefore be used with SU+ 3.0 only.

This manual will attempt to provide the assembly language programmer with information needed to interface programs with Super-Utility Plus. This is not a tutorial on machine language programming, and a basic understanding of the TRS80 microcomputer and assembly language programming is assumed. Super-Utility Plus has the capability to access most devices currently available on the Model I and III. With this manual, routines not available in SU+ may be implemented by the user. This manual contains examples in the back of routines that may be added to the SU+ program.

This manual does not contain source code for SU+, nor are specific details revealed about the intimate workings of the program. What will be covered in this manual is an explanation of the system labels and entry points available in SU+. When the label points to an entry point of a complex function of SU+, only the function of the routine is described. Thus, the entire sequence for "Display Disk Sectors" is not given, but just that it is the entry point for that complex routine. All subroutines throughout the program are defined and documented however. When one subroutine calls another, that link will usually be shown.

Throughout this manual, the following notations will apply:

First, the name of the label is shown, preceded by an @ symbol. The corresponding addresses for these labels may be found in the symbol table at the back of the manual. Note that there are separate symbol tables for Model I and Model III.

Then a brief description of the function of the routine is given.

Then, if applicable, the Entry and Exit conditions are given. If this is an entry point to a complex routine, only the routine itself is mentioned. Only CPU registers that are involved in the routine itself are mentioned. Both the entry and exit conditions are listed. If a register(s) is NOT mentioned in a routine, it is implied that it is NOT USED, or that it is NOT CHANGED. Most of the routines will preserve as many registers as possible so that parameters contained elsewhere are not disturbed.

If a register or label is shown surrounded by parentheses, then THE CONTENTS of that register/label is implied. Thus, (HL) means the contents of the address pointed to by HL. If HL=4000H, and the byte at 4000H=0AH, then (HL)=0AH.

If the symbol => appears after a register pair or label, then the registers/label POINTS TO an address in memory. Thus if HL=>@BUFFER, then HL points to a buffer to be used.

If the symbol = appears after a register or label, then that register/label EQUALS the parameter. Thus, if A=drive, then the Accumulator must contain the drive number to be used.

A sample printout of the addresses of the labels in the current version of SU+ (3.0) are given on the following page. If you have an older version, or none at all, please read ahead to 'order / upgrade information'.

**SU+ TECH MANUAL - SOME DOS NOTES**

by Pete Carr

Back in the old pre-Newdos, pre-double density, pre-Model III days there was no TRS-80 DOS compatibility problem. At that time Model I TRSDOS was the only TRS-80 DOS. All the DOS's that have followed made a point to follow TRSDOS's lead as far as diskette structure, directory organization, for the sake of compatibility. So when the first version of Super Utility was released it was easy for it to be compatible with ALL the DOS's because ALL DOS's were compatible!

But those were easier, less complicated times. With the advent of the TRS-80 Model III and various Double Density boards for the Model I, there can hardly be any so called TRSDOS-COMPATIBLE, standard format; because TRSDOS itself is changing so much. Nowadays when someone says they are TRSDOS compatible you could ask, "Which TRSDOS are you talking about?" Most of the new DOS's work in a very similar fashion to the first Model I TRSDOS. But, because Tandy has been changing TRSDOS so much lately there has really been no reason, nor would it make any sense, to attempt to play follow the leader anymore. So this has actually freed other DOS authors to pursue different roads that allow them to implement more powerful features. This is good and bad. It is good, because it allows DOS authors to apply a very creative approach to writing a DOS. They

don't have to worry about their product being EXACTLY compatible with TRSDOS anymore thus can write without being constricted by that thought in mind. The result of this is that we are able to have very powerful systems on a TRS-80 with features like double density that once were thought only feasible on a much bigger computer. Kind of a freeing of the programmers from forced conformity, if you will. Of course the bad part is the lack of compatibility between the DOS's. Thus it seems to be getting harder to write a program that will run on all DOS's.

With this reality in mind the new Super Utility Plus has made it easy to work on any of the major DOS's, and the disk formatted by them, by its CONFIG feature. I will try to give an overview of the popular DOS's. There is no reason to go into minute details that you can read for yourself in the DOS's manual. I will not cover the basic features of the DOS's, but only some of the physical ways in which they differ. Some DOS's are not being supported on the market anymore or are so obscure that I won't cover much if anything about those that fit this category. We'll start with a quick overview of the Floppy Disk Controllers used in the TRS-80.

## **1. FLOPPY DISK CONTROLLERS (FDC)**

There are mainly three different FDC's, that for different purposes, are being used in the TRS-80 Model I and III at this time. An FDC is actually a ROM (read only memory) program that can be accessed by a programmer to perform certain functions pertaining to the floppy disk. These functions include writing "Data Address Marks", Head Stepping and various other needed functions

of controlling your floppy disk. It depends on which machine (I or III) and if you have one of the available Double Density boards installed (Model I) as to which ones apply to you.

### Model I.

The stock Model I comes with a 1771 FDC. This controller can read/write up to four different "Data Address Marks".

These can be grouped as:

1. Standard
2. Read Protected
3. Deleted Data
4. User Defined

### Double Density Model I

When you install one of the available Double Density boards you then have two FDC's in your Model I. Along with the 1771 FDC you now have a 1791 FDC which gives the Model I the capability of reading/writing all the DATA ADDRESS MARKS along with reading/writing double density.

### Model III

The Model III comes with a 1793 FDC. This controller can read/write double and single density. It recognizes "Read Protected" and "Standard Data" BOTH as "Standard Data". Likewise, it also recognizes "Deleted Data" and "User Defined" as "Read Protected". Furthermore, it is only capable of writing the "Standard Data" and "Read Protect" marks which would be recognized in a Mod I as "Standard" and "Deleted Data". Due to the differences in data address mark recognition, some authors of the current DOS's are using the "User

Defined" address mark on the directory track on Mod I single density diskettes so that it may be detected directly on the Mod III with no conversion.

There are a couple of other FDC's sometimes used with the TRS-80 that are a little different (LX80 uses a Fuji) but the affects they have on compatibility for the most part are very small.

## 2. THE DISK OPERATING SYSTEMS.

### Super Utility+ DOS Config parameters

1. TS : Single Density - Non-Ldos
2. TD : Model III TRSDOS
3. LS : Model I or III Single Density LDOS
4. L1D : Model I Double Density LDOS with SOLE track
5. LD : Model I or III Double Density LDOS
6. DS : Model I or III Single Density DOSPLUS
7. D1D : Model I Double Density DOSPLUS System disks
8. DD : Model I/III Double Density DOS-PLUS
9. MS : Model I/III Single Density MUL-TIDOS
10. M1D : Model I Double Density MULTI-DOS System disks
11. MD : Model I/III Double Density MUL-TIDOS
12. BD : Model I Double Density DOUBLE-DOS
13. NS : Model I/III Single Density NEW-DOS/80 V2

14. N1D : Model I Double Density NEW-DOS/80 V2 with Track 0 reversed density
15. ND : Model I/III Double Density NEW-DOS/80 V2

\*NOTE: We will use TRSDOS 2.3 for the Model I as our model DOS. A difference is that most the other DOS's allow more than 35 tracks which could put the directory track in a different location than 17 (11H). They also give you the option of Double Density, with the appropriate hardware, which has 18 sectors per track instead of 10. NEWDOS80 V2 and Percom's DOBLEEDOS do operate quite differently concerning the track sector format which will be discussed in the NEWDOS80 section. Model III TRSDOS 1.3 also operates differently which will be discussed in its section.

### TRSDOS 2.3

Model I TRSDOS uses a single density, 35 track, 10 sectors to the track, 256 bytes per sector, configuration. This DOS was written to be used with the FDC 1771.

#### PHYSICAL STORAGE INFORMATION:

THE DIRECTORY: Location - Trk 17 (11H)

Sector 0 - Granule Allocation Table - GAT

The first sector contains the Granule Allocation Table (GAT) information. This tells the operating system which granules are allocated to files or locked out and which ones are free to use. (One GRANULE = 5 sectors. 2 granules = 1 track). This

sector also contains the disk master password, disk name, date and the AUTO command. The first 35 bytes contain information for each of the 35 tracks. In each of these 35 bytes is a HEX number representing allocation info about that track.

Their meaning follows:

FC = track empty

FF = track full.

FE = first 5 sectors available

FD = last 5 sectors available

The track lockout bytes start at byte 60H. If a track is locked out for some reason (unformatable), let's say track 8, then byte 68H will contain an FFH. If the track is available for system use it will contain an FCH. Depending on which DOS and TRS-80 Model you have, you could notice that its GAT uses a set of different HEX bytes than the ones mentioned above, but the idea will still carry through.

#### Sector 1 - Hash Index Table - HIT

The Hash Index Table contains a one byte code (hash code) for each file stored in the directory. The location of the hash code points to where a file is located in the directory sectors. They are grouped into eight sections, one for every file storage sector in the directory. Since the first two sectors of the directory contain the GAT and HIT tables, that leaves only eight sectors left (single density) for the actual file information.

**Sectors 2 to 9** File Primary Directory Entries  
**(FPDE)**  
File Extension Directory Entries  
**(FXDE)**

These sectors contain the actual directory information concerning the file names, attributes, passwords, size, end of file, etc. Each file (FPDE) is allowed 32 bytes for this information unless it needs more room for an extension to that file. An extension is used when a file can't be stored in a physically contiguous manner on the diskette. If a file needs more than 4 extensions to be written to the diskette a File Extension Directory Entry (FXDE) is created. TRSDOS 2.3 will allow as many extensions to a file as there is free disk space. Model III TRSDOS works somewhat differently concerning these file extensions which will be discussed later.

Most of the other DOS's allocate their directory information very similar to TRSDOS 2.3, except for the obvious track and density differences.

DOSPLUS and LDOS single density conform to the first TRSDOS format, but of course, its double density and Model III versions do differ where applicable. They use the term cylinder instead of track because of their doublesided and hard disk capabilities. Instead of 10 sectors per track the Model III and double density versions of DOSPLUS and LDOS contain 18 sectors per cylinder (track), 6 sectors per granule, with 3 granules per cylinder (track).

NEWDOS80 V2

Apparat's latest DOS has defined a new way of handling disk allocation. It uses a "disk relative sector" technique instead of using the real physical track sectors. This is the reason for the new term LUMP. The bytes in the GAT sector from 00 to BFH correspond to a LUMP instead of a track, so granules per lump is used instead of granules per track. NEWDOS80 V2 single density, uses 5 sectors per Granule, (more with double density) BUT can have 2 to 8 granules per LUMP. This allows the GRANS to span disk tracks, starting on one track and ending on another. According to Apparat, this maximizes the number of sectors per track while keeping a normal directory track format. NEWDOS80 V2 also allows a Single or Double density Boot track which is the reason for SU+ having the N1D and ND Config params.

DOUBLEDOS from Percom also uses this "Disk relative sector" technique like NEWDOS80. It does not have the capabilities of NEWDOS80 in defining how many granules to use per LUMP or defining which density the BOOT shall be; but its physical way of handling the disk is similar. Thus, it needs only one SU+ Config param which is BD.

TRSDOS 1.3 - Model III

This DOS uses a different way of defining its directory and physical sector format. It contains 18 sectors per track starting at sector 1 instead of the usual starting point of 0. Its Granules are 3 sectors long for a total of 6 Granules per track. Its directory uses a 48 byte FPDE instead of the usual 32 bytes. These extra bytes allow it to have

more extensions without creating an FXDE (file extended directory entry). Matter of fact Model III TRSDOS 1.3 does not use the FXDE procedure for allocating file extensions at all! As stated before all other DOS's will continue to create FXDE's until your disk is full. TRSDOS 1.3 will not do this, BUT you are allowed up to 13 extensions which should take care of all but the most unusual cases. A pretty good tradeoff for a cleaner and hopefully easier to manipulate directory, giving the DOS less chance for error! The other major difference is as mentioned above TRSDOS 1.3 uses a sector offset of 1. This means that each track's sector starts at 1 instead of 0 like the other DOS's. Use the Super Utility + config param (TD) for TRSDOS 1.3.

**@MODE**

Current modify mode number base: (1 byte)

- 0 = HEX
- 1 = Decimal
- 2 = Binary
- 3 = Octal
- 4 = Ascii

**@SECTOR**

Current setting of LAST for sector number (1 byte).

**@TRAK**

Current setting of LAST for track number (1 byte).

**@CURSOR**

Current video cursor address (2 bytes).

**@EOFB**

End of file byte for current file (1 byte).

**@EOFS**

End of file sector for current file (2 bytes).

**@EOAS**

End of allocation sector for current file (2 bytes).

**@FREEG**

Free granules on current disk (2 bytes).

**@FREEF**

Free files on current disk (1 byte).

**@CGRANS**

Holds number of grans to copy in copy files (2 bytes).

**@ADDRESS**

Current address to be displayed (2 bytes). Used by @SHOW.

**@DEFADDR**

Address to be used in jump to memory (2 bytes). Defaults to @MENU.

**@FMTBUFF**

Address used by @BUILD when formatting a track.

**@MIDMEM**

Address of the middle of the buffer area. Used by exchange disk sectors.

**@DIRSCNT**

Holds the sector count of the last directory read into memory.

**@DIRPAGE**

Holds the position in the directory of the current portion being displayed by kill/restore files. Only 8 sectors of data can be displayed at a time, and the byte here indicates the current 8 sector location.

**@TRUE**

The REAL track as it is read from the disk. Used by display disk sectors.

**@TYPE**

Indicates where the displayed data came from.

1 = from display disk sectors

2 = from memory

3 = from display file sectors

**@TOPMEM**

Holds the address of the current top of memory  
+1. Normally contains 0000H.

**@NUMTYPE**

Holds the base of the last string where  
@VALUE was extracted.

**@RESULT**

Holds the non-masked result of the last disk  
I/O.

**@TEMPO**

Temporary storage area (2 bytes).

**@TEMP1**

Temporary storage area (2 bytes).

**@TEMP2**

Temporary storage area (2 bytes).

**@TEMP3**

Temporary storage area (2 bytes).

**@TEMP4**

Temporary storage area (2 bytes).

**@TEMP5**

Temporary storage area (2 bytes).

**@TEMP6**

Temporary storage area (2 bytes).

**@TEMP7**

Temporary storage area (2 bytes).

**@TEMP8**

Temporary storage area (2 bytes).

**@NMIVECT**

Non-maskable return address for disk I/O. Used by Mod III version only, unused in the Mod I.  
Located at fixed address 4049H.

**@RETNMI**

RETN instruction, normally, @NMIVECTR points here if there is no current disk I/O in progress.  
Not executed on the Mod I.

**@FLAGA**

System parameters flag.

Bit 7 = set = DUAL currently ACTIVE  
Bit 6 = set = Highspeed clock is ON  
Bit 5 = set = Dual Flag ON  
Bit 4 = set = Task spooler de-activated  
Bit 3 = set = Extended ID marks ON  
            Bit 2 = set = Replace string in string search  
Bit 1 = set = Alive OFF  
            Bit 0 = set = Keyboard case reversal ON

**@FLAGB**

System parameter flag for printer (1 byte).

Bit 7 = set = Graphics ENABLED  
Bit 6 = set = Lower Case ENABLED  
Bit 5 = set = MX80 graphics adjust ENABLED  
Bit 4 = set = NO double density available  
Bit 3 = set = Linefeeds ENABLED  
Bit 2 = set = Radio Shack Doubler  
Bit 1 = unused  
Bit 0 = set = Trace ON

**@SDRIVE**

Binary drive number of current SOURCE drive.

**@DDRIVE**

Binary drive number of current DESTINATION drive.

**@KEYBRD**

Mask area used by keyboard driver (7 bytes).

**@MFLAG**

Flag indicating if DISK MOUNT prompts are to be issued.

**@DISP1**

Current DECRYPT mode (1 byte). Will contain one of the following symbols:

+ = addition  
- = subtraction  
 $\Lambda$  = AND  
O = OR  
X = XOR  
S = SHIFT  
R = ROTATE

**@DISP2**

Modifier byte of @DISP1 (1 byte).

If +, -,  $\Lambda$ , O, or X, then this byte contains the value to be added, subtracted, etc.

If S or R, then this byte contains L or R to indicate if shifts/rotates are Left or Right.

**@DISP3**

Modifier byte of @DISP2 (1 byte).

If @DISP1 contains S or R, then @DISP2 contains the direction, and this byte contains the number of Rotations/Shifts.

**@COUNT**

Temporary countdown storage (2 bytes).

**@INPUT**

I/O buffer and work space when reading ID marks (10 bytes).

**@STRING**

Input buffer for keyboard (70 bytes).

**@STACK**

Stack area (backward for 266 bytes).

**@FILEDCB**

Device control block for filenames (16 bytes).

**@PASSWRD**

Ascii password storage area (8 bytes).

**@DCT0 - @DCT7**

Drive Code Tables for 7 drives:

+00 = physical track count of disk  
+01 = relative track count of disk  
+02 = directory track  
+03 = current head location  
+04 = resident flags  
    7 = 1 = software NOT IN SYSTEM  
    6 = 1 = software WRITE PROTECTED  
    5 &4 = READ motor delay in 1/4 secs  
    3 &2 = WRITE motor delay in 1/4 secs  
    1&0 = disk drive step rate  
+05 = resident flags  
    7 = 1 = current operation is WRITE

6 = 1 = double step drive  
5 = 1 = active in multiple drive  
commands  
4 = 1 = auto disk detect next  
directory Access  
3 = 1 = attempt non-standard I/O  
2 = unused  
1 = 1 = read side 1 this access  
0 = 1 = double sided media  
available  
+06 = dos relative flags  
7 = 1 = double density track 0  
6 = 1 = double density disk  
5 = 1 = track 0 unavailable for  
file I/O  
4 = 1 = relative sectoring  
available  
3 = 1 = relative sectoring  
engaged  
2 = 1 = directory DAM's reversed  
1 = starting sector track 0  
0 = starting sector disk  
+07 = dos type (below)  
+08 = highest sector track 0  
+09 = highest sector disk  
+10 = sectors / granule  
+11 = granules / track  
+12 = default length of directory

**DOS types:**

UNKNOWN	00	U, U1, U3, US, UD, U1S, U1D, U3S, U3D
TRSDOS	01	T, T1, TS, T1S
	02	T1D
	03	T3, TD, T3D
LDOS	04	L, L1, LS, L1S, L3S
	05	L1D
	06	L3, LD, L3D

DOSPLUS	07	D, D1, DS, D1S, D3S
	08	D1D
	09	D3, DD, D3D
MULTIDOS	10	M, M1, MS, M1S, M3S
	11*	M1D
	12	M3, MD, M3D
NEWDOS	13	N, N1, NS, N1S, N3S
	14*	N1D
	15*	N3, ND, N3D
DOUBLE DOS	16*	B, B1, BD, B1D
EXTRA	17	X, X1, XS, X1S
	18	X1D
	19	X3S
	20	X3D

(\* indicates relative sectoring available)

#### @DCTTBL

Table of locations of each of the 8 DCT's

#### @DISPLAY (RST 08H)

Video display driver.

Entry: Data must immediately follow the CALL.

Data terminates with 00H.

Exit: Control passed to byte following terminator.

ALL registers AND Flags are preserved.

#### @CONTRL

Video display driver for control bytes (less than 20H).

Entry: A = byte to display

HL = cursor position

5 = clear screen and draw border

6 = clear screen blank

7 = clear lower portion  
of video only  
8 = backspace  
9 = center cursor on  
screen if left half, else  
linefeed  
10 = linefeed  
11 = upward linefeed  
13 = linefeed  
29 = move cursor to  
beginning of line  
30 = beginning of line  
AND erase line

Exit:            HL = cursor position

NOTE: (@CURSOR) is NOT updated.

HINT: It is best to display control codes via  
use of @DISPLAY. To clear the screen, for example:

```
RST 8      ;call @DISPLAY
DEFB 5      ;clear screen code
DEFB 0      ;terminator
```

This routine only requires 3 bytes, ALL  
registers are preserved, and (@CURSOR) IS  
updated.

#### @GETSTR (RST 10H)

Get a string of characters from keyboard.

Entry:        B = maximum length of input

Exit:        B = actual length of input

C = maximum length of input

HL => input string (=@STRING)

A = first character input

Z = NO characters input  
(represents status of B  
register)

**@KEY**

Scan keyboard.

Entry:        NONE

Exit:        A = input character

                Z = NO character input (A = 0)

NOTE: This call will perform the following functions:

Repeating keys

Upper/Lower case toggle with SHIFT 0

Screen printer with SHIFT CLEAR

There is NO keybounce delay with this call  
(very fast)

**@KIGO**

Scans keyboard for key, but keys WILL NOT repeat.

Entry:        NONE

Exit:        A = input key

BC, DE, HL are destroyed

NOTE: Calls should be preferably made to @KEY as all registers except A are preserved.

**@SCREENPRT**

Sends contents of video to the printer.

Entry:        NONE

Exit:        A = 0

Z flag set

BC, DE, HL destroyed

NOTE: Screenprinting is handled via @KEY, and registers are preserved.

**@KEYTABLE**

Lookup table for last row of keyboard (16 bytes).

Keys represented by this table in order:

ENTER

SHIFT ENTER  
CLEAR  
SHIFT CLEAR  
BREAK  
SHIFT BREAK  
UP ARROW  
SHIFT UP ARROW  
DOWN ARROW  
SHIFT DOWN ARROW  
LEFT ARROW  
SHIFT LEFT ARROW  
RIGHT ARROW  
SHIFT RIGHT ARROW  
SPACEBAR  
SHIFT SPACEBAR

**@CKDUL**

If DUAL is ACTIVE, character sent to printer buffer via @POUT.

NOTE: This call is made by @DISPLAY to check for DUAL operation.

**@DLON**

A call here copies bit 5 from @FLAGA to bit 7. If DUAL is ON, but INACTIVE, it will be made ACTIVE.

**@DLOFF**

Will DEACTIVATE DUAL, but not turn it off.

NOTE: @DLOFF is used when the message:

**Reading/Writing/Verifying Track x, Sector x**

is being displayed to prevent the repeated messages from being sent to the printer. Thus if dual is ON, and a directory is being read, for example, the directory listing will be sent to the printer, but not the READING messages.

@DLON will reinstate DUAL if it was ON prior to @DLOFF.

#### @SPOOL

This is the printer spooler driver. It is called under interrupt service. If (@PRBUFF) is zero, then there are no characters in the buffer, and nothing is done. If there ARE characters in the buffer, the shift @ key is checked. If pressed then the buffer is emptied.

#### @POUT

Send a byte to the printer buffer to be spooled.

Entry: A = character to be printed

Exit: ALL registers AND Flag are preserved

**NOTE:** This does not PRINT the byte, but merely inserts it into the printer buffer to be printed later under interrupt service. If the buffer is FULL (400H bytes), then this routine will wait till space is available.

#### @XREAD

Reverses status of IBM read, then jumps to @READNS.

Entry/Exit: Same as @READNS

#### @READNS

Read a disk sector into a memory buffer.

No SEEK operation is performed, so the drive head MUST be positioned over the correct track. This routine will read a NON-STANDARD sector.

Entry: D = Track

E = Sector

BC => Buffer

(@DRIVE) = bit pattern for drive to be read

Exit: BC => Buffer+100h (set for next READ)  
HL is destroyed  
Z = OK  
NZ = Bad, error status in A

#### **@WRITENS**

Write a disk sector from memory buffer.

Same as @READNS, but will WRITE a disk sector.

#### **@TREAD**

Reverses the density of the drive, then jumps to @READ

#### **@READ**

Normal call to read a standard disk sector.

Head is positioned to the correct track before the read operation is performed.

Entry/Exit same as @READNS.

#### **@TWRITE**

Same as @TREAD, except sector is WRITTEN to the disk.

#### **@WRITE**

Normal call to write a standard disk sector.

Seek operation is performed to position to correct track.

Entry/Exit same as @READNS.

**NOTE:** In all the above Disk I/O routines, the register contents for Entry/Exit are the same. Normally, calls to @READ and @WRITE will be used for standard diskettes. Automatic density recognition can be achieved as follows:

CALL @READ ;attempt to read the sector

```
CALL NZ,@TREAD ;if bad, try other density
CALL NZ,@BADRD ;bad in both densities
JP    NZ,SUBMENU;abort function
```

**@FLIPDEN**

Reverses the density of drive.

Entry: (@DRIV) = Binary drive number

Exit: A is destroyed

**@READ1**

Vector for @READ, perform SEEK operation.

**@READ1S**

Vector for @READNS, SEEK operation bypassed.

**@RDTYPE**

Byte to be used for READ operation.

Mod I single density:

88H = IBM format read

80H = Non-IBM format read

Mod III / Mod I double density:

80H = IBM format read

**@WRITE1**

Vector for @WRITE. A SEEK is performed.

**@WRITE1S**

Vector for @WRITENS. SEEK operation bypassed.

**@WRTYPE**

Byte to be used for sector write operation.

Mod I / Single - Double Density:

A8H = STANDARD address marks

A9H = READ PROTECTED address marks

Mod I / Single Density:

AAH = DELETED DATA address marks

ABH = USER DEFINED address marks

Mod III:

A0H = STANDARD address marks

A1H = READ PROTECTED address marks

#### @SEEK

Move drive head to specified track.

Entry: D = Track to move head to

Exit: Z = OK

NZ = Bad, error code in A

#### @SELECT

Select a drive and turn motor on.

Entry: (@DRIVE) = bit pattern for drive

Exit: Z = OK

NZ = Bad, error code in A

#### @DRIVE

Bit pattern for drive to operate on

Bit 3 = Drive 3

Bit 2 = Drive 2

Bit 1 = Drive 1

Bit 0 = Drive 0

NOTE: All routines involving (@DRIVE) may be set via @SETDRV

#### @DSKSLO

Delay loop to wait before valid status can be read from FDC.

#### @DELAY

Decrement BC till 0. If Highspeed clock is ACTIVE, the delay count will be doubled.

Entry: BC = Delay count (0 - FFFFH)

Exit: BC = 0, ALL other registers AND Flag are preserved.

#### @DRIIV

Binary drive number for current drive (0-3).

**@RESTORE**

Move head on a drive to track 0.

Entry: (@DRIVE) = bit pattern for drive  
to be used

Exit: HL = destroyed

Z = OK

NZ = Bad, error code in A

**@MOVCOMM**

Common routine for all drive head motion commands.

Entry: (@DRIVE) = bit pattern for drive  
to be used.

A = command to be issued.

Exit: HL = destroyed

Z = OK

NZ = Bad, error code in A

**@MOVEHEAD**

Issue head motion command to FDC, and wait till done.

Entry: (@DRIVE) = bit pattern for drive.

A = command to be issued

HL = 37ECH (if Mod I)

Exit: Z = OK

NZ = Bad, error code in A

NOTE: All of the above head motion commands are handled normally via calls to @READ and @WRITE.

**@STEPIN**

Move head on a drive IN one track (away from track 0).

Entry: (@DRIVE) = bit pattern for drive

Exit: HL is destroyed

Z = OK

NZ = Bad, error code in A

#### @STEPOUT

Move head on a drive OUT one track (toward track 0).

Entry/Exit: same as @STEPIN

#### @DOSEEK

Move head to track specified by D register.

Vector from @SEEK if head NOT already on specified track.

#### @SETDRV

Set @DRIVE and @DRIV for future table/disk operations.

Entry: A = Binary drive number (0-3)

Exit: A = Binary drive number IN ASCII

(@DRIVE) = bit pattern for selected drive

(@DRIV) = binary drive number supplied at Entry

#### @DRVASC

Get current drive number IN ASCII.

Entry: (@DRIV) = binary drive number

Exit: A = drive number in Ascii

#### @TASK

Interrupt service (background task).

Following operations are performed every 25 ms in the Mod I and every 33 ms in the Mod III:

1. Check if BREAK key is pressed
2. If NO, then goto step 6.
3. If YES, and NO SHIFT KEY, do step 5, then JP 4018H
4. If YES, and SHIFT KEY, do step 5, then JP 4015H. NOTE: 4015H jumps to

- master menu, 4018H returns to sub-menu
5. Print <BREAK>, wait till key is released, and return
  6. If master menu is displayed, change + to !
  7. If ALIVE is ACTIVE, change corners on video
  8. If any bytes in printer buffer, send to the printer

#### @SETNMI (Mod III only)

Setup routine for disk I/O for non-maskable interrupts.

#### @MASTER

In the event that the return vector to a sub-menu (4018H) is invalid, a jump here will be forced to initialize the @WHERE vector and normalize the pointers. Control will then be passed to @MENU.

#### @MENU

Entry point to MASTER MENU.

#### @MENWHR

Lookup table to get jump vector for selection from the master menu. First byte in table is the # of entries present.

Each entry in the table (and all menu tables), is 2 bytes long, and the table is terminated with 0. Corresponding table entry is calculated. 2 bytes indicate the address where control is to be passed.

#### @GETSEL

After printing a menu/sub-menu, DE points to a table of allowable responses and jump vectors.

DE => table of input/vectors as above

Prints 2 linefeeds, then asks "Selection?"

Call @GETSTR, allow 1 character maximum.

If "L" is entered, get LAST from table and return

If ENTER is pressed alone, jump to first address in table. Otherwise, put byte in LAST, and compare to table.

If match is found, jump to appropriate routine, else ask again.

#### @GOTABLE

Locate byte in A in table, and jump to appropriate vector.

Entry: DE=> table of 3 byte entries, 0 terminator (first byte is number to match, followed by a 2 byte address in the case of a match)  
A = number to locate in table

Exit: If match is found, return address is popped off the stack and a jump is made to the corresponding vector  
If no match is found, control will return to caller.

#### EXAMPLE of use of @GOTABLE

```
EXAMPLE RST 8      ;call display driver
        DEFB 10      ;linefeed
        DEFM 'Choice ? ' ;prompt
        DEFB 0       ;message terminator
        LD   B,1      ;allow one key input
        RST 10H      ;get keyboard input
        JR   Z,EXAMPLE ;nothing, try again

        LD   DE,TABLE ;point to table of
                      ;responses/vectors
        CALL @GOTABLE ;check if a match
        JR   EXAMPLE  ;will not return if a
                      ;match is found
```

**@ZAP**

Entry point to ZAP sub-menu

**@ZAPWHR**

Table of jump vectors for reply to Zap menu  
(used by @GOTABLE).

**@SETUPS**

Each sub menu calls here to update (@WHERE)  
for return vector.

The stack is initialize to the starting area.

Entry:      A = sub menu number

- 0 = master menu
- 1 = @ZAP
- 2 = @PURGE
- 3 = @FORMAT
- 4 = @COPY
- 5 = @REPAIR
- 6 = @TAPE
- 7 = @MEMORY
- 8 = @FILES
- 9 = @CONFIG

**@INKEY**

Call @KEY, and add debounce delay.

NOTE: This is the normal call to strobe the keyboard.

**@GOBACK**

Display "Press <ENTER> to continue." message,  
wait for the enter key, then return to sub-menu  
via @RETURN

**@RETURN**

Takes byte from @WHERE (sub-menu number),  
loads DE with @RETADD and calls @GOTABLE. If  
a match is found, then a jump is made to the

appropriate sub-menu. If no match, and jump to @MENU00 to re-normalize the @WHERE vector.

**NOTE:** This is the normal exit from ALL routines in SU+

**@WHERE**

Number of the current sub-menu. See @SETUPS for details on values found here.

**@RETADD**

Lookup table of vectors to sub-menus. Used by @RETURN.

**@PRESS**

Display "Press <ENTER> to continue " message, wait for enter key, then return to caller.

**@DISDSK**

Entry point to Display Disk Sectors routine.

**@DISTBL**

Lookup table for responses to "paging mode" in Display Disk Sectors.

**@NEXSEC**

Advances DE to next sector on the disk.

Entry: D = Track

E = Sector

Exit: DE = next track sector

A is destroyed

C = disk limits have been exceeded (D >= (@TRACKS) )

NC = OK

**@DOWNSEC**

Advances DE BACKWARDS to next sector on the disk.

Entry: DE == Track/Sector

Exit: DE = next sector BACKWARDS  
A is destroyed  
If DE is on the FIRST sector on the disk,  
nothing done

#### @TRKEND

Returns the highest sector number on current track.

Entry: D = current track  
Exit: A = highest sector number on current track

#### @FIRSTS

Load E with first sector on track indicated by D.

#### @ADDR20

Sector numbers are read from the disk 20 time via @ADDR.

Numbers are stored sequentially starting at @DAMBUFF.

Used to locate the highest/lowest sector on a track.

Entry: (@DRIVE) = bit pattern for selected drive  
Exit: A, B, HL, IX are destroyed  
20 sector numbers from current track stored at @DAMBUFF

#### @GETDAT

Get "Drive, Track, Sector ? " from keyboard.

Entry: NONE  
Exit: (@DRIVE) and (@DRIV) valid for selected drive.  
DE = Track/Sector

**NOTE:** If drive is not specified, then 0 is used.  
If track/sector not specified, then the first sector on the disk is used.

**@ASCII (RST 18H)**

Convert binary number to ascii.

Entry: A = binary number to convert

Exit: ACB = ascii number

**EXAMPLE:**

```
LD    A,127 ;start with binary number
RST    18H ;convert to ascii
LD    (NUM),A ;store MSB in string
LD    (NUM+1),BC ;store rest in string
RST    8 ;display the number
NUM   EQU    $
DEFM  'xxx' ;where it goes
DEFB  0 ;message terminator
```

**@POSHL**

Parse through string, skip spaces and commas.

Position HL to first valid character.

Entry: HL => string

Exit: A = first non-blank, non-comma  
character

HL => first character

Z set if first character is  
terminator (0DH)

**@MOVE**

Move block of data. Blocks may overlap.

Entry: HL => Source address of data

DE => Destination address of data

BC = Length of data to be moved

Exit: A is unused only

**@VALUE**

Extract value from input string.

Entry: HL => string

Exit: BC = value of input

HL => terminator (space, comma,  
C/R)

C = invalid number

**@PURGE**

Entry point to Purge Utility

**@PURWHR**

Lookup table for responses to Purge Utility Selection.

**@FORMAT**

Entry point to Format Utility.

**@FMTWHR**

Lookup table for Format Utility Selection.

**@COPY**

Entry point to Disk Copy Utility

**@CPYWHR**

Lookup table for Disk Copy Selection.

**@REPAIR**

Entry point to Disk Repair Utility

**@REPWHR**

Lookup table for Disk Repair Selection.

**@MEMORY**

Entry point to Memory Utility

**@MEMWHR**

Memory Utility lookup table for selection.

**@TAPE**

Entry point to Tape Utility.

**@TAPWHR**

Lookup table for Tape Utility Selection.

**@FILES**

Entry point to File Utility.

**@FILWHR**

Lookup table for File Utility Selection.

**@EXIT**

Entry point to EXIT program routine.

**@DEAD**

Executes a RST 0 on the Mod III, or a HALT on the Mod I.

**@MOUNTSYS**

Asks for SYSTEM disk to be mounted, and waits for ENTER key to be pressed. Video line is then cleared.

- Entry:           (@DRIV) = binary drive number to be mounted
- Exit:           A is destroyed

**@ONEKEY**

Waits for ENTER key to be pressed, then clears the current line the cursor is on.

- Entry:           NONE
- Exit:           A is destroyed

**@MOUNTSRC**

Asks for SOURCE disk to be mounted, and waits for ENTER.

- Entry:           (@DRIV) = binary drive number for disk to be mounted on.
- Exit:           A is destroyed

**@MOUNTDES**

Asks for DESTINATION to be mounted, and waits for ENTER.

Entry: (@DRIV) = binary drive number to be mounted.

Exit: A is destroyed.

#### @DRVCOMM

Executes a common routine for all active drives.

Entry: Bit 5 set in (IY+5) indicates active drive.

DE = Address of common call for all drives.

BC = Address of exit to make when all drives completed.

IY => DCT for current drive

Exit: Made to address supplied by BC.  
BC, DE A are destroyed

NOTE: The ENABLE bit in @DCTx is set properly by making a call to @GETDRV\$. HL, IX are not used, and may be passed along.

@DRIVE and @DRIV are updated to each current drive before the call is made to each subroutine.

#### @POSA

Current drive counter used by @DRVCOMM. User subroutines may alter the contents of @DRIVE and @DRIV if needed, and they will be corrected by @POSA when the current drive is completed.

NOTE: (@POSA) and (@DRIV) are equal when entry is made to the subroutine for each drive.

#### @INITDRV\$

Sets all drives in DCT's as ACTIVE. (Sets bit 5 (IY+5))

ALL registers are preserved.

**@INITDRV**

Sets all drives in DCT's as INACTIVE (Resets bit 5). ALL registers are preserved.

**@SHOCONFG**

Inserts SOFT CONFIGURE setting onto video display.

Registers AF, BC, DE, HL, IY are used.

**NOTE:** This displays the SETTINGS ONLY, not the headings.

**@SETYES**

Loads "Y" or "N" into A depending on Z flag.

Entry:           Z flag set/reset as applies.

Exit:           If NZ, then A = "Y"

                  If Z, then A = "N"

**NOTE:** This is useful for inserting ascii Y or N into a string depending on the setting of a flag bit. EXAMPLE:

BIT 3,(IY)       ;check for status bit

CALL @SETYES     ;load A with Y or N

LD (MSG),A       ;put into string

**@DSTAT**

Checks status of disk drive (disk mounted and door closed).

Entry:           (@DRIV) and (@DRIVE) indicate the drive to be checked.

If drive NOT READY, message to correct the problem are displayed.

Exit:           Z = OK  
                  NZ = "Skip this drive" was selected

**@STAT**

Checks status of disk drive.

Entry: (@DRIVE) and (@DRIIV) indicates drive number.  
If drive NOT READY, no message is displayed.

Exit: Z = OK, drive mounted and door closed.  
NZ = Drive NOT READY, HL => error condition message.

#### @SELDEN (Mod I only)

Selects the proper FDC controller in the Mod I.

Entry: (@DRIVE) and (@DRIIV) = drive number to use.

Exit: If single density selected then the 1771 controller is selected.  
If double density selected then the 1791 controller is selected.  
A and A' are used.

**NOTE:** This routine only applies if a hardware double density modification has been installed. The Mod I has TWO floppy disk controllers (FDC) for double density operation. The Mod III has ONE controller that performs both single and double density. Hence this routine is not needed in that machine.

#### @ZBUFF

Zeros a memory buffer.

Entry: BC => Address to be cleared.

Exit: 256 bytes of zeroes written to the buffer.  
ALL registers are preserved.

#### @RXFER

Handshaking routine used for disk I/O.

Entry:      Mod I: HL = 37ECH  
                DE = 37EFH  
                BC => Memory buffer  
Mod III HL => Memory buffer  
                D = (@DRIVE)  
                E = 2  
                B = 0 (byte counter)  
                C = F3H (port address)  
Exit:        Data transferred to address supplied.  
                (@RESULT) = non-masked result of operation.  
                (@TEMPFF) = address +1 of last byte transferred.

NOTE: Read command must be issued BEFORE calling this. This is normally called via @READ.

#### @WFER

Transfer data from FDC to memory buffer.

Entry/Exit: same as @RXFER, except data is transferred TO disk.

#### @ADDR

Reads 1 address mark from disk on current track.

Entry:        (@DRIVE) is valid for drive to be used.

Exit:        HL=> 6 byte string of values read in: Track, Head, Sector, Length, and 2 CRC bytes

All other registers preserved except A.

Z = OK

NZ = Bad, error code in A.

**NOTE:** this is called by @DISDSK to determine the REAL track on a disk, versus the RELATIVE track.

```

LD   A,1          ;use drive 1
CALL @SETDRV      ;set it up for use
CALL @ADDR        ;read address mark from
                   ;disk
LD   A,(HL)       ;get REAL track as read
                   ;from disk

```

#### @UCASE

Convert byte in A to upper case.

Entry:           A = byte to be converted.  
 Exit:           A = upper case equivalent.

**NOTE:** Only bytes from a-z (lowercase) are converted.

#### @FIGDRV

Used to interpret a drive number from an input string.

Entry:           HL => input string  
 Exit:           If no drive is specified, then  
                   drive 0 is defaulted.  
                   If DOS specifier is appended, then  
                   DCT +7 is set.  
                   If =tks is specified, then  
                   DCT+0/+1 is set.  
                   NC = OK  
                   C = invalid input for drive  
                   number.

#### @DDOSFIX

DE (track/sector) is adjusted to DISK RELATIVE sector if ND80 DD, Multidos, or DoubleDOS sector is being read.

Entry:           DE = Track/Sector to be read

Exit:              Original value left on stack, and  
                    DE is returned with the correct  
                    RELATIVE value.

```
LD   DE,0          ;track 0/ sector 0
CALL @DDOSFIX      ;compute RELATIVE
                     sector
POP  HL           ;DE = RELATIVE sector,
                     HL = REAL sector
```

NOTE: This routine is called by @READ and @WRITE to adjust to the correct value for the corresponding DOS.

#### @TASKDRV

If a disk sector is NOT FOUND on a bad read, this routine will insert a 0 into the current track for that drive. This forces an @RESTORE call to be made the next time that drive is read to position to the correct track.

Entry:            (@TASKDRV) = C9H (RET) if NO error

                  (@TASKDRV) = 00H (NOP) if error HAS occurred.

Exit:            If error HAS occurred, then the current track for the drive specified by (@DRIV) is updated to 0

                  If NO error, then nothing is done.

NOTE: This vector is handled normally by the system, and is called by @RETURN at the completion of each routine.

#### @TURNNSPEED

Turns high speed clock ON or OFF appropriately.

Entry: Bit 6 of @FLAGA = set = turn speed ON  
          Bit 6 of @FLAGA = reset = turn speed OFF

Exit: If Bit 6 = 0, then @SPEEDOFF is executed.  
       If Bit 6 = 1, then @SPEEDON is executed.

**@SPEEDOFF**

Eight bytes of instructions to turn high speed clock OFF.

**@SPEEDON**

Eight bytes of instruction to turn high speed clock ON.

The instructions may be modified via CONFIGURE commands. Normally only A is used, but may be user definable to use other registers.

**@CONFIG**

Entry point to SOFT CONFIGURE routine.

**@FIXFLG**

Sets a bit of a flag byte.

Entry: DE => flag byte to be used.  
          C = mask byte of bits to be RESET.

Exit: (DE) are updated

Exit is made via @POSHL to move HL to next byte in string.

LD DE,@FLAGA	;point to flag A
CALL @YESNO	;set Z or NZ if A = ;"Y" or "N"
LD BC,807FH	;bit 7 mask bytes
JR Z,CONT	;skip if answer is Y

LD B,0	;else set bit 7 to be ;RESET
CONT EQU \$	;label CONT
CALL @FIXFLG	;bit 7 (DE) is SET or ;RESET if "Y" or "N"

**@ASKFIG**

HL = video location to get keyboard input.  
(Normally called through @CONFIG)

Entry: HL = video location for prompt  
Exit: Left 2 characters on EACH video  
line is erased.  
=> prompt displayed at location  
(HL)

Exit condition for @GETSTR applies, 60 chars  
maximum.

**@YESNO**

Checks if A contains "Y" or "N"

Entry: A = character to check  
Exit: Z = A contained "Y" or "y"  
NZ = A contained "N" or "n"  
C = neither of the above was  
found

**@SHOW**

Displays block of data in memory to the screen  
Entry: (@ADDRESS) = starting address to  
display  
Exit: 256 bytes of data are displayed  
to the screen  
If decrypting is specified, it will  
be displayed.  
Screen will contain HEX and  
ASCII representations of data.

This routine is normally called through display disk sectors, file sectors, memory, or build track. Any address may be displayed.

#### @HEXCV (RST 20H)

Convert binary number in A to HEX ASCII.

Entry: A = binary number to convert

Exit: BC = HEX ASCII representation  
(LSB, MSB)

```
LD    A,16          ;number to be converted
RST  20H          ;convert to ASCII
LD   (MSG),BC      ;insert into string
RST  8           ;display the message
MSG  EQU  $        ;label MSG
DEFM 'xx'         ;actual string to display
DEFB 0            ;terminator
```

#### @SHOWST

Displays error message from disk I/O.

Entry: A = error condition

C = error occurred during READ  
operation

NC = error occurred during WRITE  
operation

Exit: Message is displayed to the  
screen

ALL registers are preserved.

This is normally called through @BADRD,  
@BADWRT.

#### @SHOWWH

Displays current Drive, Track, and Sector.

Entry: (@DRIV) = current drive number  
DE = track sector

Exit: "Drive x, Track xxx, Sector xxx."  
is displayed.  
ALL registers are preserved.

```
LD A,1           ;drive 1
CALL @SETDRV    ;set it up
LD DE,0          ;track 0, sector 0
CALL @READ       ;read the sector
CALL NZ,@SHOWWH  ;displays drive, track,
                  ;sector
CALL NZ,@SHOWST  ;displays error condition
```

**@SHOWLF**

Displays source of data. Use with @SHOW.

If data from MEMORY, then addresses are displayed.

If data from DISK, then drive, track, sector displayed.

If data from FILE, then drive, track, sector, filename and relative sector in file are displayed.

**@MODTYPE**

Fifteen byte string of Modify Mode number bases ("HEXDECBIN OCTASC"), used by @SHOWLF to display current setting.

**@MODDRV**

Ascii string for "DRV" (Used by @SHOWLF).

**@MODMEM**

Ascii string for "MEM" (Used by @SHOWLF).

**@MODTRK**

Ascii string for "TRK" (Used by @SHOWLF).

**@MODTRU**

Ascii string for "TRU" (Used by @SHOWLF).

**@MODSEC**

Ascii string for "SEC" (Used by @SHOWLF).

**@MODDAT**

Twelve byte string for the different data address marks used by @SHOWLF ("STDRPTDDTUDF")

#### @MXDDE

Ascii string for "ISD" (Used by @SHOWLF).

#### @MODIFY

Modify mode entry point (Call @SHOW first to display).

#### @MODFIX

Used by the flashing cursor routines in the modify mode to restore the 3 bytes being flashed to their normal setting.

Entry: DE => current buffer address  
IX => current video location on HEX side of display.

IY => current video location on ASCII side of display.

Exit: Byte from (DE) is displayed to (IY), (IX), and (IX+1)  
Byte is adjusted first if encrypting is specified.

#### @MODTBL

Table of vectors to be used by the modify mode when keys are pressed.

#### @BLKS

Three bytes of 8FH (graphic blocks). Used as the cursor ON Character in the modify mode.

#### @DECODE

Get keyboard input, and set decoding parameters (normally entered through @MODIFY when @ is pressed)

#### @UPDATE

Entry point to update data when ENTER pressed in modify mode.

If data is from disk, it will be written back. If data is from memory, it is already updated.

#### @BDRDCLS

Clears the screen, calls @BADRD, clears the screen again and returns. A call is made here instead of @BADRD if the screen is filled with data, such as in Display Disk Sectors.

#### @BDWTCLS

Same as @BDRDCLS, except a call is made here for Write error.

#### @BADRD

Used after a disk Read error to display where the error occurred, what type of error it was, and prompts for "Retry, Skip, Continuous, Nonstop, or Quit."

Entry:           A = error code  
                  DE = Track, Sector  
                  (@DRV) = binary drive accessed

If the CLEAR key is pressed, turn off NONSTOP and CONTINUOUS

Exit:           If QUIT is selected, program branches to 4018H and returns to last sub-menu.

If SKIP is selected, B register (buffer pointer) is incremented, Z flag is set.

If NONSTOP or CONTINUOUS is set, program returns with NZ flag set, and turn off prompting mode.

If RETRY is selected, NZ flag is set.

ROUTINE	EQU	\$	;start of routine
LD	A,1		;use drive 1

```

CALL @SETDRV          ;set it up
LD    DE,0             ;track 0, sector 0
LD    BC,@BUFFER       ;where to read it
                      ;in
CALL @READ            ;read a sector
CALL NZ,@BADRD        ;if no good, go
                      ;error
JR    NZ,ROUTINE      ;if RETRY, read
                      ;again

```

**@BADWRT**

Same as @BADRD, except called after a bad Write operation.

**NOTE:** @BADRD and @BADWRT can be DISABLED by inserting a C9H (RET opcode) into the first byte. Insert a 00H (NOP opcode) to ENABLE it.

**@INITBAD**

Turns off CONTINUOUS mode in @BADRD and @BADWRT.

Entry:	NONE
Exit:	A is destroyed

**@INITBD1**

Turns off NONSTOP, and enables @BADRD and @BADWRT.

Exit:	A is destroyed
-------	----------------

**@INITBD2**

Disables @BADRD and @BADWRT.

Exit:	A is destroyed
-------	----------------

**@ADJBYTE**

Adjust a byte to the current DECRYPTING mode setting (decrypting defined by setting @DISP1, @DISP2, and @DISP3)

Entry:	A = byte to be adjusted
Exit:	A = adjusted byte

A' is destroyed

#### @CKCONF

Check for Dos specifier.

Entry:      HL => bytes to be interpreted.

Exit:      NZ = invalid selection, ALL  
              registers preserved

Z = selection OK, DCT updated to  
new dos type.

LD    A,0                          ;use drive 0

CALL @SETDRV                     ;set it up for use

CALL @CKCONF                     ;get byte from @DTTBL

#### @AASHOW

Saves current registers, forces prime registers,  
and call @SHOW to display a memory buffer.

Entry:      (@ADDRESS) = address to be  
              displayed.

Exit:      BC, DE, HL, IX and IY are  
              preserved  
              AF, and ALL alternate registers  
              are used

#### @CKTRAKS

Checks input string for track count specified.

Entry:      HL => input string

Exit:      If (HL) = and equal sign (=), then  
              @VALUE is extracted from  
              following bytes.

If VALUE is OK (no carry), then  
it is inserted into the DCT table  
for the current drive.

This routine is called normally through  
@FIGDRV.

#### @SHOREAD (RST 28H)

Displays "Reading Drive, Track, Sector", calls @READ.

Entry:           (@DRIVE) valid for current drive  
                   DE = Track/ Sector to read  
                   BC => Buffer address  
 Exit:            Z flag is set

If @READ is successful, then a RET is executed. If @READ is Bad, then a call to @BADRD is made. If SKIP is selected, the a call to @ADDCOUNT is made to bump an error counter.

```

LD   A,0           ;select drive 0
CALL @SETDRV       ;set it up
LD   HL,20         ;20 sectors to read
LD   DE,0          ;track 0, sector 0
CALL @INITCNT     ;zero the error counter
LOOP EQU $         ;label LOOP
LD   BC,@BUFFER    ;where to load the data
CALL @MREAD        ;multiple read routine
LD   A,H           ;check for any more
OR   L              ;set flags for HL
JR   NZ,LOOP        ;some more left
CALL @SHOCNT       ;display error counter
RST  8              ;display message
DEFM 'Errors.'     ;message
DEFB 0              ;terminator

```

#### @SHOWRITE (RST 30H)

Same as @READ, except @WRITE and @BADWRT are the vectors.

#### @SHOVERF

Same as @READ, except "Verifying Drive, Track, Sector" is displayed.

#### @SHOVERX

Same as @SHOVERF, except a call is NOT made to @BADRD in the case of a disk error. Two

attempts are made at a successful read. This routine is used when formatting a disk to detect bad sectors.

**@SHOFMT**

Displays "Formatting Track xxx", and writes track to disk.

Entry: Mod I, (37EDH) = current track number  
Mod III, port (F1H) = current track number  
(@FMTBUFF) = address of format data

Exit: Z = successful write operation  
NZ = bad, error message is displayed.

**@VERSEC**

Verify Disk Sectors entry point.

**@SHOACNT**

Displays Counter B.

**@SHOCNT**

Displays Counter A.

**@XCOUNT**

Counter A string.

**@XACOUNT**

Counter B string.

**@INTACNT**

Zeros counter B.

**@INTCNT**

Zeros counter A.

**@ADDACNT**

Bumps counter B.

#### @ADDCNT

Bumps counter A.

**NOTE:** The above counters may be used as follows: Counter A is normally used as a disk I/O error counter. When calling @SHOREAD, etc. @ADDCNT is called whenever a disk sector is skipped (bad.) Counter B is user definable. When calling the bump counter routines, ALL registers are preserved. When the counters are displayed, ALL registers are preserved, and the cursor is left with one space following the number. A linefeedddd displayed BEFORE the number.

See @SHOREAD for an example of use.

#### @GETCNT

Prompts for "Sector Count ?"

Entry: (@DRIVE) is valid for current drive.

Exit: HL = number of sectors specified.  
If no input is supplied (ENTER pressed only), then HL will be loaded with the total number of sectors on the diskette.

A = L register (LSB of sector count)

#### @CNTOTAL

Computes the total number of sectors on a disk.  
This is normally called through @GETCNT.

Exit: HL = sector count on the diskette

```
LD   A,0          ;use drive 0
CALL @SETDRV      ;set it up
CALL @CNTOTAL     ;compute total
```

#### @MVERIFY

Multiple sector read routine. Same as @MREAD, except "Verifying" is displayed.

#### @MREAD

Multiple sector read routine. @SHOREAD is called to display the current sector being written.

Entry: (@DRIVE) is valid for current drive

DE = Track, Sector to begin

HL = Number of sectors to read

Exit: @ADDCTN called once for each sector SKIPPED if read error.

HL = number of sectors remaining to be read.

Top of memory is checked after each sector. Thus if more sectors are specified than can be held in memory, the routine will exit, and HL will contain the count of sectors remaining.

See @SHOREAD for usage of this call.

#### @MWRITE

Multiple sector Write operation. Same as @MREAD, except sectors are written to the disk from memory.

#### @COMSEC

Compare Disk Sectors entry point.

#### @COMPARE

Compares two strings.

Entry: HL => source string

DE => compare string

B = length to compare

Exit: Z = all bytes match

HL => source string + length

DE => compare string + length

B = 0

NZ = mismatch  
HL => first mismatch source  
DE => first mismatch compare  
B = # bytes remaining to compare

#### @IFSAME

Asks if disk mounts are to be prompted, and sets @MFLAG according to keyboard input.

Entry:

NONE

Exit:

If NO is selected, (@MFLAG) = -1  
If YES is selected, (@MFLAG) = 0

#### @SMOUNT

Prompts for source diskette to be mounted.

Entry:

NONE

Exit:

If (@MFLAG) = -1 (NO), nothing is done.

If YES, then prompt is issued to mount SOURCE diskette on the current drive, and program waits for ENTER key.

#### @DMOUNT

Same as @SMOUNT, except DESTINATION disk is prompted for.

NOTE: See @SSETUP and @DSETUP

#### @DRV SAME

Source and Destination drives are compared.

Entry:

(@SDRIVE) and (@DDRIVE) are valid.

Exit:

Z = same drives

NZ = different

#### @PAUSE

Check for pause key. (spacebar)

Entry:

NONE

Exit:            If SHIFT SPACEBAR is pressed, wait till it is released.  
                   If SPACEBAR is pressed, wait till ENTER key is pressed.  
                   A is destroyed.

**@COPSEC**

Copy Disk Sectors entry point.

**@ZERSEC**

Zero Disk Sectors entry point.

**@WRITETR**

Writes track of data from memory to disk.

Entry:            BC => buffer where track data is.  
                   (@DRIVE) is valid for selected drive.  
                   Head is over the correct track.  
 Exit:            Z = OK  
                   NZ = Bad, error code in A. (5 tries made)

**@DAMARKS**

Read Data Address Marks entry point.

**@BUFFEND**

Checks for buffer at top of memory.

Entry:            BC => current buffer pointer  
 Exit:            Z = at top of memory  
                   NZ = more buffer left

```

LOOP EQU $      ;label LOOP
PUSH HL       ;save counter
RST 28H      ;read a sector
POP HL       ;restore counter
JR NZ,QUIT   ;Quit if no good
CALL @NEXSEC ;bump sector pointer
DEC HL       ;check for # sectors
             ;completed

```

```

LD   A,H          ;check for any bits on
OR   L
RET  Z          ;all sectors done
CALL @BUFFEND    ;any memory left?
RET  Z          ;done for now
JR   LOOP        ;else go some more

```

**@EXCSEC**

Exchange Disk Sectors entry point.

**@SECDATA**

Exchange Disk Sectors entry point.

**@GETBYTE**      **620C (61E7)**

Asks for "Relative Byte ?".

Entry:        NONE

Exit:          L = relative byte 0-FFH  
                 BC, H, A are destroyed.

**@GETBCNT**

Asks for "Byte Count ?".

Entry:        NONE

Exit:          BC = byte count (defaults to 256  
                 if ENTER pressed)  
                 A is destroyed, HL, DE are  
                 unchanged.

**@IFCLEAR**

Checks for CLEAR key being pressed.

Entry:        NONE

Exit:          C = key is pressed  
                 NC = key NOT pressed  
                 A is destroyed

**@SWAPDAM**

Exchange bytes in @DAMBUFF

Entry:        NONE

Exit: First 128 bytes in @DAMBUFF are exchanged with the Second 128 bytes.

This routine is called by @EXCSEC to swap the contents of the Data Address Mark buffer.

A is destroyed.

#### @SSETUP

Setup parameters for source drive.

Entry: (@SDRIVE) contain the current drive.

Exit: (@DRIVE) and (@DRIV) are set for that drive.

If (@MFLAG) is 0, then a prompt for SOURCE disk is issued. The disk status is then checked for ready via @STAT.

Z = drive is mounted and door is closed.

NZ = drive not mounted, and SKIP was selected.

#### @DSETUP

Setup for Destination Drive. Same as @SSETUP except (@DDRIVE) is valid for drive to be used.

#### @STRSER

String Search through disk entry point.

#### @MSTRSER

String Search through memory entry point.

#### @ASKSTR

Asks for string for above 2 routines.

Entry: NONE

Exit: Input MUST be made, ENTER alone is not accepted.

If first characters are # or ##, then the VALUE of the following numbers are inserted into the string.

If @ is specified, then the resulting string is decrypted to whatever the current settings are (Call is made via @ADJBYTE to encode the string.)

B = length of resulting input string.

HL => string

#### @ASKREPL

Asks for Replacement String, in conjunction with the above.

Entry: NONE

Exit: if ENTER pressed alone, then bit 2 of @FLAGA is RESET.

If input is supplied, then the string is interpreted in the same manner as @ASKSTRNG, and bit 2 of @FLAGA is SET.

#### @REVSEC

Reverse Disk Sectors entry point.

#### @SWAP

A single page of memory is reversed (256 bytes).

Entry: HL => page of memory to be reversed.

Exit: 256 byte block of memory is reversed.

A and A' are destroyed.

#### @SECSER

Sector Search routine entry point.

**@IDMARKS**

Read ID Marks entry point.

**@IDTABLE**

Lookup table for responses to key input in @IDMARKS.

**@CKADDR**

Reads ID address marks from disk.

Entry:       (@DRIVE) is valid for drive  
                Head is positioned over the  
                desired track.  
Exit:         NZ and NC means data is valid.  
                NZ and C means disk error during  
                ID read.  
                Z means track is possibly not  
                formatted, but no error occurred  
                reading the ID

**@ONEDRIV**

One drive is prompted for, and @SETUP is set up.

Entry:       NONE  
Exit:         If ENTER pressed alone, then  
                drive 0 is defaulted. If input is  
                supplied, then it is interpreted to  
                setup the drive number, dos  
                specifier if supplied, and track  
                count if supplied.

**@ACOMPARE**

Compares two strings, after being decrypted.

Entry:       DE => source string.  
                HL => dest string, to be  
                decrypted.  
                B = length to compare.  
Exit:         Z => strings match

A byte is taken from (DE), and a call is made to @ADJBYTE. Then, the resulting byte is compared to (HL)

#### @QCOMPAR

Compare two strings, pass over ? symbols.

Entry:            HL => source string

                  DE => destination string

                  B = number of bytes to compare

Exit:            Z = all bytes match

                  NZ = does not match

If any ? symbols are found in the source string (HL=>), then they are not compared. Thus, all ? characters in the source string will match anything in the dest string. This is used with the String Search routines.

#### @DISMEM

Display Memory entry point.

#### @DOADDR

Display memory pointed to by BC.

Entry:            BC = address to be displayed.

Exit:            (@ADDRESS) = (BC), and the data is displayed to the screen.

#### @DISMTBL

Lookup table for responses to Display Memory.

#### @GETADDR

Prompt for "Address ?".

Entry:            NONE

Exit:            BC = Input value

If ENTER is pressed alone, then (@DEFADDR) is used as the default value.

#### @DOBUILD

Build Track to Memory entry point.

**@GETSES**

Prompt for "Start, End, Start ?".

Entry:        NONE

Exit:        (@TEMPO) = Start (default) =  
                  (@PGMEND)

                  (@TEMP1) = End (default) =  
                  (@TOPMEM)-1)

                  (@TEMP2) = Start (default) =

                  @PGMEND)

**@GETSE**

Prompt for "Start, End ?".

Entry:        NONE

Exit:        (@TEMPO) = Start (default) =  
                  (@PGMEND)

                  (@TEMP1) = End (default) =  
                  (@TOPMEM)-1)

**@MOVMEM**

Move Memory entry point.

**@EXCMEM**

Exchange Memory entry point.

**@REVMEM**

Reverse Memory entry point.

**@JUMPMEM**

Jump to Memory entry point.

**@FILLMEM**

Fill Memory entry point.

**@COMMEM**

Compare Memory entry point.

**@TESTMEM**

Test Memory entry point.

**@INPUT**

Input Byte from Port entry point.

**@OUTPORT**

Output Byte to Port entry point.

**@MEM2SEC**

Memory to Sectors entry point.

**@SEC2MEM**

Sectors to Memory entry point.

**@WRBUILD**

Write Format Track entry point.

**@MEM2TRK**

Memory to Track entry point.

**@TRK2MEM**

Track to Memory entry point.

**@GETDT**

Prompt for "Drive, Track ?".

Entry:           NONE

Exit:           (@DRIVE) and (@DRIV) are set (0  
                  is defaulted)

D = Input Track (0 is defaulted)

**@SFMTW**

Standard Format Without Erase entry point.

**@SFMT**

Standard Format entry point.

**@GETDRV**

**Prompt for "Drive(s) ?"**

Entry:        NONE  
Exit:        Input string is scanned for multiple drive inputs.  
                Bit 5 of the corresponding DCT +5 byte is set to activate the selected drive.  
                String is handled via @FIGDRV so parameters such as Dos specifier and track counts may be interpreted.

**@GETNMDT**

Prompt for Name, Date, and Password. (for formatting)

Entry:        NONE  
Exit:        (@GATBUFF+D0H) = 8 byte name  
                (@GATBUFF+D8H) = 8 byte date  
                (@GATBUFF+CEH) = 2 byte encoded password.

If ENTER alone is pressed for the above, the corresponding positions are not changed. Call @INITGAT for initial set up.

**@INITGAT**

GAT table is initialized.

Entry:        (@DRIV) is valid to fetch Dos specifier.  
Exit:        First the entire buffer (256 bytes) is filled with FFH (sector filled with 00H for TRSDOS III). Then, starting at @DEFGAT, 21 bytes are moved into the GAT table starting at @GATBUFF+CBH. Then an 0DH is left at (@GATBUFF + EOII)

NOTE: See @FILLGAT and @MAKEGAT.

#### @DEFGAT

Twenty-one bytes of GAT table initialization data.

The bytes and their corresponding locations in the GAT:

21H => @GATBUFF+CBH = 3.0 version number for SU+

0000H => @GATBUFF+CCH = 2 bytes of 0 (used in DOS+, LDOS, DoubleDOS)

42E0H => @GATBUFF+CEH = 2 byte encoded password

(@FMTNAME) => @GATBUFF+D0H = 8 byte disk name

(@FMTDATE) => @GATBUFF+D8H = 8 byte disk date

#### @FMTNAME

Eight byte string of the default name for Format.

#### @FMTDATE

Eight byte string of the default date for Format.

#### @SPACES

Thirty two spaces. (20H)

#### @FORMAT

Formats an entire disk.

Entry: (@DRIVE) and (@DRIV) valid for drive to be used.

(@STRTRK) is valid for track where formatting is to begin.

(@FMTYPE) = 1 if format without erase

(@IFVERF) = 0 if disk is to be verified, else 0  
Exit: If (@FMTYPE) = 2, then a RET will be executed after the disk is formatted without verifying. If anything else, the disk will be verified (if (@IFVERF) =0), then the BOOT and DIRECTORY will be written to the disk.

#### @STRTRK

Holds the number of the track where formatting is to begin.

The user may format any number of tracks, anywhere on a disk. For example, to format tracks 21-27 only, set the track count to 28, and the starting track to 21 before calling @FORMAT.

#### @FMTYPE

Flag which identifies the calling vector to @FORMAT.

- 0 = Standard format
- 1 = Format without erase
- 2 = Standard disk copy (returns after formatting)

#### @IFVERF

Flags if a Verify is to be performed on a formatted disk.

1 = skip verify phase, anything else will verify.

If (@FMTYPE) = 2, then NO verify is attempted, even if (@IFVERF) indicates a verify.

#### @BUILDTRK

Build a format track in memory.

Entry: (@DRIV) is valid for current drive  
(needed to fetch the correct dos

specifier byte from @TYPEA table).

(@FMTBUFF) is the address where the data is to be placed.

Mod I, (@37EDH) = current track number.

Mod III, (port F1H) = current track number.

Exit: Data is stored in to memory buffer at specified address.  
AF, BC, DE, HL are used.

### @MOVEIN

Takes a length and fill byte pointed to by DE, and fills into a memory buffer pointed to by HL.

Entry: DE => 2 bytes, length and fill byte

HL => memory buffer address

Exit: Address pointed to by HL is filled with bytes specified by 2 byte table pointed to by HL

HL => next address

DE => next byte in table

B = 0

A = fill byte used

LD	DE, TABLE	;point to table
LD	HL,@BUFFER	;point to buffer
CALL	@MOVEIN	;fill the buffer
JP	CONT	;continue here
TABLE	EQU \$	;table starts here
DEFB	33	;33 bytes to be filled
DEFB	07	;fill byte is 7

### @FILL

Fills buffer pointed to by HL with B bytes of A.

Entry: HL => buffer to be filled  
B = number of bytes to fill  
A = byte to fill with

Exit:            HL => next byte in buffer  
                  A = byte used for fill  
                  B = 0

```
LD   HL,@BUFFER    ;point to buffer
XOR  A             ;set Accumulator to 0
LD   B,0           ;# bytes to fill
CALL @FILL         ;fill it up
```

After this routine executes, HL = @BUFFER +100h, and @BUFFER through @BUFFER+OFFH will be filled with 0's.

#### @SORDER0

Table of information used in formatting SINGLE DENSITY.

#### @DORDER0

Table of information used in formatting DOUBLE DENSITY (not TRSDOS Mod III)

#### @DORDER1

Table of information used in formatting DOUBLE DENSITY (for TRSDOS Mod III)

#### @DORDER2

Table of information used in formatting DOUBLE DENSITY (for TRSDOS Mod I double den only)

NOTE: The above 4 tables contain the following information.

+0 = number of sectors on a track

+1+2 = address of table of the order of the sectors on a track

+3+4 = post-index gap length, fill byte

+5+6 = pre ID sync field

+7+8 = pre ID sync field

+9+10 = gap between ID and DATA fields  
+11+12 = pre DATA sync field  
+13+14 = pre DATA sync field  
+15+16 = post DATA gap length, fill byte

#### @ORDERS

Ten byte table of the order that sectors are to appear on a track when it is formatted.

#### @ORDERD0

Eighteen byte table of the order sectors appear on a double density (non-TRSDOS III) track.

#### @ORDERD1/2

Eighteen byte table of the order sectors appear on a double density TRSDOS Mod IDD/III track.

**NOTE:** The sectors on a track are not in sequential order for minimum read times. If you examine the tables, you will see that a track can be read in 2 revolutions on single density and 3 revolutions on double density.

See @ORDNEW for sector skewing information.

#### @HASDATA

Before a disk is formatted, an attempt is made to read track 0, sector 0 (sector 1 on TRSIII). If it is readable then a call is made here. An attempt is made to read the directory. If it is NOT readable, an appropriate message is displayed. If it IS readable, the directory name and date will be displayed. The user is then prompted to Continue or Quit. If quit is selected, then an exit is made to 4018H to return to the last sub-menu, else a RET is executed.

#### @CODE

Codes an ASCII string into a two byte password.

Entry:            HL => input string to be encoded.  
                  B = length of the input.  
                  (@TYPEA) bit 3 defines which  
                  encoding scheme to be used (Mod  
                  III TRSDOS or all others).  
Exit:            HL = 2 bytes password  
                  AF, BC, DE are destroyed

NOTE: All operating systems use the same  
encoding scheme except TRSDOS III and 2.7DD.

#### @GET10

If format without erase is selected, then a call  
is made here before formatting each track to read  
a full track of data into a memory buffer.

#### @PUT10

If format without erase is selected, then a call  
is made here after each track is formatted to  
write the previously found data back on the track.

These 2 calls are normally made through  
@FORMAT.

#### @ORDNEW

Shifts the current sector order table for sector  
skewing. If single density, 3 rotations are  
performed, else 4.

In order to make a disk readable in the fastest  
manner, it is important to consider the step time  
involved from track to track. A single turn of the  
disk takes 200 ms. At a 40 ms step rate (slowest),  
1/5 of the disk will turn by before the step  
command has completed. If the sectors appear in  
the same order on every track, then the remaining  
4/5 revolution must be made to come back around  
to the first sector on the next track. Using the  
technique applied here, the first sector on each  
successive track will be located 3/8 of a

revolution ahead of the last track, so that when the step command has completed, the first sector will be positioned properly.

**@NSBOOT**

NON-SYSTEM boot that is written to a disk during the format process. If the disk is booted on drive 0, then a message informing the user that the diskette does not contain a system is displayed, and the program halts.

**@BOOTENT**

Directory entry for BOOT/SYS, applied during format.

**@DIRENT**

Directory entry for DIR/SYS, applied during format.

**@SFMT3**

Normal exit address for format and backup to ask if another copy is requested. If NO, then an exit to 4018H is made to return, if YES, then another copy is made.

**@LOKIT**

Locks out current granule when error detected during verify phase on disk formatting.

Entry: DE = track/sector

Exit: GAT table located at @GATBUFF  
is updated to reflect the locked  
out granule.

DE = first sector AFTER the  
current granule.

**@FILLGAT**

Prepares a GAT table in memory.

Entry: HL => 256 byte buffer

Exit: All bytes to relative byte CDH are affected.  
Bytes CEH - FFH are unchanged.  
Complete GAT and Allocation tables are constructed.  
All tracks are marked as available.  
BC destroyed.

**@RELTKS**

Computes the RELATIVE track count of a diskette.

Entry: NONE

Exit: DCT+0 = relative track count of diskette.

NOTE: The TRUE and RELATIVE track counts for a diskette are always the same EXCEPT with DoubleDOS and ND80. The RELATIVE track count is computed as DISK RELATIVE SECTORS.

If the TRUE track count is desired (such as when formatting) then a call to @GETTKS should be made. The RELATIVE track count is needed when stepping through a disk via sector read/writes.

**@LOKGRAN**

Performs the actual GAT table track lockout.

Entry: DE = current track/sector

Exit: The granule pointed to by DE is locked out on the GAT table located at @GATBUFF.

A is destroyed.

**@NEXGRAN**

Advances DE to the first sector in the following granule.

Entry: DE = track/sector

Exit: DE = first sector in next gran.  
B is destroyed.

#### @BULKGO

A call here will completely bulk erase a disk.

Entry: (@DRIV) valid for desired drive.

Exit: Z = OK  
NZ = not completed.

**NOTE:** This routine can be fatal to a diskette!  
A stream of 0's are written to every track on  
the diskette to erase all data!

#### @BLKERAS

Diskette bulk erase entry point.

#### @UFMT

Special format utility entry point.

#### @RANDOM

Compute a pseudo-random number.

Entry: NONE

Exit: A = pseudo-random number from  
0-255 inclusive.

#### @SPATTERN

DATA pattern to be used in single density. This  
is a 16 byte string, and is duplicated 16 times  
when formatting a 256 byte sector in S/D.

#### @DPATTERN

DATA pattern to be used in double density.  
This 16 byte string is duplicated 16 times to fill  
256 bytes of data during formatting.

**NOTE:** SU+ comes initialized with E5H for  
single density, and 6DB6H for double density.  
These have proven to be worst case patterns, and

will result in a maximum number of formatting failures. The logic behind this is that a marginal disk should be revealed during the format process than after valued data is on the disk.

#### @CLRBUFF

Zeroes out a format buffer prior to building the track data.

Entry: (@FMTBUFF) = format buffer to be used.

Exit: 2000H bytes of 0's written to the indicated buffer.

DE and BC are destroyed, HL is preserved.

NOTE: It is good practice to clear out a format buffer prior to building the track data. If a short track is prepared and there is any extraneous data after the buffer, it too will be written to the disk and possibly picked up by the FDC during a sector read operation.

#### @COMPCOD

Computes the difference in password encoding between TRSDOS III and all others. This routine is called by @CODE. All DOS's use a common password encoding scheme for compatibility except TRSDOS III and 2.7DD. The coding method is identical except for 2 bytes. @CODE calls @COMPCODE to execute the proper 2 byte instruction.

#### @NAMEDAT

Displays disk name and date.

Entry: @GATBUFF + DOH-DFH hold the current name/date.

Exit: Name and Date are displayed on the video.

The name and date are checked prior to displaying. If any bytes are less than 20H (control codes), then the data is considered non-standard and a message "Invalid Name/Date Data" is displayed.

#### @WHERDIR

Computes the directory location on a disk. This routine is called by @RDDIR. If the disk is specified as DoubleDOS, then track 17 is returned (DoubleDOS MUST have directory on 17). Else track 0, sector 0 (sector 1 Mod III disks) is read. The 3'rd byte (2'nd byte TRSDOS III) is fetched as the location of the directory.

The resulting byte is checked and must be from 1-191 to indicate the real range of allowed directory tracks. If the byte is outside this range, then 17 is returned.

#### @RDDIR

Read a directory into memory.

Entry: (@DRIV) valid for desired drive.

Exit: Directory is read into memory.

C = read error, or invalid sector count.

Z = no sectors read into memory.

NZ and NC = successful.

(@DIRSCNT) = sector count of the directory.

DATA starts at @GATBUFF

First, the track indicated by the current table setting at @DIRTRK is read in. If it is a read protected sector (NON-read protected TRSDOS III), then the read continues. If not valid, a call is made to @WHEREDIR to determine the location of the directory, and the table is updated.

If the new track does not meet the read-protect status, an error return is made.

Successful sectors are read in until the end of the track is reached. If single density or ND80 DD is set, then sectors reads are continued until the first non read-protected sector is found. This is so that SU+ can read in extended directories created by ND80 in either single or double density.

#### @WRDIR

Writes the last directory read in back to the disk.

Entry:        (@DIRTRK) holds the current directory track.  
                (@DRIV) = current drive number.

                (@DIRCOUNT) = number of sectors to be written.

                @GATBUFF = starting address of data.

Exit:         Z = OK  
                NZ = Error

#### @SHOWDIR

Full screen display of a directory block.

Entry:        Directory data begins at  
                @GATBUFF  
                (@DIRPAGE) holds starting sector -2.

Exit:         All files in 8 continuous sectors are displayed.

There is only room on the screen to display 64 files at the same time. Since double density disks and those with extended directories can hold quite a few more, a block of 8 sectors is displayed at a time. Usually (@DIRPAGE) is advanced by 8 sectors at a time.

#### @SHOWIT

Displays a single filename to the video.

Entry: IX => first byte of directory entry  
 HL = video address where to display  
 Exit: Z = filespec OK  
 NZ = invalid filespec, not displayed

This is a useful call to determine if a directory entry contains a valid entry. Load HL with 0 if you want to determine the entry status, but not display the name. This call will display a killed file also.

@DSKDIR

## Display Disk Directory Entry Point.

**@DIRPART**

**D**isplays disk name, date, free grans, free files.

**Entry:**      **Directory**    must    already    be    in  
                 **memory**

Exit: Diskette data displayed to video.

@GETGCNT

Computes number of free granules from GAT table.

Entry:      HL => GAT table  
              B = diskette track count  
              A = D4H if OFF bits to be  
              counted  
              = DCH if ON bits to be  
              counted

Exit: DE = sum of counted bits  
(@FREEG) = (DE)

@CHDNAME

Change disk name entry point.

**@PUTNMDT**

Writes GAT table to a disk.

**@ZUNUSE**

Zero unused directory entries entry point.

**@DIRLIST**

Displays a directory. Indicates if a file is Invisible, System, and the Protection Level setting.

**@ALINE**

Line counter routine to prevent scrolling.

Whenever a long listing is being presented to the video screen, a call to @ALINE should be made with each linefeed. The program will pause when the screen is filled and wait for the ENTER key.

**@MODDIR**

Entry point to the full screen 'purge/restore' utility. A directory is read into memory, and the files are killed/restored individually.

**@MODKILL**

Subroutine to kill a file from @MODDIR.

Entry: IX => directory entry of file to be killed.

Exit: File is killed, and screen is re-draw via @SHOWDIR.

**@PUTBKDR**

Prompts if directory is to be written back.

Entry: NONE

Exit: Screen is cleared, and user prompted if directory is to be written back. If yes, a call to @WRDIR is made, else a RET is executed.

**@KILLIT**

Kills a file from a directory.

Entry: IX => first byte of directory entry

Exit: File is killed.

**NOTE:** This routine will only reset bit 4 of all active primary and extended entries. The grans are not released. A call to @MAKEGAT will free up the allocated grans.

**@KILLW**

This address holds the address of the beginning of the buffer area where the filenames reside. @KILLIT uses this address to compute the location of an extended directory entry.

**@RESCUE**

Same as @KILLIT, except the file is restored.

**@CLRFILE**

Zeroes out all non-active directory entries. Directory must already be in memory.

**@ZERFILE**

Zero out a single directory entry.

Entry: IX => beginning of directory entry.

Exit: All bytes contained in the specified entry are set to 0.

**@RSYST**

Remove system files entry point.

**@ENTRIES**

Computes the number of directory entries in the current directory in memory.

Entry: NONE

Exit: E = number of directory entries.

HL, BC are destroyed.

#### @RPASSW

Remove passwords entry point.

#### @CLEARF

Zeros out a single directory entry.

Entry: IX => first byte of directory entry

Exit: Primary and all extensions belonging to the specified file are zeroed.

#### @MAKEGAT

Construct a GAT table.

Entry: Directory located in memory starting at @GATBUFF.

Exit: Complete G A T table reconstructed.

This routine will first make a call to @FIGTRAKS to establish the current track count of the diskette. If you wish to reconstruct the directory with a different track count, make the call to @MAKEGAT+3. All files in the directory are allocated.

#### @ALLOCAT

Allocates grans belonging to current file.

Entry: IX => directory entry

Exit: All grans associated with this entry are allocated. Only the current entry is operated on, extensions (if any) are ignored.

CALL @ONEDRIV	;ask for a drive
CALL @RDDIR	;read the directory
CALL @DIRPART	;display name, date, ;free space

```

CALL @INITGAT      ;clear out GAT table
CALL @ENTRIES      ;compute number of
                     ;entries
LD    IX,@FILBUFF  ;start of filenames
LOOP EQU $          ;where to loop
BIT   4,(IX)        ;active entry?
CALL NZ,@ALLOCATE ;allocate this entry
CALL IXDIR         ;point to next entry
DEC   DE            ;reduce counter
LD    A,D
OR    E              ;any bits left?
JR    NZ,LOOP        ;continue if not done

```

**@TOPGRAN**

Computes highest bit position for usable grans.

Entry:            NONE

Exit:            A' = bit set for highest gran on  
track.

**@MAKEHIT**

Completely rebuilds HIT table starting at  
@GATBUFF+100H.

The entire table is zeroed out and  
re-constructed from scratch. On TRSDOS III, the  
last 32 bytes in the sector are not affected as  
this is the system file allocation table.

**@HASH**

Compute HASH code for a filename.

Entry:            IX => first byte in directory  
record.

Exit:            A = hash code byte.

All other registers preserved.

**@PUTHIT**

Insert HIT byte into table.

Entry:            IX => directory record.

H I T   t a b l e   s t a r t s   a t  
@GATBUFF+100H.

Exit: All corresponding HIT bytes for the associated files are inserted into the table. File extensions if any are taken care of also.

**@DSKFREE**

Free Grans routine entry point.

All 4 drives are scanned, and if a directory is found, the name, date, and free grans/files are displayed.

**@KILLCAT**

Kill Files by Category entry point.

**@FIGTKS**

Computes the track count of a diskette from the GAT table.

NOTE: If ND80 or TRS III is indicated, the track count cannot be interpreted from the GAT table and must be entered explicitly. If DoubleDOS, the track count byte is @GATBUFF+CDH. If LDOS, track count is at @GATBUFF+CCH minus 35. If none of the above, the LOCKOUT table is scanned backwards for the first non-locked out track. This is then assumed to be the track count of the diskette.

A call is usually made here after a directory has been read in to establish the actual track count in case it is different than the last one read.

**@COMPDIR**

Compute the correct command to write a directory sector.

Entry: (@DRIV) = drive to be written to.

Exit: A = command for FDC

**@COMPDAT**

Compute the correct command to write a data sector.

Entry/Exit: Same as @COMPDIR

There are several different ways that the floppy disk controller (FDC) can mark sector data on a disk. Normally, the data and the directory are written with different data address marks (DAM) to help identify that a directory is actually read in. Normally, the data is written as STANDARD, and the directory as READ PROTECTED. This doesn't mean that it can't be read, but is simply a bit on the disk that can be identified.

TRSDOS Mod III has the marks BACKWARDS. That is, DATA sectors are read protected, and the directory is standard. Making a call to @COMPDIR and @COMPDAT assures that the correct write command will be issued to the FDC.

**@CHFILE**

Change file parameters entry point.

**@BCDIR**

Loads BC with displacement between filenames.

Entry: (@DRIV) = current drive

Exit: BC = 30H for TRSDOS III

BC = 20H for all others.

**@IXDIR**

Advances IX pointer to next directory record.

**@HLDIR**

Advances HL pointer to next directory record.

**@IXDIRB**

Moves IX pointer BACKWARD to next directory record.

The directory is normally arranged with each record being 32 bytes long, and 8 per sector. On TRSDOS III, the directory records are 48 bytes long, and 5 per sector. The last 16 bytes in each sector is unused. Calls to the above will move the pointer correctly to the next file record.

CALL @ONEDRIV	;ask for a drive
CALL @RDDIR	;read the directory
JP NZ,4018H	; last sub-menu
CALL @INITCNT	;zero the counter
CALL @GETENTRY	;compute number of ;directory records
LD HL,@FILBUFF	;start of filenames
LOOP EQU \$	;loop label
BIT 4,(HL)	;active record?
CALL NZ,@ADDCNT	;bump counter if active
CALL @HLDIR	;point to next record
DEC DE	;reduce counter
LD A,D	
OR E	;any bits left?
JR NZ,LOOP	;do the rest
CALL @SHOCNT	;display the counter
RST 8	;display message
DB 'Active Records.',0	
JP @GOBACK	;press <ENTER> to ;continue

#### @RPDIR

Read Protect Directory entry point.

#### @URPDIR

Un-Read Protect Directory entry point.

#### @REPGAT

Repair GAT table entry point.

#### @REPHIT

Repair HIT table entry point.

**@REPBOOT**

Repair BOOT sector entry point.

**@REPCOMM**

Execute common subroutine for all activated drives.

Entry:        HL = subroutine address.

Exit:        branch to @GOBACK made when  
                all drives completed.

**@GATREP**

Code to repair GAT table. Vector from  
@REPGAT via @REPCOMM.

**@HITREP**

Code to repair HIT table for each drive  
selected via @REPCOMM.

**@BOOTREP**

Code to repair BOOT sector for selected drives  
via @REPCOMM.

**@ZUNGRANS**

Zero Unused Granules entry point.

**@ZEGRAN**

Zeroes current granule DE points to.

Entry:        DE = current track/sector

Exit:        Gran containing sector pointed to  
                by DE is zeroed.

    Z = OK

    NZ = Error, code in A.

**@GRNSIZE**

Compute number of sectors per granule.

Entry:        (@DRIV) = current drive

Exit:        B = # sectors per gran.

A is destroyed (= byte from @TYPEA)

**@NEXGRAN**

Advances DE to start of next granule.

Entry: DE = current track/sector.

Exit: DE = starting sector of following granule.

**@CHKDIR**

Check Directory Entry Point.

**@MOVEDIR**

Move Directory entry point.

**@MNYGRNS**

Returns a mask byte in E. Off bits indicate a gran is available in the current operating system.

Entry: NONE

Exit: E = mask byte, set bits cannot be used as a gran.

**@SCOPY**

Standard Disk Copy entry point.

**@ASKFMT**

Prompt if destination disks are to be formatted.

Entry: NONE

Exit: (@IFBFMT) = 1 = YES, format  
(@IFBFMT) = 0 = NO, don't format

**@REMDRIV**

Removes the current drive from the active table.

Entry: (@DRIVE) = current drive

Exit: Bit 3 of the corresponding @TYPEB table is reset.

**@UPDIRX**

Updates GAT table in memory to track count specified by (@TRACKS), and then updates to the current disk.

Entry:            Directory in memory starting at @GATBUFF

Exit:            GAT table re-created to the track count indicated by (@TRACKS), and is then written back to disk.

This is the normal exit for Format Without Erase, and Standard Backup to unlike track counts. If a 40 track diskette is copied to an 80 track formatted disk, the additional 40 tracks are opened up and made available to the users operating system.

**@UCOPY**

Special Disk Copy entry point.

**@SHOMNY**

Used by special disk copy. This call displays the current new data when scanning a disk prior to the copy.

**@GOCOPY**

This routine will format a non-standard disk.

The disk must already have been pre-scanned, and the table of data must have been created starting at @DAMBUFF.

The format of the table to copy special disks is as follows:

Each track has the following format for the number of tracks indicated by (@TRACKS).

First byte = # single density sectors

Second byte = # double density sectors

Then follows 6 bytes for each sector on the track.

First byte = Track Number.  
Second byte = Head Number.  
Third byte = Sector Number.  
Fourth byte = Sector length.  
Fifth byte = result of sector being read in.

**NOTE:** The fifth byte for each sector is used to determine the AM status of a sector. False sectors cannot be duplicated.

**@PERCOPY**

Transfers a full non-standard disk a sector at a time. Uses the table built via special copy to determine the sectors to be transferred. A call to @GOCOPY is made prior to @PERCOPY to format the disk first.

**@COMPASS**

Encode/Decode Password entry point.

**@FIGPASS**

Computes a password.

Entry: DE = password to be computed.  
Exit: 8 byte ascii string displayed showing a working password.

**@DRVSTAT**

Drive Status entry point.

**@DRVCHK**

Display's the status for a single disk drive.

Entry: (@DRIV) = current drive  
Exit: Message displayed if disk is mounted, empty, etc.

**@FILEMAP**

Disk Allocation entry point.

**@FILETOT**

Displays relative data found in a files directory entry.

Entry: IX => first byte of a directory record.

Exit: Filename, Track, Sector, Byte and DEC of the current record.

#### **@FILMORE**

Displays relative data found in a directory record.

Entry: IX => directory record.

Exit: End of file sector (EOFS), End of file byte (EOFB), Logical record length (LRL), encoded passwords, # grans, and all extents belonging to a directory record displayed.

#### **@SHOEXTS**

If a directory record has an extended entry, a call here will display the data located in it.

#### **@DISFILE**

Display File Sector entry point.

#### **@BADFILE**

Displays 'Invalid Filespec.' message.

#### **@NOFILE**

Displays 'File Not Found' message.

#### **@SHOWFL**

Displays a files sectors data on the video screen. Filename and relative sector are indicated.

#### **@DISFLTB**

Lookup table used in response to keys pressed in display file sectors.

#### **@CKASCI**

Checks to see if a character is a valid filename character.

Entry: A = character to be checked.  
Exit: C = invalid character  
NZ = OK

#### @MOVFILE

Takes an ascii filename string, and moves it into the file device control block (DCB).

Entry: HL => ascii filename.  
Exit: C = invalid filespec  
NC = OK.

If OK, filename is moved to @FILEDCB in the same configuration as it would be found in the directory, i.e., an eight byte filename padded with blanks followed by the 3 byte extension padded with blanks. If a password was specified, it is moved to @PASSWORD, although it is never checked for validity.

If a drive was specified, (@DRIVE) and (@DRIV) will be updated, and (@SCANFLAG) will be set to 1 to disable a drive search. If a drive is NOT specified, (@SCANFLG) is set to 0 to enable a drive search.

#### @FNDFILE

Locates a filespec in a directory.

Entry: Filename must have been pre-moved to @FILEDCB  
Exit: C = file not found  
NC = file found.

If found, directory will be in memory and IX will point to the directory record.

```
BEGIN EQU $  
CALL ASKFILE      ;ask for filename from  
                  ;keyboard  
CALL @MOVFILE    ;move to DCB  
CALL C,@BADFILE  ;invalid filespec
```

JR C,BEGIN	;ask again
CALL @FNDFILE	;locate the file
CALL C,@NOFILE	;file not found
JR C,BEGIN	;ask for another
LD HL,(@CURSOR)	;get current cursor position
CALL @SHOWIT	;display the name

**@RELSEC**

Displays EOF sector and EOA sector for a file.

Entry: IX => directory record

Exit: EOF and EOA displayed to video.

**@POSIT**

Position to a relative sector in a file.

Entry: (@TEMP9) = sector to position to  
IX => first byte in directory record.  
DE = address of first directory record (@GATBUFF+200H).

Exit: C = sector is beyond file limits  
NC = OK  
DE = track, sector of requested position in file.

**@POSITBC**

Position to relative file sector.

Entry/Exit: Same as @POSIT, except relative sector is supplied in BC register instead of (@TEMP9).

**@GRNSIZE**

Compute # of sectors in a granule.

Entry: (@DRIV) = current drive

Exit: A = # sectors per granule.

**@FSECTOR**

Sector Allocation entry point.

**@WRLNO**

Converts a 2 byte hex number to decimal ascii.  
Entry:            HL = 2 byte word in HEX  
                  IY => 5 bytes where string is to  
                  be written.  
Exit:            5 byte ascii string written to (IY)  
                  DE destroyed

```
LD   HL,(@TEMP9) ;fetch a number
LD   IY,(@CURSOR) ;write to current cursor
                    ;location
CALL @WRLNO       ;ascii string displayed
                    ;on video
```

**@OFSFILE**

Offset Disk File entry point.

**@SHOWOFF**

Displays module load range and entry point.  
Called by @OFFSETFL to display the current  
setting.

**@ASKFILE**

Prompts for a filespec to be entered from  
keyboard.

**@CLEARF**

Clear File Sector entry point.

**@COPFILE**

Copy Files entry point.

**@IFCOPY**

Displays a filename and asks if it is to be  
copied.

Entry:            IX => file directory record.  
Exit:            If YES selected, nothing is done.

If NO selected, the file is killed from the directory in memory.

#### @PERCOPY

Performs the copy of files after they have all been selected.

#### @ALLO00

Allocates all granules of a file.

Entry: IY => first byte in filename.

Directory begins at @GATBUFF.

Exit: All grans assigned to the file are allocated.

Extensions are allocated also.

IY unchanged, BC, HL destroyed.

#### @NEXEXT

Called by @ALLO00 to create a file extension if more directory records are needed.

Entry: IY => first byte of existing directory record.

Directory must start at @GATBUFF

Exit: Extension is created.

IY => first byte of extension.

All links forward and backward are maintained.

#### @COMFILE

Compare Files entry point.

#### @CHUNK

Allocate granules to a file.

Entry: Directory begins at @GATBUFF

HL = total number of grans needed.

Exit: C = number of continuous grans  
(1 directory extent)

DE destroyed.

Carry = directory full (all grans  
allocated)

@FNDSPOT

Locates an empty directory record.

Entry:      Directory at @GATBUFF  
              (@FNDSPOT+1,+2) = # available  
              records in directory  
              (result of a call to @ENTRIES)  
Exit:        HL => empty directory record.  
              Z = OK  
              NZ = no more records available

@EXECOPY

Called by @COPFILE. Performs the allocation of  
a file.

@CREATFIL

Build File entry point.

@DALLOC

Disk Allocation entry point.

@CMPGRNS

Computes the number of grans in a file.

Entry:        IX => directory record  
Exit:         C = # of grans assigned to a file

@DSPGRNS

Computes and displays the number of grans in a  
file.

Entry:        IX => directory record  
Exit:         "xxx Grans." displayed.

NOTE: A call is made to @CMPGRNS from this  
routine.

@CMPIHASH

Compute Hash Code entry point.

**@COMPADD**

Adjust the number of grans found in a directory record.

Entry: A = number of grans read from a directory record.

Exit: A = actual number of grans in the record.

```
LD  A,(IX+23)      ;fetch # of grans/start
                ;gran
AND  1FH          ;mask the number of
                ;grans
CALL @COMPADD     ;adjust to real number
```

On all DOS's except TRSIII, the right 5 bits = the number of continuous grans in a record -1. In TRSIII, the number is actual, and not -1. If the TRSIII bit in @TYPEA is set, then the number is NOT adjusted, otherwise it is incremented by one.

**@CTAPE**

Copy Tape entry point.

**@BYTEIN**

Reads one byte from tape.

Entry: Tape is running

Exit: A = byte input from tape.

**@BITIN**

Called by @TAPEIN 8 times to read one byte.

**@BYTEOUT**

Writes one byte to tape.

Entry: Tape deck on.

A = byte to write.

Exit: C = <clear> key pressed.

**@BITOUT**

Called by @TAPEOUT 8 times to write one byte.

**@ASKDECK**

Asks which deck is to be used (1 or 2), then prompts to press <enter> when tape is prepared.

**NOTE:** The Mod III has only one deck, and therefore the deck prompt is skipped, and <enter> is prompted for.

**@TPROMPT**

Prompts to press <enter> to begin. When enter is pressed, message press <clear> to terminate is displayed.

**@TAPEINT**

Sets up parameters used by tape input/output.

Entry:            NONE

Exit:            HL = 3840H (for <break> <clear> key checks)

                  DE = @DAMBUFF (where data is read from/ written to

                  IY = 3C00H+896 (for data display on video)

                  HL' = 0

                  DE' = 0

                  A = 0

**@FNDSYNC**

Displays "looking for sync", and jumps to @TAPELDR.

**@TAPELDR**

Turns on tape deck, and reads leader and sync byte.

**@TAPSHOW**

Displays data read from/ written to tape.

Entry:            HL' = checksum  
                  DE' = byte counter  
                  A = input/output byte  
                  IY => current video display location.  
Exit:            Byte counter is displayed.  
                  Video is updated to reflect the new data.  
                  Checksum and Byte counter are updated with new data.

**@RTAPE**

Read Tape entry point.

**@VTAPE**

Verify Tape entry point.

**@WTAPE**

Write Tape entry point.

**@PUTSYNC**

Turns on tape deck, and writes leader and sync byte.

**@BOOT1S**

Boot written for Mod I to boot TRSDOS

**@BOOT1D**

Boot written for Mod I DD to boot TRSDOS

**@BOOT3**

Boot written for Mod III to boot TRSDOS

The above 3 boots are used in the 'repair boot sector' in the disk repair utilities.

**@PGMEND**

Last byte in the program plus one.

This represents the first available byte for buffer use.

#### @PRBUFF

128 byte buffer used for printer spooler I/O.

When a call is made to @POUT to send a byte to the printer it is actually saved in this buffer. The interrupt service (@TASK) will take a byte out of this FIFO buffer and send it to the printer. This limits the speed of the printer to 40 cps (25 ms interrupts), and slows down a fast printer, but it does free up the CPU to perform other tasks instead of dedicated printing.

#### @DAMBUFF

Data Address Mark buffer.

When several sectors are read into memory, such as via a call to @MREAD, the data address marks for each sector are saved contiguously in this buffer so that when they are written back out to disk they will be marked correctly.

#### @GATBUFF

Normal buffer area used to read in a directory.

The number of sectors in the directory can be fetched from @DIRCOUNT.

#### @BUFFER

A buffer starting 256 bytes past @GATBUFF.

This buffer points at a HIT table in memory.

#### @FBUFF

Normally used as the starting address of a format buffer.

When a disk is formatted, every byte is prepared ahead of time starting at this address, and is written to a disk with a write track

command. This address is high enough so that a full track of sectors can be held below it.

This is how the format without erase works. It first reads all sectors on a track starting at @GATBUFF, and the data address marks are stored at @DAMBUFF. The track is prepared starting at @FBUFF, and written to disk. Then the tracks data starting at @GATBUFF is written back to the track a sector at a time.

#### @ENTRY

The normal entry point to the initialization code of SU+.

This address is PAST the program end, and all the memory used by the initialization code is reclaimed as buffer area after the code is run one time. This will set up the interrupt service, print the program labels, and read the configuration data from the disk. Once the program is running, 2 alternate entry points are available: 4015H branches back to the Master Menu. 4018H branches back to the last sub-menu viewed.

The ONLY memory NEVER used by SU+ is from @PGMEND to (@PRBUFF)-1. This represents the uneven block of memory from the last byte of the program to the next even page of memory. If the user desires, @TOPMEM may be reduced by even pages of memory if more is needed.

## \*\*\*\*\* SAMPLE ROUTINES \*\*\*\*\*

;compute the number of free directory records on a disk

START	RST	8	;display message
	DB	7	;clear screen
Records	DB	***	Computing Number of Directory
	DB	***'	
	DB	10	;linefeed
	DB	0	;string terminator
	CALL	@ONEDRIV	;ask for a single drive
	CALL	@RDDIR	;read the directory
	JP	NZ,@NOTDIR	;not a directory
records	CALL	@ENTRIES	;load E with # of directory
	CALL	@INTCNT	;clear the counter
	LD	HL,@FILBUFF	;start of files
LOOP	BIT	4,(HL)	;active record?
available	CALL	Z,@ADDCNT	;bump counter if
	CALL	@JILDIR	;bump HL to next file area
	DEC	E	;decrement file counter
	JR	NZ,LOOP	;do 'em all
	CALL	@SHOCNT	;display the counter
	CALL	@DRVASC	;get ascii drive number
	LD	(DRVE),A	;insert into message
	RST	8	;and print this message
	DB	'Free Files on Drive '	
DRVE	DB	'x.',10,0	;finish message
	JP	@GOBACK	;press <ENTER> to continue

; read a directory into memory, then display it by sector

START CALL @ONEDRIV ;ask for drive  
CALL @RDDIR ;read the directory  
JP NZ,@NOTDIR ;not a directory  
LD HL,@GATBUFF ;where data begins  
JP @DOADDR ;display it in memory

; verify all sectors on a diskette

START CALL @ONEDRIV ;ask for a drive  
CALL @INTCNT ;zero the counter  
LD D,0 ;starting track on disk  
CALL @FIRSTS ;E = first sector  
BEGIN LD BC,@BUFFER ;where to read data  
RST 28H ;display track/sector and  
read it  
CALL @NEXSEC ;bump to next sector  
JR NC,BEGIN ;C = end of disk  
CALL @SHOCNT ;display error counter  
RST 8 ;message  
DB 'Bad Sectors on the disk.',10,0  
JP @GOBACK ;terminate here

```
; zero a single sector on a disk
START CALL @GETDAT ;ask for drive, track, sector
        LD BC,@BUFFER ;point to buffer area
        CALL @ZBUFF ;zero the buffer
AGAIN  RST 30H ;write to disk
        JP Z,@GOBACK ;successful
        JR AGAIN ;write again if 'retry'
selected
```

```
; check to see if a file exists on a disk
START CALL @ASKFILE ;fetch filename
        CALL NZ,@NOFILE ;file not found
        JR NZ,START ;ask again
        CALL @DRVASC ;get drive where found
        LD (MSG),A ;put into message
        RST 8 ;display message
        DB 'File Found on Drive '
MSG   DB 'x.',10,0
        JR START ;ask for another
```

```

; log TRSDOS III system files into a directory
START CALL @ONEDRIV ;ask for drive number
        CALL @RDDIR ;read the directory
        JP NZ,@NOTDIR ;non disk error, invalid
format
        LD A,(IY+7) ;fetch dos type
        CP 3 ;TRS III DD?
        JR Z,OKGO ;ok if yes
        RST 8 ;else print message
        DB 'Not a TRSDOS III diskette.',10,0
        JR START ;ask again
OKGO grans
        CALL @DIRPART ;display name/date, free
start
        LD IX,@HITBUFF+0E0H ;where sys files
        LD C,0 ;current system number
        LD B,16 ;16 system files possible
LOOPA LD A,(IX) ;get a byte
        INC A ;FF = inactive
        JR Z,DEAD ;this one not active
        CALL @FNDSBOT ;locate an available space
        PUSH BC ;save counters
        LD C,(IX+0) ;fetch # grans/offset
        LD B,(IX+1) ;fetch start track
        PUSH IX ;save IX for next loop
        PUSH HL ;pass to IX
        POP IX ;IX = start of record
        LD (IX),10H ;set as active, non-system,
visible
        CALL PUTNAME ;insert name into the entry
        LD (IX+22),B ;insert into file entry
        LD (IX+23),C ;put into file entry
        LD A,C ;fetch it
        AND 1FH ;get number of grans
        LD E,A ;save it
        ADD A,A ;A times 2

```

gran)	ADD A,E	;A times 3 (3 sectors per
	LD (IX+20),A	;put end of file sector
	LD (IX+16),0EFH	;insert null passwords
	LD (IX+17),5CH	;EF5CH = null password
TRSIII	LD (IX+18),0EFH	
	LD (IX+19),5CH	
	LD (IX+24),0FFH	; i n s e r t e x t e n t
terminator	POP IX	;restore pointer
	POP BC	;restore counters
DEAD	INC IX	;point to next one
	INC IX	;2 bytes per entry
counter	INC C	;bump system number
	DJNZ LOOPA	;finish them all
	CALL @MAKEHIT	;insert names in HIT table
disk	JP @WRDIR	;write directory back to the
PUTNAME	LD (IX+5),'S'	;insert filename
	LD (IX+6),'Y'	
	LD (IX+7),'S'	
	LD (IX+8),'T'	
	LD (IX+9),'E'	
	LD (IX+10),'M'	
	LD A,C	;get system number
	PUSH BC	;must save counter
	RST 18H	;convert to ascii
	LD (IX+11),C	;put into filename
	LD (IX+12),B	
	POP BC	;can restore it now
	LD (IX+13),'S'	
	LD (IX+14),'Y'	
	LD (IX+15),'S'	
place	RET	;name now in the right

**TECH-106**

**Super Utility Plus 3.0**

**This page intentionally left blank**

**Copyright (c) 1983 by Breeze/QSD, Inc.**

## SU Plus 3.1a - Symbol Table - Model I

\$B	425A \$D	421B \$D10	4222 \$D3	4229 \$D1F3	78CA \$D1FD	78CA
\$DIFS	79BC \$L	420E \$L10	420D \$L3	4214 \$M	4230 \$H10	4237
\$M3	423E \$N	4245 \$N10	424C \$N3	4253 \$SECH3	0012 \$SECH0	0012
\$SECHS	0009 \$SEC13	0001 \$SECLO	0001 \$SECLS	0000 \$T	41F1 \$110	41F8
\$T3	41FF \$U	41EA \$X	4261 \$X10	4268 \$X3	4276 \$X35	426F
\$00	0000 308	0008 \$J10	0010 \$J18	0018 \$J10ER	C993 \$10DSP	C9A3
\$10ERR	C994 \$10GETB	C9AC \$10GETL	C97A \$10HAVB	C9C1 \$10LOAD	C968 \$10LOOP	C94F
\$10LOOP2	C960 \$10LOOP3	C96E \$10MER	C989 \$10RD	C9E3 \$10RD1	C9F4 \$10RD2	CA12
\$10RD3	C91B \$10RD4	CA1B \$10ROS	CA2F \$10RDL	C9E5 \$10REO	C909 \$10REM	C9C4
\$10RES	C9C8 \$10RRE	C905 \$10SEK	CA05 \$10SER	C98E \$10SLO	C90C \$10STR	C967
\$10SER	C9A1 \$10SOP	C981 \$10SER	C9A1 \$10SE18	C98A \$10GETL	C980 \$10HAVB	C9CF
\$10LOOP	C979 \$10LOOP	C950 \$10SOP2	C96E \$10SOP3	C97C \$10MER	C997 \$10RD	C9F1
\$10R01	C902 \$10SOP2	C922 \$10SOP3	C925 \$10SOP4	C92F \$10ROL	C9F3 \$10SER	C9E0
\$10SER	C902 \$10SOP5	C909 \$10SER	C9E3 \$10SEK	C913 \$10SER	C9C9 \$10SSLO	C9E4
\$10SSTR	C975 \$20	0020 \$28	0028 \$30	0030 \$38	0038 \$30DER	CAB2
\$30ERR	C985 \$30GETB	C974 \$30GETL	C969 \$30HAVB	C9A7 \$30LOAD	C5A4 \$30LOOP	CAB3
\$30LOOP2	CA4F \$30LOOP3	CAS0 \$30MER	CAT8 \$30NMIR	CB18 \$30RD	CAC9 \$30RD0	CA08
\$30RD1	CACB \$30RD2	CB09 \$30RD3	CB15 \$30RED	CB86 \$30REM	CAA4 \$30RES	CA81
\$30RM1	CB2F \$30RD4	CARB \$30SER	CA70 \$30SK	CAE0 \$30SLO	CAC2 \$30STR	CAS6
\$31	0000 302	0000 303	0000 304	0000 AASHOW	6EEB ABTCY	C517
ACC1	9E44 ACC2	9E6E ACONF	8F60 ACONF	BFF0 ACTIVI	7C1C ACTIV2	7C24
ACTIVE1	7C17 ACTIVE2	7C1F ADATE	8FBC ADATE1	BFAF ADDACNT	6051 ADDAPP	8C5C
ADDBIN	5C6E ADDBLP	526C ADDCLP	6060 ADDCNT1	6057 ADDDEC	5C68 ADDHEX	5C70
ADOLP	7839 ADDOCT	5C71 ADDR	59A5 ADDR20	5ACF ADDRZ0A	5A05 ADDRD	5B5C
ADDRESS	4020 ADDOPT	50F1 ADDVEC	5C52 AOJ1	760E AOJ2	7613 ADJBYTE	6E50
ADJERR	7610 ADJF	6E7A ADJGRN3	8040 ADJGRNS	84CA ADJSTR	75C0 ADJUST1	75E0
A0JSTR2	75EA ADJUST0	7645 AE0FS	85C5 AFGMG	C2BF AGGMG	C2A0 ALINT	A108
AIUWTX	A1FA AIUWTY	A205 AKBAK	9BCC AKSCLR	4F6C AKSHR	91DF ALINE	81AB
AL00	C3E4 ALLOCAT	0439 ALLOCFI	B61F ALLOCPL	8442 ALLOON	C42E	
ALLOGO	C650 ALLO1X	C440 ALLOLP	C3F9 ALLOTAP	846C ALVFLAG	442F ALVFTBL	452A
ALVTBL	4506 ANAME	8F66 ANAME1	8F89 ANOBAK	9F00 A0103	9411 APASS	8F82
APASS1	8F01 APASSWD	807C APPEND0	8027 APPEND0	806C APPEND1	806F APPEND2	B097
ASC1	5087 ASC2	5B8E ASC3	5B03 ASC4	5B2A ASCII	5B05 ASCI1	678E
ASENUM	6E40 ASK1	C002 ASK1A	C040 ASK1B	C0C8 ASK1C	C0E8 ASK1D	C0E8
ASKIE	CE2E ASKIF	CE40 ASKIG	CE5A ASKIH	CE71 ASKII	CE05 ASK2	CEAF
ASKAUTO	8098 ASKFO	8028 ASKFIG	802A ASKFIL	80C2 ASKFMT	9847 ASKGRP	B986
ASKIBM	7026 ASKIX	7029 ASKIXN	7057 ASKIXY	7052 ASKRE1	7591 ASKREP2	75B0
ASKREPL	75B1 ASKSAME	6114 ASKSTR	755F ASKSTRD	7640 ASKSIRE	7650 ASKSTRU	756A
ASKSYNC	AC49 ASVE	5071 AVALO	9779 BAEDFS	B5C1 BACKSP	0008 BADCAT	841E
BADCLSD	SE48 BADCNT	6065 BADCONX	6523 BAODAM	7168 BAODAT	5AE0 BADEF	B06A
BAFILE	B497 BADFOR	80C6 BAOFXR	6830 BADISRT	6600 BADMEM	A47F BADMEM2	A48A
BADRO	SE4E BADSEL	5969 BADWRT	5E66 BAKFM10	98F3 BAKFM11	98FA BAKFM15	98EE
BAKLP1	6830 BAKLP2	6842 BAKLP3	6844 BAKLPG	6826 BAKMOD	6828 BAKSEC	682A
BAKSPCE	5083 BAKVG0	98EB BASE	3000 BC0DR	899E BDRCLS	5E2F BDTK	A715
BOTCL5	SE34 BINASC	61F9 BINASC1	6201 BINASC2	6209 BINASC3	6211 BINIT	6786
BIT0	50C8 BIT0DS	50E8 BIT1	50AE BIT2	509E BIT3	5074 BIT4	5050
BITS	5042 BIT6	5025 BIT7	5014 BITCAN	8037 BITIN	8016 BITINH	8025
BITINL	801A BITINL	B006 BITOUT	AFF6 BITOUTD	8001 BKSPA	4F21 BLANK	0020
BLD001	C59F BLD01	C592 BLD0PUT	A056 BLDRET	C57A BLKERAS	980A BLKS	6AC8
BLOCK	00FB BM11	AA9F BM12	AAB5 BM13	AA01 BM14	AA0E BMN	C619
BOL	0010 BOLX	4F31 BOOT1D	C936 BOOT1DL	0100 BOOT1S	C844 BOOT1SL	00F2
BOOT3	CA36 BOOT13L	00FB BOOTENT	9607 BOOTREP	8A06 BOTMEM	4039 BOTOFIL	B629
BREAK	0001 BRKCK1	42F5 BRKCK2	42FB BSDIR	67A6 BT4	A763 BTKD	A72E
BUFFADD	S28A BUFFEND	618A BUFFER	0500 BUFFLEN	2800 BUFFPRI	0184 BUFPAS	4078
BUFTAKE	S2CF BUILD	9370 BUILD1	937F BUILD2	93A2 BUILD11	93B2 BUILDNO	938F
BULKGO	989C BULKLP	9880 BULKSS	98C1 BVM	C5F1 BURTCP1	SE96 BYTECNT	A82C
BYTEIL	8000 BYTEN	8008 BYTEOL	AF08 BYTEOUT	AF09 CANNOT	7066 CASDAT	00FF
CASMOT	00FF CASSEL	37E4 CCAATT	8632 COATMSK	8782 COIRNSK	875F CENTR	5000
CF0	CB00 CFF2	CFBF CGRANS	4028 CHAROK	5070 CHASH	C789 CHASHH	C78C
CHAVE	CB1B CHAVE1	6B1F CHAXE	6B1B CHONAMC	8028 CHONAME	801C CHFILE	8784

## SU Plus 3.1a - Symbol Table - Model I

CHKDIR	8AF0 CHKITX	8C00 CKHOUT	8BC0 CKHTASK	441F CHUNK	BF48 CHUNK0	BF98
CHUNK1	BFAB CHUNK2	BFB0 CHUNK3	BFCC CHUNK4	BFDE CHUNK5	BFEA CHUNK6	BFF6
CHUNKA	BFS7 CHUNKAA	BF66 CHUNKB	BF6A CHUNKC	BF7C CHUNKX	BF86 CHUT	6ACE
CHWT1	6ACF CHWT2	6AD2 CHWT3	6AE1 CHWT4	6AE7 CHWTS	6AFA CKADDR	79A0
CKADDRL	7984 CKADDRS	798E CKASCI	B693 CKBL0	9393 CKC0	C678 CKC00	C68C
CKC1	C699 CKC11	C6A3 CKC2	C6C6 CKC3	C6AB CKC4	C6CC CKC44	C606
CKC444	C6D9 CKC5	C4B8 CKCOMFO	BEF7 CKCONF	6397 CKCTL	S160 CKDEND	64E8
CKOENDX	64E4 CKOIRD	8B00 CKOIRTK	9081 CKDLY2	5103 CKOSAV	S1BB CKOSID	64C5
CKOSIE	64C4 CKOSIZE	C4E4 CKOSIZ0	C515 CKDUL	S1B2 CKENUM	7005 CKEXT	8800
CKSEP	6380 CKSET	64F7 CKSIDS	64D1 CKSL1	653A CKSL3	653F CKSLD	652E
CKSLP	64F8 CKSLS	6529 CKSRET	6527 CKTBL5	5981 CKTBLX	5980 CKTBLY	59E
CKTKS	6349 CLEAR	0003 CLEARFL	COEC CLEARX	D658 CLEARY	0018 CLRBUFF	9913
CLRFILE	8109 CLRFLP	C127 CLRTHIS	7E11 CLSD	4FCA CLS02	4FF9 CLS1	4F96
CLS2	4FAA CLSA	0004 CLSB	0007 CLSF	0005 CLSMOVE	5372 CLSOL	4FB1
CLSON	5363 CLSONOK	536F CLSTIT	52F1 CMFNEXT	BF20 CMH1	A7AE CMH2	A7B8
CMOWXE	BF04 CMPAD0	C7EA CMPG8K	C71F CMPGRNS	C6F4 CMPQE	C2F6 CNFCT	CB4C
CNTL08	BA23 CNTL0D	6098 CNTMSG	6038 CNTOTAL	6086 CNTRET	608A CNUFF	C005
CODDD	963A CODDDU	9610 CODDOZ	9627 CODE	9617 CODEZZ	9641 CODLP1	9635
COOLP2	9646 COLS	0040 COMBI	6E95 COMB2	6E96 COMDEC	9E1A COMENC	9E1A
COMPFILE	B09E COMPILX	BDD3 COMFIN	7122 COMFLP	BE18 COMFLP1	BE72 COMFOK	BE0B
COMPILP	8E33 COMMA	002C COMMEN	A96F COMMEMB	A978 COMMEL	A97B COMMO	714C
COMMSTR	715E COMOFLP	BE55 COMPAAD	C70B COMPARE	6104 COMPC03	9934 COMPCOD	9924
COMPCOE	9938 COMPCOF	9941 COMPDAT	676F COMPOIR	8748 COMPEXT	C723 COMPGD	C6F7
COMPGL	C702 COMPPI	9DC2 COMPSP	7A7B COMPW	9E11 COMSEC	7094 COMSECL	7000
COMSL	7D5F CONF0	CB60 CONF1	901E CONF1C	9027 CONF2	905C CONF0	902A
CONF1G	CC45 CONF5	920B CORNET	63A3 CONTBL	6346 CONTRL	4E53 CONTRPT	50E1
COPFILE	C163 COPFLP	C192 COPSEC	7188 COPSECL	712C COPWHR	4447 COPY	4F2
COPY1	C23A COPY1L	C2FF COUNT	40A7 CPASSW	908F CPLP1	9C0B CPLP2	9D3B
CPLPX	904C CPLPY	9089 CPLPZ	9058 CPPDR	CF28 CPYVER	9000 CPYVER1	9D12
CR	000D CRAFT	C540 CREAT0	C52C CREATF	C53D CS01RZ	67A0 CT1	A9F9
CT2	AA03 CTAPE	A0B1 CTAPED	A0B2 CTAPEL	A0A2 CUBP	922B CURCHR	0098
CURCHR0	004 CURCUR	502D CURKE	8B07 CUROFF	5033 CURON	5031 CURSOR	4021
CXBOT	8019 CXKB0	8024 CXTOP	8008 CXTOPN	8016 DALLOC	C644 DAMARKS	725C
DMABUFF	0400 DAMK	7266 DAMNOTR	7986 DAMOK	717C DAMSEL	72CA DARR	005C
DBUFF	F800 DATERR	5080 DATHSK	5844 DBLIX	8C2A DBUFF	0200 DBUILD	A2F0
DCT0	4172 OCT1	417F OCT2	418C OCT3	4199 OCT4	41A6 OCT5	4183
DCT6	41C0 OCT7	41C0 OCTFF	559A OCTPAS	8C12 OCTTBL	410A ODAA1	8170
DDAA2	8179 DDAA3	818C DDIRR0	892F DDIRRS	8930 DDIVD	58FF DDIVD1	5904
DDIVD2	5908 DDIVD3	5900 DDIFIX	55E5 DDIRVE	4072 DDTCN	7C0E DDTXC	7C07
DE1	7FC9 DE2	7FD6 DE3	7FEE DE3LP	7FB0 DE4	7FE3 DE4A	7FE9
DEALP	7FB5 DECBA0	6B22 DECBL0	A712 DECBTXT	6CBF DECCON	6C49 DECCON1	6C56
DECCON	6B39 DECUPC	6C2E DECUP1	6C3B DECIT	67B2 DECK1	ADCF DECK2	A007
DECNUM	701A DECOOE	6B35 DECOFF	6F98 DEORG	6B41 DECTBL	6223 DECTXT1	6C93
DEFADDR	402F DEFDIR	5877 DEFGET	90E0 DEFLTS	587C DEFUD	8F21 DEFUTP	58B1
DELAY	5403 DELAYX	54E3 DELBYTE	6F08 DESV	9307 DETEC0	7CCE DETEC00	7CE6
DETEC1	700E DETEC11	7025 DETEC05	7046 DETEC5	7054 DETECT	7CAT DETPUT	7057
DFDF0R	8974 DFDF0R3	8985 DFILCNX	8465 DFILCON	8408 DFILELP	B510 DFILEWT	8519
DFILWTX	B513 DFILFLWT	B532 D1GOK	5C4A DIRDN	7E44 DIRENT	96F7 DIRLIP	78E4
DIRLST	811C DIRLSTC	812F DIRLST0	81A1 DIRLST1	8150 DIRLST0	814E DIRMAX	7888
DIRNOTA	8934 DIRNOTB	894F DIRPAGE	4034 DIRPART	7F36 DIRSCNT	4033 DIRTKK	9700
DIRUP	7EF4 DIRX	C2F9 DIRY	C031 DISCON	6644 DISCON1	66A7 DISCTL	4EEE
DISON	4EE6 DISDSK	6544 DISOSKO	6585 DISDSK0	6588 DISOSK1	658A DISDSKX	6580
DISOSKY	6570 DISFIL1	848A DISFILE	8468 DISFLTB	8508 DISHV	66B5 DISHVXX	66BF
DISKLP	65CD DISLP	4ECC DISMCN1	A618 DISMCN	A618 DISMEM	A5F3 DISMRET	A660
DISMTBL	A665 DISOK	4E00 DISP1	40A3 DISP2	40A4 DISP3	40A5 DISPD	7AC5
DISPLY	4EC2 DISPN	7A8A DISPNXN	7A00 DISRET	67F1 DISRETM	A615 DISTBL	66E0
DISWT	66AA DIVD	58F2 DIVD1	58FA DKOK	66A0 DKSR1	6620 DLCKMM	51C8
DLOFF	5100 DLON	51BF DLY1	50F9 DLY2	5104 DLYCNT	6C7C DMEMB	A6A5
DMEMD	A699 DMEMDP	A6A2 DMEMT	A697 DMEMU	A696 DMEMUP	A69C DMOUNT	6143
DMSV	72E5 DMULT	58D8 DMULT1	58E1 DMULT2	58EA DNAMES	427D 0015	8108

SJ Plus 3.1a - Symbol Table - Model I

D02S	B1E4 D03S	B1F6 DOADDR	A5FD DOADDR	A6D2 DOALLT	B9F1 DOAND	6B02
DORLD	A6F8 DOBUILD	A6E7 DORDERD	9462 DORDER1	9470 DORDER2	9498 DORDER8	94CE
DORETRY	SEFO DOROT	6BE2 DOROT1	6BF3 DOSEEK	5564 DOSKIP	SEFO DOSPEED	6C7A
DOWNFL	8162 DOWNSEC	5AB8 DOXR	8079 DOYR	8088 DOYYX	8091 DPASSP	8B66
DPASSV	80EF DPATRN	9703 DRIV	4074 DRIVE	4075 DRIVES	4046 DRVASC	55A7
DRVCHK	B105 DRVCHKB	B155 DRVCKD	B0FE DRVCKOK	B13C DRVCOMM	62B3 DRVLP	62BC
DRVSAME	6150 DRVSTAT	80E9 DSEUP	61ED DSKDIR	708A DSKDIRS	7096 DSPFREC	85EE
DSKFREE	850F DSKSLO	54CB DSKSLO	54CE DSPGRNS	C762 DSPSEC8	C752 DSPSECS	C732
OSTAT	5678 DUMP1	9E02 DUMP2	9E44 DUMP3	9EEB DUMP4	9EE2 DUMP44	9EE1
DUMPS	9EFA DUMP6	9EFC DUMPASS	9E05 DUMPU	9E90 DVK1	B1AA DVK2	B1A3
DXTIX	7F52 DYTTY	7F46 EO	5750 ES	5710 E9	5742 EFANO	6EA3
EFOR	6EAT EFROT	6E87 EFSHIFT	6E7C EFXOP	6EAB ENDFILE	B61A ENDMEM	FFFF
ENDOFF	6F90 ENTER	0000 ENTRIES	8323 ENTRIS	8337 ENTRY	0600 EOAS	4026
EOFB	4023 EOFS	4024 EOJ	001E EOLX	4F38 EPCT	440B ERAFILE	8388
ERAFILE	B38E ERAFILE	8390 ERAFILE	839F ERAFILE	8344 ERMSG0	5747 ERMSG5	5716
ERMSG9	572E ETBL	0000 ETX	0000 EXCMEM	A841 EXCMEL	A844 EXCSAV	730F
EXSEC	730F EXSEC1	7351 EXECOPY	C042 EXECPYO	C040 EXECPY1	C05F EXECPY2	C076
EXECPY3	C048 EXIT	6230 EXTHIT	8508 EXTHITH	85CD EXTHITL	8556 EXTLPL	7CSF
FBBG	6E8C FBUFF	E700 FOCCMD	37EC FOCAOT	37EF FOCAOT	4460 FODESC	37EE
FOCAEL	37E1 FOCTIA	37EC FOCTRK	37ED FOCAWT	6247 FOFILE1	B7CC FOR	5FFA
FEICH8	B839 FEICHG0	B867 FFILCON	B849 FFLFLF	B287 FGSSV	B256 FGST	6C08
FIGDRV	631E FIGL005	8712 FIGLPL1	C0F6 FIGM005	B719 FIGN005	8726 FIGO1	E0D9
FIGPASS	9E77 FIGPOT	9E78 FIGPSLP	9EAB FIGPSST	88F6 FIGPST	8903 FIGPUT	872E
FIGTK1	873C FIGTK2	8746 FIGTKS	8AE7 FIGTRO	B711 FIGTRET	B734 FIGTRS	B735
FILBUFF	0800 FILBYTS	6F26 FILECT	B4EC FILEDCB	415F FILELP	B1A1 FILEMAP	B178
FILES	4069 FILETO1	B1B2 FILE	9442 FILEDEF	A92D FILLEM	A007 FILLREY	B225
FILM3	A950 FILMA	A968 FILMAPR	B1B8 FILMEMI	A0EB FILMGO	B2C0 FILMORO	B2AC
FILMOR1	B2B2 FILMOR2	B2A5 FILMORE	B2B8 FILMORX	B3F5 FILMORY	B338 FILREL	B22E
FILRELO	B210 FILREL1	B21A FILWR	4EAS FINBYTE	6EAF FINDEOF	C147 FINDS0	C036
FINGRLP	845E FIRSTS	5A55 FIX3LLP	83FF FIX4LLP	841F FIX5LLP	8431 FIXCON	4F3F
FIXREAD	5584 FIXSET	5500 FIXSETO	55E1 FIXURIT	55C0 FLAGA	40AC FLAGC	4060
FLCOUNT	503A FLGB0	6F36 FLICK3	5CAB FLICK4	5C43 FLICK5	5C53 FLICK6	5C68
FLICK7	5C7 FLICKS	5C75 FLICKSP	5C96 FLIPDN	5404 FLIPGO	5414 FLIPK	5C84
FLK2	5C7A FLKD0N	5C94 FLKPUT	5C91 FLKRES	5C79 FLKSDON	5C02 FLKSRET	5C42
FLKSWT	5C7A FLKWT	5C7E FLOORP1	B3D0 FLOORP2	8326 FLPOEN	541B FLSCUR	6FF9
FLSNRM	6FDD FLSSPD	6FFC FM1	8342 FM2	8350 FM3	8358 FM4	B36C
FMS	B37A FM6	8389 FMIBUFF	4031 FMTDATE	90FA FMTD0NL	91C0 FMTD0NV	91A0
FMIERR	BCE FMIL00P	9103 FMILP	9158 FMILP2	918F FMINAME	90F2 FMIST	9152
FMTWDON	9324 FMTWIR	49E5 FMTYPE	917C FNO	B272 FNI	B270 FN2	B28A
FN3	8295 FN4	B2A0 FNDFILO	B7C5 FNDFIL1	B7CB FNDFIL2	B7F1 FNDFIL3	B708
FNDFIL4	8702 FNDFILE	B719 FNDFSPOT	C030 FNGRIV	B463 F01	B3F9 F02	B3F0
F02ZK	B303 F03	B400 F03X3	83CE F03XX	B3EF F03YY	B3EC F04	B430
FORCONF	9015 FORMAT	4920 FORMIT	9111 FORSEC	682E FPCRET	B42C FPREEF	402A
FREEG	4028 FRSTG	5A85 FRSTH	5A8B FRSTI	SAC2 FSPEC1	B6E4 FSPEC2	B6EF
FSPEC3	86FC FT10	9031 FTR	6003 GAP0BL	A3EC GAPSNG	A308 GATBUFB	0600
GATREP	8940 GBLS1	6F8A GCOUNT	6C64 GCOUNT1	6C74 GEODE	88AB GEOLAST	B881
GET10	966A GET1B	8821 GETADDR	A6AA GETA0R1	A6AD GETBC1	61BA GETBC2	6106
GET1BCNT	6186 GETBYT1	6194 GETBYT2	61B3 GETBYT	6191 GETCF	CEC3 GETCF1	CED0
GETCFN	0004 GETCNF	C0E0 GETCNFA	CF48 GETCNFB	CF5F GETCNFC	CF57 GETCNF1	6062
GETDAT	5A55 GETDIR	5A69 GETDIRD	5506 GETDLP	8F3A GETD0RS	8F0B GETDT	A025
GETD10	A02E GETD0B	8F0E GETD0N	8F5E GETE	6766 GETEXTS	B210 GETFADR	BA44
GETGAT1	8013 GETGAT2	8005 GETGAT3	8012 GETGAT4	8016 GETGAT5	8010 GETGCNT	7FFF
GETIN0T	8F63 GETIP3	7984 GETPUTA	793F GETSE	A7C4 GETSED	A707 GETSEL	5965
GETSES	A770 GETSES1	A783 GETSTR	5010 GST	6C19 GHOKI	889E GHOKO	8894
GOBACK	59E1 GORG0	6F6A GOBG01	6F81 GOBLST	6F89 GOBYT	6F52 GODO0IS	8013
GODSKLL	6608 GODSKLM	65F5 GOEO	8040 GOEOAS	8088 GOEOF5	8088 GOFLS	444F
GONKEY	5003 GOREST	67E8 GOTSPIN	5555 GOTSPOT	5560 GOTABL	59A3 GOTKEY	519A
GOTISL	7630 GOTSTR	761A GOTSTRC	763B GOTSTRZ	7625 GOTTT	6601 GOTXCC	7633
GOTXCL	7A70 GOVAL	5C22 GOXE0F	87A9 GOYE0F	8780 GOZE0F	87A6 GRNSIZE	BAE6
GTOU	5820 GTDUV	5835 GTDV	5836 GTKO	8841 HACKD	6970 HACKD0	6974

## SU Plus 3.1a - Symbol Table - Model I

HADAM	72A4 HA1BM	695D HALFLEN	1580 HASD	697A HASDAT	4083 HASDATA	9557
HASDMXX	A63E HASDMXY	A657 HASDMXZ	A64D HASFLAG	9571 HASH	8578 HASHLP	8581
HASHOK	858F HAVOBRL	064C HAVEIT	S131 HAVTAB	598E HYBT	C704 HEADCOL	0020
HEADCON	0171 HER10	6780 HEXCV	5C08 HEXIT	67AF HEXTST	5CE7 HFLG	5604
HGHG81	C24F HH15	8573 HISEC	67C8 HITBUFF	0700 HITLP1	8563 HITLP2	8565
HITREP	89F7 HLDIR	8884 HLPUAY	670E HOLDER	5602 HOLDIN	5AE2 HTLHV	9298
HTNNRM	9280 HTNRM	9282 HVHTO	B76F HYTON	728B HYTRS	728A IB0	500E
I85	50EC 19478	C3A6 IDBOT	7920 IDDBL	7940 IDDEN	7870 IDDOWN	7914
IMARKS	7807 IDMK	79C0 IDOMK	79F3 IDNANG	7A3F IDNOT	7973 IDRET	78CF
ID51DD	7950 ID51D1	7968 ID51NL	7930 IDTABLE	7B81 IDTOP	7925 IDUP	7937
IOUPX	7903 IOXCHG	7A34 IFBFMT	98FE IFBREAK	4301 IFCLEAR	6108 IFCOPY	C1F3
IFCOPY1	C324 IFSAME	610C IFTHER	A175 IFTHER1	A181 IFTHER2	A189 IFTHER3	A1BE
IFTHER4	A19A IFX2	7A04 IFX2A	7A19 IGH	9048 INCOEG	6FB0 INCOEL	6FB3
INCDECO	6F9E INCDEQ	6FC8 INCDEV	6FC4 INCDECX	6FB6 INCOFF	6F96 INCPS1	C4CC
INCPAS2	C4CE INCPASS	C4A9 INITB0	SF02 INITBD1	SF07 INITB02	SF12 INITB03	SF08
INITDRV	620E INITDV	62E0 INITD0	6209 INITGAT	9097 INITLP	62EB INKEY	5904
INP1	A820 INPIX	AB25 IMPORT	A9EE IMPORTB	A1A1 INPUT	D1AE INSBYTE	6EEE
INSGTC	C7EC INSGTU	C823 INSGTV	C841 INSGTX	C801 INSGTXX	C804 INSGTY	C81A
INSGTZ	C815 INSHIT	8581 INSHIT3	8580 INTACNT	6040 INTCNT	6046 INTFDC	5951
INTGTL	90DA INTGTM	90E4 INTLGT	9004 INTBL	B646 I0COMM	5456 IPDEN	7876
ISDEN	7874 ITCONT	67C0 IX01R	88AA IX01RB	88C4 IX01RBB	880F IX01RDD	88E0
IXSAVX	B4E0 IXSAVY	BE74 JUMPMEM	A8A8 JUMPROT	A8CA KB0R0	3801 KB0R1	3802
KB0R2	3804 KB0R3	3808 KB0R4	3810 KB0R5	3820 KB0R6	3840 KB0R7	3880
KCATLP	8659 KACTGO	8651 KEXT	866F KEY	50CA KEYBRO	40BC KEYTABLE	4093
K13	514F KICAT	8608 KIGO	511A KILCATC	8612 KILOISK	0143 KILDSK	0158
KILEXT	81E3 KILLCAT	8603 KILLCT	8688 KILLIT	81C0 KILLLP	8104 KILLOP	81C0
KILLPGM	4403 KILLSYM	8404 KILLV	81E8 KILRET	81D0 KIM	5320 KIMCHK	74CC
KNEXT	8680 KSYMB	8680 KSYMBG0	8683 LARR	0050 LBLST	6EE5 LDIDR	8C94
LEN256	A3A0 LENFLG	A38E LF	000A LINEC	81AC LINFEED	4F26 LKSEC	5B69
LOADCHK	B402 LOADCKD	BACB LOADCON	D6E7 LOADDUM	BA16 LOADPAT	D765 LOCSYNC	AE17
LOGO	07C5 LOGODSP	0709 LOGOLPA	070F LOGOUT	07C8 LOKDF	9866 LOKDL	9850
LOKF1N	9874 LOKGRAN	9855 LOKIT	9742 LOKITU	976A LOSEC	67CC LOSECL	6704
LOW1	BAA7 LOW2	B480 LOW3	BAC0 LS01R	6795 LSTBYTE	6E44 LSTKEY	50F2
LXH3	5178 LXH4	5184 LXH5	518C LXH6	5192 M00	6ABB M2SEC0	AB8B
M2SEC1	A87E MADEKEC	8E8A MAKEGAT	838B MAKEHIT	8531 MAKGATL	83CB MAKTRSD	8EC2
MALDD000	60FB MASTER	4D15 MAXB0	0000 MCPL	6007 MDCMM	60CC MDLCNT	601C
MDONE	68AF MDONEY	68AA MEM2PUT	AC10 MEM2SEC	A858 MEM2TB0	AC38 MEM2TRK	AC13
MEMORY	4B77 MEMWHR	4CDB MENU	45C7 MENUJD	595E MENUJR	468A MEXEX	6B9C
MEXXE	68A0 MFLAG	4073 MGGIH	8E7C MGGIJ	8E03 MIDMEM	4037 MIDPNT	EA80
MKHI	8552 MKHINT	B550 MKT1	5139 MKT2	5142 MLADD	60E3 MNTH0	6B2C
MNT0ES	626A MNTSRC	6259 MNTSYS	6280 MNYGRN1	8E53 MNYGRN3	8E47 MNYGRNB	8E70
MNYGRN9	8E87 MNYGRNS	8E2F MOCOMM	628F MOD3FL	406F MODCONT	6661 MODO	6B89
MODDAT	6A14 MODE	6D91 MODDEX	6D94 MODDIR	70DB MODD01R0	70CD MODD0JMP	70E6
MODDRV	6A05 MODDX	6D9D MODE	401C MODELN	6FED MODFIX	6AAB MODI	FFFF
MODIFY	6A23 MODIII	0000 MOD1RB	70AA MOD1RD	7E8A MOD1RL	7E56 MOD1RLB	7E65
MODIRLN	7E67 MODIRR	7E3A MODIRRN	7E4C MODIRU	7E71 MODIRXT	7E4A MODKILL	7E20
MODL	6D12 MODE	6D36 MODELEX	6D39 MODLX	6042 MODMEM	6A0B MODR	6C87
MODRCNT	6CC1 MODE	6D06 MODEREST	7E2C MODREX	6009 MODRX	6CEA MODSEC	6A11
MODTBL	6A80 MODTBL0	6A68 MODTOP	6C0B MODTOPL	6C2E MODTRK	6A0B MODTRU	6A0E
MODTYP	69F6 MODU	6D5E MODUE	6D66 MODUEX	6D69 MODUX	6D72 MODUT	6A2F
MOTORF	B040 MOTORO	B041 MOUNTC	6147 MOVBAK	58E7 MOVBLK	8704 MOVBLK1	B706
MOVLBK5	B716 MOVLBK	B701 MOVCOMM	5523 MOVIDR0	8C9C MOVIDR1	8CD3 MOVIDR2	8098
MOVIDR3	80AE MOVROR	8C00 MOVE	5800 MOVEIDR	8C90 MOVEIN	9430 MOVEFILE	8684
MOVHEAD	5538 MOVHEM	A804 MOVHEM1	A80D MOVT1	5541 MREAD	6C2C MSERL	74FE
MSERN	750C MST1	753D MSTDOON	7517 MSTHT	9CA1 MSTRSAM	7526 MSTRSER	74E0
MU13	8559 MULT	58C9 MVERIFY	60BC MRWITE	60C8 MXODE	6A20 N000	6ABE
NACCESS	8823 NAMDATE	7A84 NAPASS	8703 NBACESS	882E NBPLP	69B8 NBPASS	87F5
NCCPCF	CEFE NC PASS	8801 NEWBTP	8062 NEWDATE	8058 NEWMEM	A601 NEWMMIE	A6E1
NEWOTS	672C NEWMEM	A60A NEWENT	BA2E NEWEXT	C430 NEWLOAD	8A10 NEWS	6732
NEWST	508E NEWSTL	50C2 NEWTS	6751 NEGLX	987A NEGLY	9882 NEGLZ	988A

# Technical Manual

TECH-111

## SU Plus 3.1a - Symbol Table - Model I

NEXGRAN	9877 NEXROW	5122 NEXSEC	5A6B NEXTUI	5A73 NMOTNO	7ACF NMIVECT	4049
NNAME	8793 NFILE	844C NOGOODV	9EFA NONSTP	SE97 NOPUT	9600 NOSPT	5EF6
NOTABF	60F0 NOTAXT	5A1B NOTCPC	CF3B NOTDIR	7F0A NOTIRM	7F31 NOTDOSP	83E9
NOTDSD	CBD4 NOTINT	5B97 NOTREG	CBDA NOTTH	8B76 NOTW	919A NPASS	87C1
NSBOOT	9649 NSBOOTH	968E NSBOOTL	002E NSBOOTM	96C2 NSBOOTP	9683 NTFS	9814
NTOCTH	77C2 NUMBER	6E35 NUMHAV	7016 NUMON	6FF7 NUMTYPE	4043 NUNUSED	7E34
NXDIR	6786 NXGGZ	9895 NXGZ	9880 NXIGRAN	9A6A NXWI	9564 0341X	B416
OCTIT	678A OOKM	7180 OFFAOK	4F61 OFFSEI	89E5 OFFVID	4F58 OFSFILE	B9DE
OLDDIR	8040 OLDET	BEAA ONEIR	5B8A ONEIRD	5B87 ONEKEY	62A8 OPENGAT	8400
OPENREL	84EF ORDER00	9AF3 ORDER01	9505 ORDER02	9517 ORDER08	9539 ORDERS	94E9
ORDERSB	9529 ORDNEW	9689 ORONEWI	9678 ORDNEW2	969C OUTPORT	AB34 OUTPRTB	AB37
OUTRNG	8632 OUTSPAC	C47F OUTSPAF	C48A OUTSPC	C49E OVERURT	C01B P103C	C5C2
PASSB	8838 PASSWRD	416A PAT1	7D8A PAT1B	07AC PAT1H	07C0 PATIL	0780
PAT2	0796 PATCAN	07C3 PATCHEN	03FF PATCHES	021B PATLD	07D0 PATLP	0771
PATREM	0783 PATPUT	9A1A PATTUSE	9416 PAUFLG	615C PAUSE	615B PAUSEO	6161
PAUSE1	616E PAUSE2	6175 PAUSEC	6183 PAUSES	6184 PER84	5384 PER86A	5386
PER810	7479 PERCOPY	C341 PERHDEC	6CA7 PERMLXP	6CAC PERTRC	44F2 PGMDEN	D400
PINV	8887 POCONT	B805 POSA	4076 POSEND	5C5D POSEXT	B920 POSHL	5B05
POSIT	88C4 POSIT10	B808 POSITBC	B8C8 POSLPD	B8F6 POSLPD0	B8F7 POSLP1	B90A
POSND	5C66 POSNPO	8883 POSSHOW	8840 POSXXX	8829 POSYYY	8836 POUT	5220
POUTO	5233 POUT11	5242 POUT12	5249 POUT13	5255 POUT14	5267 PPP00	C77C
PRADDO	403F PRBUFF	4030 PRESS	5A50 PROMPT	005F PRSIZE	4038 PRTAKO	4041
PSYS	8800 PTFWV	852E PURGE	4800 PUTWIR	4918 PUT1D	9679 PUTABC	8856
PUTABCC	8859 PUTADE	892C PUTBRC	8895 PUTCONF	0041 PUTDATA	6E31 PUTDIR	7E43
PUTD1R0	7680 PUTEOF5	85CB PUTHEX	6220 PUTHIT	8593 PUTLD	888E PUTL00	8802
PUTD1	8809 PUTL02	88F4 PUTLD4	BCBA PUTLDX	BBF4 PUTOFF	BBF7 PUTOFF1	8840
PUTOFF2	8842 PUTPER	5024 PUTPSW	C5C0 PUTPSY	ACFB PUTSYNC	AFAC PVIS	8870
PZ013	7462 RAND1	984E RAND2	9855 RAND3	9858 RANDOM	984B RARR	005E
PATSTP	017C ROCKET	7874 RODIR	7837 RODIRO	7890 RODIR0	7891 RODIR1	7845
RODIRO	7854 RPOINT	37E0 ROPR11	8924 ROPR12	891F ROPRT	37E8 RTOPTE	4079
RDWINS	534E READ1	5387 READ1	5422 READ1S	5426 READIX	5420 READHRM	6483
READNS	5378 READIR	5876 READTRL	5878 READURT	53CA REGFIX	5F27 RELGRON	8477
RELGRNS	8465 RELSEC	8820 RELTKS	564C REMDRIV	9FCB REOGEND	B939 REORGIT	8977
REORGPL	8951 REPAIR	444C REPBROUT	899F REPMCOMM	8942 REPGAT	8795 REPHIT	67A4
REPLEN	4082 REPRTSO	8A70 REPRTSX	8A48 REPWR	4862 REREAD	6E27 RESA	8867
RESB	8878 RESCEXT	820A RESCUE	55A0 RESLEC	5540 RESFDC	5956 RESLLP	81FB
RESTOP	81F4 RESTORE	5510 RESULT	4044 RET	8F62 RETADD	8A2E RETINI	62C7
RETIN2	6207 RETINMI	404C RETRY	5E70 RETRY1	SE03 RETSEC	5A79 RETURN	59E7
REVDE	7730 REVKEY	5170 REVHEN	4871 REVMEMO	A878 REVMEML	A9B8 REVSEC	78F7
REVSECL	770C REVSECT	7725 RINGBUF	52E3 RNG	8679 ROG	8902 RORGLP	8982
ROVS	0010 RPASS2	8379 RPASSL	835A RPASSW	833B RPASSWC	834A RPDIR	8905
RPD1RC	8927 RPD1RD	890E RSECT	4077 RSSSS	8882 RSSSS1	8858 RS100	4000
RS110	4003 RS118	4006 RS120	4009 RS128	400C RS130	400F RS138	4012
RS1LP	0678 RSYSDON	82E1 RSY5G0	82F7 RSYST	8279 RSY512	82D0 RSY5TC	8288
RSYSILP	8310 RSYSTM	82C1 RSYSTIN	8310 RSYSTRS	8303 RT1	659C RTAPE	AE68
RTAPED	AE6 ATRAPEL	AE96 RTTRIES	580C RWNT	5E91 RWNT1	SE70 RWTL1	5308
RTWLCL2	53F0 RTWLCL3	53FC RX0	57C9 RX1	57CF RX2	5702 RXFER	57C6
RXPBOOT	B420 S2MEMD	AB0E S2MEML	ABC9 SAVELON	0046 SAVEREG	SF16 SAVMOV	552F
SBC01	7435 SBC02	7466 SBC03	744E SBREAK	0002 SBUIL0	A2F1 SCANX	5039
SCAN2	5048 SCANB	5090 SCANBO	5065 SCANF	508C SCANFLG	4070 SCANXY	5049
SCDON	5074 SCLEAR	0004 SCOPY	9860 SCOPYCK	9C6A SCOPYKD	9C92 SCOPYLP	9C63
SCOPY0	9C30 SCPYBAK	9C48 SCRIL	4F78 SCRNPRT	5108 SCROL	4F76 SDARR	001A
SOBLOP	A31E SOBUILD	A308 SDE01	746A SOHNLN	A3B7 SOLPO	A330 SDRIVE	4071
SOSKP	A3C3 SEC2HEM	AA86 SECALLO	8934 SECBOT	6815 SECDATA	7302 SECDAIL	7412
SECOFIN	74AE SECOMOR	7424 SECON	681F SECOPUT	7455 SECLO	7420 SECLO1	743A
SECLP	9308 SECPSAS	4076 SECSEER	776A SECSERL	77F3 SECSEL	77A SECFSIN	77CF
SECTOR	6812 SECTOR	401E SECUP	6818 SEEK	5A83 SEEKWHIT	5673 SEKTNT	5492
SELD01	5782 SELDEN	5763 SELDNOK	5705 SELECT	5A43 SELERET	57BF SELLOORP	54A9
SELM1	5780 SELRS	579F SELSNG	5770 SELURT	5A8F SENTER	000E SER085	43F2
SELB7X	6344 SELRNT	449E SERIAL1	4560 SERIALS	A060 SERIALT	A068 SERIALU	0021

## SU Plus 3.1a - Symbol Table - Model I

SERIALV	AAD3	SERIALW	AAE5	SERSAV2	A5E3	SERSAVE	B521	SERSTART	4340	SETB	6446
SETBSS	97BB	SETCON	6481	SETO	6404	SETDOON	5582	SETDEC	6C22	SETOET	8F37
SETDLP	5570	SETDRV	554F	SETDSS	97C9	SETGAT	9782	SETL	63EC	SETLSS	9809
SETM	641A	SETMFL	9837	SETMFL	0008	SETHSS	970C	SETN	6430	SETNS10	8F33
SETNS2	97C8	SETSD0	C880	SETSI0A	5437	SETSI0B	5646	SETSI0C	5640	SETSIDE	5624
SETSIDX	5612	SETT	63CE	SETU	63C5	SETUPS	59CB	SETX	6456	SETYES	C9F
SFMT	8E00	SFMFT2	8EED	SFMFT3	9717	SFMFT4	971A	SFMFT5	8F02	SFMFTW	8EC0
SHB00	7C8C	SHCON	7C4F	SHDMOV	7474	SHDWT	748A	SHFLG	4046	SHL01	7438
SHL02	7472	SH0001	5FA0	SH0002	5FA4	SH0003	5FAE	SHOACNT	6023	SHOBMEM	AA92
SHOCFG	CB31	SHOCNT	6028	SHO015	6E9B	SHOD15M	6E9F	SHOEXTS	B3A0	SHOFMT	5F01
SHOFT1	8E07	SHOFIT2	8DFF	SHOFIT3	8E17	SHOFTKS	8D4C	SHOFTL	8D8B	SHOFTLL	8D82
SHOKOX	7C71	SHOMMY	A074	SHOMODE	68E8	SHONUMB	69CE	SHONUMJ	69DA	SHOREAD	5F2F
SHOSYS1	823C	SHOSYS3	824E	SHOSYX	822D	SHOVERF	5F79	SHOVERX	5FC4	SHOW	6850
SHOWDIR	7B2A	SHOWFL	8542	SHOWFL1	B540	SHOWFL2	B556	SHOWFL3	B564	SHOWFL4	B560
SHOWFLS	8571	SHOWIT	7C27	SHOWL1	7C41	SHOWLF	68F9	SHOWLP	6894	SHOWLX1	7C8F
SHOWOFF	B473	SHOWPL	68A1	SHOWPL1	689F	SHOWPP	68C9	SHOWRIT	5F54	SHOWST	5CF1
SHOWSY2	8252	SHOWSY4	8259	SHOWSYS	8223	SHOWWH	50F2	SHOX1	8421	SHOX2	B428
SHOXTL	B3A7	SHOYY1	8012	SHOYY2	8018	SHRTT	6873	SHRWCOM	5F0B	SHUGH	6885
SHVERF	5F80	SHWTLL	7C41	SIT	6E90	SJRT	688A	SKD	936B	SKERR	5536
SKKWT	B11B	SKPDD	930B	SKPDRO	935A	SKPFLSG	4457	SKPRST	53EB	SKPZU	60E0
SKVER	7001	SLARK	0018	SMOUNT	613B	SMOL0P	9C02	SORDERO	9447	SORDERB	9483
SPACE	0020	SPACES	6C97	SPACATR	C493	SPATRIN	98F3	SPBLONI	A3B1	SPBL0OK	A3B3
SPBUILD	A2CB	SPDOFF	637B	SPDON	6384	SPFFLAG	A287	SPFM1	A46A	SPFM1A	A46E
SPFM2	A458	SPFM	A400	SPPOOL	528A	SPPOOL0	529D	SPPOOLA	5292	SPPOOLB	5298
SPPOOLC	S2A4	SPPOOL	5280	SPR01	A04E	GRARR	0019	SSETUP	61E1	SSPACE	0080
STACK	415F	STAENDO	6F92	STAOFF	6FB0	STAT	567E	STAT2	5678	STEPIN	554E
STEPOUT	5559	STRAT	D191	STRING	40A0	STRINGF	7C7F	STRLEN	4081	STRMAT	7683
STRHIX	76C5	STRNEX	767B	STRSER	7404	STRSERC	765C	STRSERD	760F	STRSERL	7660
STRSERP	766F	STRTRK	9149	SUARR	0018	SUBMENU	4018	SWAP	7751	SWAPDM	7380
SWAPL	775C	SWPDMO	73CB	SYNC	50A1	SYNCHAV	AC93	SYNCNO	ACB5	TAB	0009
TABLP	59AA	TADOR	4088	TADOS	408A	TAPE	4CF7	TAPELD	AE2F	TAPEND	8005
TAPESET	AE09	TAPESHO	AE3D	TAPESH1	AE50	TAPEWR	4060	TASK	42C0	TASKCH	449D
TASKCHK	4404	TASKCHL	4401	TASKCHN	4312	TASKCHN	43F7	TASKON	4309	TASKDRV	6343
TCOUNT	4084	TDIV0	593C	TDIV01	5941	TDIV02	594A	TDIV03	594C	TEMPO	404E
TEMP1	4050	TEMP2	4052	TEMP3	4054	TEMP4	4056	TEMP5	4058	TEMP6	405A
TEMP7	405C	TEMP0	405E	TEMP9	4060	TEMPFF	407F	TESTIFIN	AA60	TESTIM1	A422
TESTH2	AA2C	TESTM3	AA35	TESTMEM	A901	TESTML	AA16	TKBO	A00D	TKCKY	44B0
TKE	ACEE	TKS	ACES	TMSSG	AA65	TMULT	5912	TMULT1	5920	TMULT2	592F
TOGALV	453B	TOGALVC	454C	TOGINK	4552	TOGSER	4559	TOGSERU	457E	TOGTRA	4530
TOGVER	458B	TOGVERU	458D	TOGHWO	4341	TOPGDB	848B	TOPGRAN	847E	TOPG513	84AE
TOPG58	8481	TOPGSS	8494	TOPMEM	4035	TOPOFIL	8624	TRACADD	44F3	TRAK	401F
TRANSE	B065	TRANES	B059	TRREAD	53B4	TRFLRD	850B	TRIFLUR	84BF	TRK2MEM	AC52
TRKB0T	680E	TRKD0	6800	TRKEN0	5A0B	TRKENDO	5A77	TRXTOP	680B	TRKUP	67F5
TRNG1	809F	TRNG2	8048	TRNG3	B0C0	TRNG4	B0B8	TRREAD	5895	TRUE	4020
TRWRITE	5860	TRYHAV	9585	TRYHAVZ	959C	TRYHV6	95D8	TRYHV7	95EF	TS01R	679C
TSKCK1	44A0	TSKCNX	445A	TSKCNY	4435	TSKCNZ	445B	TSKDONE	44E1	TSKNOT	44E4
TSKXX	4468	TSKXY	4476	TSKXZ	449A	TSUM	4086	TSVSFX	84FA	TSYSFX	851C
TSYSFX1	8504	TTO	54C6	TT1	56AF	TT2	5687	TT3	56BF	TT88	B59F
TURNSPD	6374	TURITE	53CD	TYPE	4010	TYUT	6910	U723	C413	UARR	0058
UCASE	6315	UCD	A152	UCD0	A117	UCDR	A0F7	UCFA	A169	UCH	A137
UCL	A140	UCH	A125	UCN	A157	UCNO	A16F	UCNOT	A162	UCOPY	F709
UCS	A142	UCSO	A10E	UCT	A12E	UCTB	A52B	UCTC	A0F9	UCTR	A105
UFADDR	9A8A	UFADS0	9A05	UFCCM	9A57	UFCHX	9A70	UFCPUT	9A6E	UFCRC	9A9C
UFCRND	9A6A	UFDEN	9A0D	UFED	0008	UFHEAD	9A40	UFIBM	9A72	UFLENG	9455
UFMT	9943	UFNONI	9A87	UFRAND	9AFe	UFSD	9903	UFSDF	99CB	UFSECT	9A51
UFSS	99C6	UFSSF	99B8	UFT0	9946	UFT2	996A	UFT3	997C	UFTBL	9A2B
UFTINV	9A14	UFTNS	9835	UFTNS1	9840	UFTNS2	9843	UFTNT	9818	UFTRAK	9A4A
UFTT	9989	UILP	C667	ULINE	005F	UNUSED	825E	ULP	C665	UPARSE	9A03
UPDATE	6085	UPDATE1	60E8	UPDIRX	9063	UPDIRY	9086	UPDIRZ	9090	UPDVEC	76EF
UPFEED	5014	UPFILE	B6DF	UR94	C473	URHA	7230	URPDIR	8914	USDIR	67AA

## SU Plus 3.1a - Symbol Table - Model I

USEDIX	BC24 USEGR	C27E USEGROK	C2F1 UTA000	A06E UTA005	A06B UTCX1	9FC5
UTCOPY	A471 UTCPY1	A48C UTCPY2	A503 UTCPYX	A50D UTCPMA	A26F UTCPMB	A27B
UTFMC	A266 UTFM0	A25E UTFMORE	A1AB UTFM1	A1A4 UTFM10	A1BA UTFM11	A229
UTFM12	A283 UTFM15	A28D UTFM16	A29D UTFM17	A2A2 UTLM00	A50C UTLM05	A59A
UTLDSV	A561 UTLD5VL	A5A4 UTLD5VN	A500 UTLD1	A595 UTLOAD	A543 UTLOOP	9F4B
UTLOOP1	9FAE UTLOOP2	9FCC UTLOOP3	9F01 UTLOOP4	9FDA UTLOOP5	9F00 UTLP1	9F72
UTLP2	9F88 UTLP3	9F99 UTSAVE	A553 UTSCALL	A506 VALLP	SC30 VALMAX	5C4B
VALTRM	5C3C VALUE	5BF0 VECTOR0	4062 VECTOR1	4064 VECTOR2	4066 VECTOR3	406B
VECTOR4	406A VERSEC	7042 VERSEC1	7077 VERSECO	7001 VERSKP	98E6 VI04	520B
VIDBK	521E VIDBK	5100 VIDEOJ	51E5 VIDENO	4000 VIDEO	3C00 VIDIT	51F5
VIDIT1	51F6 VIDEX	5212 VTAPE	AEF2 VTAPE1	AF32 VTAPEO	AF45 VTAPEL	AF23
VTAPENO	AEC0 WAINOT	BASC WAI0FF	BASF WAIOK	BASA WDVV	62A5 WE41	C433
WHATDEC	BE8E WHATTRS	8E44 WHDIRZ	7B2B WHDR	SE13 WHERO1	7B00 WHERDR	7AE3
WHERE	401B WHDIRY	7B35 WHDIRZ	7B1C WHSC	SE2B WHTR	SE1C WIFCI	C20B
WRBUIL0	AC0A WRDIR	7B4B WRDIRT	7B4B WRFLAG	BB42 WRFUN	6540 WRINT	37E0
WRITE	53C3 WRITE1	545F WRITE15	5463 WRITE1X	5470 WRITENS	53A7 WRITETR	5036
WRITSEC	C153 WRPR1	37EB WRREST	92EE WRTH1	92A7 WRTYPE	407A WTAPE	AF65
WTAPED	AFA1 WTAPEL	AFB0 WT13	43E8 WTRIE	5041 WTRIES	5057 WTC	9E4E
WX1	5800 WX2	5811 WX3	5815 WX4	5818 WX5	5834 WXFER	57F3
XACOUNT	40AB XBADRD	6688 XBADRDX	664B XB0X1	6663 XCOUNT	40A9 XFER1	57E6
XIDMARK	79C7 XREAD	5392 ZAP	46CF ZAPFLAG	406E ZAPN	6696 ZAPO	6690
ZAPWIR	47F4 ZBUFF	6302 ZEGRAN	8ACC ZEGRANI	8A01 ZERFIN	7242 ZEROFF	6F49
ZEPSEC	7205 ZEPSEC1	7227 ZUGRNS	8A84 ZUNGILP	8A86 ZUNGO	8A90 ZUNGOLP	8AA8
ZUMUSE	8008 ZUMUSEC	80E4 ZUMUSEL	826B			

**TECH-114**

**Super Utility Plus 3.0**

**This page intentionally left blank**

**Copyright (c) 1983 by Breeze/QSD, Inc.**

## SU Plus 3.1a - Symbol Table - Model III

\$B	425A \$D	421B \$D10	4222 \$D3	4229 \$DIF3	7981 \$DIFO	7981
\$DIFS	7A3 \$L	420E \$L10	4200 \$L3	4214 \$M	4230 \$M10	4237
\$H3	423E \$N	4245 \$N10	424C \$N3	4253 \$SECH3	0012 \$SECHO	0012
\$SECHS	0009 \$SECL3	0001 \$SECL0	0001 \$SECLS	0000 \$T	41F1 \$T10	41F8
\$T3	41FF \$U	41EA \$X	4261 \$X10	4268 \$X3	4276 \$X35	426F
\$U0	0000 \$008	0008 \$10	0010 \$18	0018 \$10DER	C0A8 \$10DSP	C0B0
\$10ERR	C8AF \$10GETB	C8C5 \$10GETL	C893 \$10HAVB	C0A0 \$10LOAD	C084 \$10LOOP	C0B0
\$10LOP2	C879 \$10LOP3	C887 \$10MER	C8A2 \$10RD	C0FC \$10RDI	C700 \$10RD2	C728
\$10RD3	C931 \$10RD4	C934 \$10ROS	C948 \$10ROL	C0FE \$10RED	C0E9 \$10REM	C0D0
\$10RES	C8E4 \$10RRE	C8E6 \$10SEK	C91E \$10SER	C0A7 \$10SL0	C0F5 \$10STRT	C0B0
\$10SER	C78A \$10SOP	C7CA \$10SER	C780 \$10SETB	C703 \$10SETL	C7A1 \$10SHVB	C7E8
\$15LOAD	C792 \$15LOOP	C776 \$15L0P2	C787 \$15L0P3	C795 \$15MER	C780 \$15RD	C80A
\$15RD1	C818 \$15R02	C838 \$15RD3	C83E \$15RD4	C848 \$15RDL	C800 \$15RED	C7F7
\$15REM	C7E8 \$15RES	C7F2 \$15RRE	C7FC \$15SER	C82C \$15SER	C785 \$15SS0	C803
\$15STRT	C78E \$20	0020 \$28	0028 \$30	0030 \$38	0038 \$30DER	C998
\$2DERR	C98E \$30GE1B	C98B \$30GETL	C982 \$30HAVB	C9C0 \$30LOAD	C973 \$30LOOP	C957
\$30LOP2	C968 \$30LOP3	C976 \$30MER	C991 \$30NNIR	C9A4 \$30RD	C9E2 \$30RD0	C9F1
\$30RD1	C9E4 \$30RD2	C2A2 \$30RD3	C9E2 \$30RED	C9F9 \$30REM	C9C3 \$30RES	C9CA
\$30RM1	CA48 \$30RRE	C904 \$30SER	C996 \$30SK	C906 \$30SLO	C908 \$30STRT	C96F
\$31	0000 \$32	0000 \$33	0000 \$34	0000 AASHOW	6E51 ABTCY	C430
ACC1	9088 ACC2	90E2 ACONF	8F54 ACNONF	8F71 ACTIV1	789E ACTIV2	78A6
ACTIVE1	7899 ACTIVE2	78A1 ADATE	8F00 ADATE1	8F23 ADDACNT	8034 ADDAPP	B075
ADOBIN	5C51 ADOBDF	5260 ADOLCP	6043 ADDONT	603A ADDDEC	5C4E ADDHEX	5C53
ADOLP	77C0 ADDOCT	5C54 ADDR	5881 ADDR20	5A0A ADDR20A	5AB0 ADDR0	5898
ADDRESS	4020 ADORPT	50E5 ADOVEC	5C35 ADJ1	7595 ADJ2	759A ADJBYTE	6E29
ADJERR	759F ADJF	6E53 ADJGRN3	8450 ADJGRNS	844C ADJSTR	7556 ADJUST1	7567
ADJSTR2	7571 ADJSTR7	75CC AE0FS	840E AGHMG	C108 AGGMG	C189 AIUWT	A124
AIUWTX	A146 AIUWTY	A151 ARBAK	9840 AKSCLR	4F60 AKSHR	9153 ALINE	B120
ALLO0	C303 ALLO00	C2F0 ALLOCAT	83B8 ALLOCFI	8538 ALLOCPL	83C4 ALLOON	C347
ALLOGO	C567 ALLO1G	C366 ALLOLP	C312 ALLOTOP	83EE ALVFLAG	4429 ALVFIBL	451E
ALVIBL	44FA ANAME	860A ANAME1	8EFD ANOBAK	9871 A0103	9385 APASS	BF26
APASSI	0F45 APASSWD	7FFE APPEND	BC40 APPEND0	BC85 APPEND1	BC88 APPEND2	BC80
ASC1	5894 ASC2	58A1 ASC3	58A6 ASC4	58A0 ASCII	5898 ASCIT	679A
ASCNUM	E621 ASKI	CC98 ASK1A	CC99 ASK1B	CCE1 ASKIC	C001 ASKID	C021
ASKIE	C047 ASKIF	C066 ASKIG	C073 ASKIH	C08A ASKII	C09E ASK2	C0C0
ASKAUTO	8010 ASKODECK	AC67 ASKF1G	CF43 ASKFILE	BF0B ASKFM1	9818 ASKGRP	892A
ASKREP1	7510 ASKREP2	7513 ASKREP3	7505 ASKSAME	8077 ASKSTR	7AE6 ASKSTRO	7504
ASKSTRE	7507 ASKSTR1	74F1 ASKSYN	AB05 ASVE	5065 AVALD	9708 B4E0FS	B4D0
BACKSP	0008 BAOCAT	85A0 BAOCLS0	SE2B BAOCNT	6048 BADCONX	6500 BADDM	7105
BADATT	5A53 BADF1	BA83 BADFILE	B3B0 BADFOR	88E0 BADFXR	680C BADISR	6689
BADMEN	A984 BADMEN2	A97B BADR0	SE31 BAOSEL	5930 BAOUWT	5E49 BAKFM10	9867
BAKFM11	986E BAKFM15	9862 BAKLP1	6819 BAKLP2	681E BAKLP3	6824 BAKLP6	6802
BAKMOD	6E04 BAKSEC	6006 BAKSPCE	50A7 BAKVGO	985F BASE	3000 BC01R	8812
BODUCL	SE12 BDTC	A64A BDWTCLS	5E17 BINASC	610C BINASC1	61E4 BINASC2	61EC
BINASC3	61FA BINIT	6792 BITD	50A0 BIT005	50CE BIT11	5019 BIT21	5081
BIT3	5057 BIT4	5040 BIT5	5025 BIT6	5008 BIT7	5CF7 BITCAN	AF53
BITIN	AF32 BITINH	AF41 BITINL	AF36 BITINL	AF22 BITOUT	AFO1 BITOUTO	AF10
BKSPA	F415 BLANK	0020 BLD001T	C4B8 BLD01L	C4AB BLD0UP	AC94 BLDRET	C493
BLKERAS	984E BLKS	6A47 BLOCK	008F BMH1	A704 BMH2	A9E4 BMH3	A9F6
BMM4	A93E BMN	C532 BOL	0010 BOLX	4F25 B0011D	C04F B0011D	0100
BOOT1S	C750 BOOT1S	00F2 BOOT3	C94F B0013L	00FB BOOTENT	9648 BOOTREP	897A
BOTMEM	4D39 BOT01F0L	B542 BREAK	0001 BRKCK1	42EF BRKCK2	42F5 BS0IR	6782
BTA	A698 BTX0	A6A3 BUFFFAD0	52AC BUFFFNO	6160 BUFFER	0400 BUFLLEN	2C00
BUFPRT	00C0 BUFPAS	407B BUFTAKE	52C1 BUILD1	72E4 BUILD1	72F3 BUILD2	9316
BUILDOI	9326 BUILDO	9303 BULK0	9818 BULKLP	9824 BULKSS	9835 BVM	C50A
BURTCPT	SE79 BYTECT	A761 BYTEL	AF29 BYTEIN	AF27 BYTEOL	AEF7 BYTEOUT	AEF5
CANNOT	7CEB CASDAT	00FF CASHOT	00EC CAAIT	8584 COATMSK	86F6 COIRMSK	B8E1
CENTR	5001 CFD	CAB9 CFZ2	CF14 CRANS	4028 CHAR0	5071 CHASH	C6A2
CHASIN	C6A5 CHAVE	BF31 CHAVE1	6A7B CHAXE	6A7F CHONAMC	7FA0 CHONANE	7F9E
CHFILE	86FB CHKDIR	8A64 CHKITX	8881 CHKOUT	8834 CHKTASK	4419 CHUNK	BE61

## SU Plus 3.1a - Symbol Table - Model III

CHUNK0	BEB4	CHUNK1	BE01	CHUNK2	BE03	CHUNK3	BE05	CHUNK4	BEF7	CHUNKS	BF03
CHUNK6	BF0F	CHUNKA	BE70	CHUNKAA	BE7F	CHUNKB	BE83	CHUNKC	BE95	CHUNKX	BE9F
CHUT	6AAA	CHUT1	6AAB	CHUT2	6AEC	CHUT3	6ABD	CHUT4	6AC3	CHUTS	6AD6
CKADDR	7927	CKADDRL	793B	CKADDRS	7945	CKASCI	794C	CKBLD	9307	CKCD	C594
CKC00	C5A5	CKC1	C5B2	CKC11	C5BC	CKC2	C5DF	CKC3	C5C4	CKC4	C5E5
CKC44	C5EF	CKC44	C5F2	CKC5	C5D4	CKCOMFD	BE10	CKCONF	6374	CKCTL	5154
CKDENO	64C5	CKDENDX	64C1	CKD1R0	8A74	CKD1RTK	8FF5	CKDLY2	S0F7	CKDSAV	S1AF
CKDSID	64A2	CKOSIE	64A1	CKOSIZE	C3FD	CKOSIZ0	C42E	CKOUL	S1A6	CKENUM	6FDE
CKEXT	8844	CKSEP	636A	CKSET	6404	CKS105	64AE	CKSL1	6517	CKSL3	651C
CKSLD	650B	CKSLP	6405	CKSL5	6506	CKSRET	6504	CKTBLX	5955	CKTBLX	5961
CKTBLY	5972	CKTKS	6326	CLEAR	0003	CLEARFL	C005	CLEARX	0522	CLEARY	001B
CLRBUFF	9887	CLRFILE	8088	CLRFLP	C040	CLRTHIS	7093	CL50	4FB8	CL502	4FED
CLS1	4FB8	CLS2	4F9E	CLSA	0006	CLSB	0007	CLSF	0005	CLSMOVE	5364
CLSL0	4FA5	CLSON	5355	CLSONOK	5361	CLSTIT	52E3	CMPENEXT	8E46	CM1M1	A8E3
CMM2	A8E0	CMOXWE	BE10	CMPADO	C703	CMPGBK	C63B	CMPGRNS	C600	CMPQOE	C188
CNFCT	C655	CNTL0B	B93C	CNTLP	6078	CNTMSG	601B	CNTOTAL	6069	CNTRET	609D
CNUFF	BF1E	CODDO0	9591	CODDOZ	9598	CODE	9598	CODE2Z	95B5	CODEZ	95B5
CODLP1	95A9	COOLP2	958A	COLS	0040	COMBI	6EE6	COMB2	6E6F	COMDEC	90E8
COMENC	905E	CONFILE	B8C7	CONFILX	BCEC	COMPIN	70BF	CONFLP	B031	COMFLP1	8088
COMFOK	B0F4	CONFILP	B04C	COMMA	002E	COMMEN	AB4A	COMMEMB	ABCD	COMMEM	A8A0
COMNO	70E9	CONMSTR	70FB	COMFOLP	B0E6	COMPAD0	C4F4	COMPARE	60E7	COMPDC3	9848
COMPCCD	9898	COMPCOE	98AC	COMPCOF	98B5	COMPDAT	86E3	COMPDIR	86C0	COMPEXT	C63C
COMPGD	C610	COMPGL	C61B	COMPPI	9036	COMPSP	7A01	COMPV	9085	COMSEC	7031
COMSECL	706D	COMSL	7092	CONF0	CAB6	CONF	8F92	CONF1	8F98	CONF2	8F0D
CONFCD	8F9E	CONFIG	C8B8	CONFSK	917F	CONRET	6380	CONTBL	6383	CONTRL	4EE7
CONTRPT	50D5	COPFILE	C07C	COPFLP	C048	COPSEC	7125	COPSECL	715F	COPWHR	4A3B
COPY	49E6	COPY1	C153	COPY1L	C218	COUNT	40A7	CPASSU	9033	CPLP1	9C4F
CPLP2	9CAF	CPLPX	9CC0	CPLPY	9020	CPLPZ	9CCC	CPOIR	C4E1	CPTYVER	9C74
CPYVER1	9C86	CR	0000	CREAFT	C459	CREATO	C445	CREATF	C456	CS01RZ	677C
CT1	A92E	C72	A938	CTAPE	ACB7	CTAPED	ACF0	CTAPEL	ACE0	CUBP	919F
CURCHR	0098	CURCHR0	0044	CURCUR	5021	CURKE	8848	CUROFF	5027	CURON	5025
CURSOR	4021	CXB0T	8C32	CXKB0	BC3D	CXTOP	BC24	CXTOPN	BC2F	DALLC	C55D
DAMARKS	71F9	DAMBUFF	0300	DAMK	5203	DAMNOTR	7900	DAMOK	7119	DAMSECL	7251
DARR	505C	DATBUFF	F700	DATERR	5070	DATHSK	5827	DBLIX	889E	DBUFF	0100
DBUILD	A249	OCT0	4172	OCT1	417F	OCT2	418C	OCT3	4199	OCT4	41A6
DCTS	41B3	OCT6	41C0	OCT7	41C0	OCTFF	5574	OCTPAS	8886	OCTTBL	41DA
DDA01	80F2	DD0A2	80FB	DD0A3	810E	DD1R0	88A3	DD1RRS	8844	DD1VO	5805
DDIV01	5808	DD1V02	5808	DD1V03	58E3	DD005FX	5630	DD0RVE	4072	DDOTCON	7890
DDOTXC	7889	DE1	7F4B	DE2	7F58	DE3	7F70	DE3LP	7F32	DE4	7F65
DE4A	7F68	DE4LP	7F37	DECBA0	6AFE	DECBLD	A647	DECBTXT	6C6B	DECDCN	6C25
DECCON1	6C32	DECCON	6B15	DECUP	6C0A	DECUP1	6C17	DECUT	678E	DECNUM	6FF3
DECODE	6B11	DECOff	6F74	DECOKG	6B10	DECUBL	6206	DECUTXT	6C6F	DEFADDR	402F
DEFDIR	585A	DEFGAT	9061	DEFL1S	585F	DEFUD	8E95	DEFUPT	5864	DELAY	548B
DELAYX	54C8	DELBYTE	AEE1	DESVE	9340	DETECO	7C50	DETEC0D	7C68	DETEC1	7C90
DETEC11	7C47	DETECOS	7C0B	DETECS	7C08	DETECT	7C29	DETPUT	7C09	OFFD0R	80E8
DEFDIR3	88F9	DF1LCN	B33E	DF1LCN	B3F4	DF1LELP	B429	DF1LEV	B432	DF1LWX	B42C
DEFFLVT	8448	DI0GOK	5C20	DI0RUN	7E66	DI0RENT	968B	DI0RLP	786C	DI0RLST	809E
DIRLSTC	80B1	DIRLST0	8123	DIRLST1	8002	DIRLST0	8000	DIRMAX	780A	DIRNOTA	88A8
DIRNOTB	88C3	DIRPAGE	4034	DIRPART	7E8B	DIRSCNT	4033	DIRTKK	9681	DIRUP	7E76
DIRX	C212	DIRY	BF4A	DISCON	6680	DISCONI	6683	DISCIL	4EE2	DISDN	4EDA
DIS0SK	6521	DIS0SK0	6592	DIS0SK00	6595	DIS0SK1	6567	DIS0SKX	6550	DIS0SKY	655A
DISFL1	B3A3	DISFILE	B384	DISFLTB	B4F1	DISHV	6691	DISHVXX	6698	DISKLP	65AA
DISLP	4EC0	DISMCN1	A550	DISMCON	A540	DISMEM	A528	DISMRET	A595	DISMBL	A59A
DISOK	4EA4	DISP1	40A3	DISP2	40A4	DISP3	40A5	DISPO	7A4B	DISPLY	4EB8
DISPN	7A3A	DISPNXN	7A26	DISRET	67C0	DISRETM	654A	DISTBL	668C	DISWT	668B
DIVD	58C8	DIV01	5800	DOK0	667C	DSRST	65FC	DLCMN	518F	DLOFF	51C4
DLON	51B3	DLY1	50ED	DLY2	50F8	DLYCNT	6C58	DMEMB	A5D0	DMEMD	A5CE
DMEMDP	A507	DMEMT	A504	DMEMU	A5CB	DMEMUP	A5D1	DMOUNT	6126	DMSV	726C
DMULT	58AE	DMULT1	58B7	DMUL2	58C0	DNAME3	4270	DO1S	80F1	DO2S	80F0
DO3S	B10F	DOADDR	A532	DOADDRD	A537	DOALLT	8965	DOAND	6BAE	DOBBLD	A62D

## SU Plus 3.1a - Symbol Table - Model III

DOBUILD	A61C DORDER0	93D6 DORDER1	93F1 DORDER2	940C DORDER8	9442 DORETRY	SE03
DOROT	6B8E DOROT1	6BCF DOSEEK	5547 DOSKIP	5E05 DOSPEED	6C56 DOWNFIL	B52B
DOWNSEC	5A5F DOXR	6CED DOYYR	8002 DOYYX	8005 DPASSV	805A DPASSV	8863
DPATRN	9877 DRIV	4074 DRIVE	4075 DRIVES	4046 DRVASC	5581 DRVCHK	B020
DRVCKB	B06E DRVCKD	B01A DRVCKOK	B055 DRVCOMM	6270 DRVLP	6299 DRVSAME	6133
DRVSTAT	B005 DSCTUP	6100 DSKDIR	700C DSKDIRS	7018 DSKFREC	8570 DSKFREE	8561
DSKLS0	5480 DSKLSL0	5483 DSGRNS	C670 DSPSEC8	C668 DSPSECS	C648 DSTAT	5600
DUMPL	9E46 DUMP2	9E58 DUMP3	9E5C DUMP4	9E56 DUMP44	9E55 DUMPS	9E68
DUMP6	9E68 DUMPASS	90F9 DUMPU	9E04 DVX1	B063 DVX2	B07C DXTIX	7E04
DYTY	7E28 EO	5781 ES	5771 E9	5796 EFAND	6E7C EFOR	6E80
EFROT	6E60 EFSHIFT	6E55 EXFOR	6E84 ENDFILE	8533 ENDMEM	FFFF ENDOFF	6F69
ENTER	0000 ENTRIES	62A5 ENTRIS	8289 ENTRY	D500 EOAS	4026 EOFB	4023
EOFS	4024 EO	001E EOLX	4F2C EPCTN	4405 ERAFILE	830A ERAFLG	8310
ERAFLH	8312 ERAFLI	8321 ERAFLJ	8328 ERMSG0	5798 ERMSG5	576A ERMSG7	5782
ETBL	00000 ETX	00000 EXCMEM	A77A EXCMEM	A77F EXCSAV	7316 EXCSEC	7276
EXCSEL	7208 EXECOPY	BFSB EXECPY0	BF66 EXECPY1	BF78 EXECPY2	BF8F EXECPY3	BFA5
EXIT	6220 EXITHT	6530 EXITHTH	854F EXITHTL	8540 EXITLP	78E1 FBG0	6EA1
FBUFF	E600 FOCMD	0DFO FOCDAT	00F3 FOCSEC	00F2 FOCSEL	00F4 FOCSТА	00F0
FOCTRK	00F1 FOFILE1	B6E5 FOR	5F00 FETCHB	B452 FETCHG0	B400 FFILCON	B402
FFLFFF	8100 FGSSV	816F FGST	68E4 FIGORV	62F8 FIGLDD0S	8694 FIGLPI	CC17
FIGM005	8678 FIGM005	66A8 FIGO1	6E82 FIGPASS	9E0B FIGPOT	9DEF FIGPSL	9E1C
FIGPSST	886A FIGPST	8877 FIGPUT	8680 FIGTK1	868E FIGTK2	86C0 FIGTKS	8669
FIGTRO	8493 FIGTRET	8686 FIGTRS	8887 FILBUFF	0700 FILBYT05	6EFF FILECT	8405
FILEDCB	415F FILELP	B08A FILEMAP	8091 FILES	4050 FILETOT	80CB FILL	9386
FILEDEF	A662 FILMLEM	A80C FILMREL	B13E FILM3	A805 FILM4	A890 FILMAPR	80A1
FILMEMI	A820 FILMGO	B1E1 FILMORD	B1C5 FILMORI	B1C8 FILMOR2	B1B8 FILMORE	8104
FILMORK	B278 FILMORY	B251 FILREL	B147 FILREL0	B129 FILREL1	B133 FILWR	4E99
FINBYTE	6E88 FINDEF	C040 FINDS0	8F4F FINGLRP	83E0 FIRS15	S480 FIX3LLP	B3d1
FIX4LLP	83A1 FIXSLLP	8383 FIXCON	4F33 FIXREAD	5603 FIXSET	5620 FIXSETO	5620
FIXS10	5631 FIXURIT	5616 FLAGA	406C FLAGB	4060 FLCOUNT	S02E FLGB0	6FD
FLICK3	5C90 FLICK4	5C80 FLICK5	5C46 FLICK6	5C59 FLICK7	5CAA FLICKS	5C58
FLICKSP	5C79 FLIPD0N	53FB FLIPG0	540B FLK1	5C67 FLK2	5C50 FLKD0N	5C77
FLKPUT	5C74 FLKRES	5C5C FLKSD0N	5C85 FLKSRET	5C85 FLKSUT	5C8A FLKV1	5C61
FLOORP11	B226 FLOORP12	B23F FLPDN	5D0F FLSCUR	6F02 FLNUM	6FA9 FLSSPD	6F05
FMI	B258 FM2	B269 FM3	B274 FM4	B285 FM5	B293 FM6	B2A2
FMTBUFF	4031 FMTDATE	90E6 FMTDOML	9134 FMTD0NV	9121 FMTERR	BC00 FMTLOOP	9077
FMLP1	90CC FNLPL2	9103 FMINNAME	9064 FMFT1	90C6 FMW0ON	9299 FNTWHR	4909
FMTYPE	90F0 FND	B108 FNIN	B196 FN2	B1A3 FN3	B1A4 FN4	B189
FNFIL0	860E FN0FIL1	86E1 FN0FIL2	870A FN0FIL3	B6F1 FN0FIL4	B6E8 FN0FILE	B632
FNOSPOT	BF49 FNGRMRV	B3E5 FO1	B312 F02	B316 F02ZK	B2EC F03	B319
F033X	B2E7 FO3XX	B308 FO3YY	B305 F04	B356 FORCONF	BF89 FORMAT	4921
FORMIT	9005 FORSEC	680A FPCRET	B545 FREEF	402A FREEG	4028 FRSTG	5A70
FRSTH	SA98 FRST1	SA9D FSPEC1	85FD FSPEC2	B608 FSPEC3	B615 FT10	97A5
FTR	SE64 GAOPLB	A323 GAPSNG	A30F GATBUF	0500 GATREP	8921 GLST1	6F63
GCOUNT	6C40 GCOUNT1	6C50 GEODE	87C1 GEOLAST	87CA GET10	950E GET18	8A3A
GETAODR	A50F GETA0R1	A5E2 GETBC1	619D GETBC2	6189 GETBCNT	6199 GETBYT1	6177
GETBYT2	6196 GETBYTE	6174 GETCF	C0DC GETCFL	C0E9 GETCFN	CF10 GETCNF	C0F7
GETCMA	CE61 GETCMF	CE70 GETCMF	CE70 GETCNT	6045 GETDAT	SACD GETOIR	54CF
GETD10	54EC GETDLP	6EAE GETDVS	8E7F GET10	AC63 GET1010	AC6C GET0VB	BE82
GETD0N	BED2 GETE	67C2 GETEXTS	819A GETFADR	8970 GETGAT1	7F95 GETGA12	7F87
GETGAT3	7F94 GETGATA	7F98 GETGATS	7F92 GETGCT	7F81 GETHM01	8E07 GETNM1B	5586
GETNMIC	55C7 GETP3	97CD GETPUTA	9783 GETSE	A6F9 GETSED	A70C GETSEL	5939
GETSES	A6A5 GETSE1	A688 GETSTR	5011 GFST	68F5 GHOKI	8812 GHOKO	8808
GOBACK	59B5 GOBGO	6F43 GOBGO1	6F54 GOBLST	6F62 GOBYIE	6F2B GODO015	B42C
GOODSKL	65E5 GOODSLM	6502 GOEO	87C6 GOEOAS	B707 GOEOF5	8701 GOFLS	4449
GOODKEY	50C7 GOREST	67C4 GOTSPIN	5538 GOTSPOT	5543 GOTABL	5977 GOTKEY	518E
GOTSL	7587 GOTSTR	75A1 GOTSTRC	75C2 GOTSTRZ	75AC GOTTT	66AD GOTXCL	758A
GOTXCL	7A03 GOVAL	5C05 GOXE0F	B6C2 GOYE0F	86D6 GOZE0F	86BF GRNSIZE	8A5A
GTOU	5808 GTOU	5818 GTOU	5819 GTOKO	8AB5 HACKO	694C HACKOO	4950
HADAM	7228 HA1BM	6939 HALFLEN	1600 HASD0	6956 HASDAT	4083 HASDATA	94CB

## SU Plus 3.1a - Symbol Table - Model III

HASDMXX	A573 HASDMXY	A58C HASDMXZ	A582 HASFLAG	94E5 HASH	84FD HASHLP	8503
HASHOK	8511 HAVEIT	5125 HAVTAB	5992 HBYT	C6ED HEADCOL	0020 HEADCON	008A
HERIO	675C HEXCV	5C8B HEXIT	6788 HEXTST	5CCA HFLG	572E HGHGBL	C168
HVIS	84F5 HISEC	67A4 HITBUFF	6800 HITLP1	84E5 HITLP2	84E7 HITREP	8768
HLDIR	882A HLPWAY	678A HOLDER	5726 HOLDIN	5736 HTLVH	920F HTNNRM	9201
HTNRM	91F6 HWHTO	8888 HYTON	723F HYTRS	7230 180	5002 185	50E0
I9478	C28F IDBOT	7884 IDDBL	7804 IDDEN	77F7 IDDOWN	7898 IDMARKS	778E
IDMK	7947 IDMOK	7797 IDNAME	79C5 IDNOT	78FA IDRET	7854 IOS100	78E4
IOS101	78EF IDSNGL	78C4 IDTABLE	7868 IDTOP	78AC IDUP	788E IOUPX	788A
IDXCHG	78BA IFBFMT	9872 IFBREAK	42FB IFCLEAR	61B8 IFCOPY	C1DC IFCOPY1	C230
IFSAM6	60EF IFITHER	A0C1 IFITHER1	A0C0 IFITHER2	A0D5 IFITHER3	A0DA IFITHER4	A0E6
IFX2	78C IFX2A	799F IGH	901C INCDECG	6F96 INCDECL	6F8C INCDECO	6F77
INCDECQ	6FA1 INCDECV	6FA3 INCDEX	6FBF INCOFF	6F6F INCPS1	C3E5 INCPS2	C3E7
INCPASS	C3C2 INITBAD	SEES INITB01	SEEA INITB02	SEFS INITB03	SEEI INITDRV	62B8
INIT0V	62B0 INITIVO	628A INITIGT	9008 INITLP	62C8 INKEY	59A8 INPI	A4A2
INPIX	AA5A IMPORT	AA23 IMPORTB	AA26 INPUT	00C7 INSBYTE	6EC7 INSGTC	C705
INSGTU	C73C INSGTV	C754 INSGTX	C71A INSGTXX	C710 INSGTY	C733 INSGTZ	C72E
INSHIT	8533 INSHIT3	8532 INTACNT	6023 INTCNT	6029 INTFDC	5927 INTGTL	904E
INTGTM	905A INTLGT	9048 INTVBL	858F ICOMM	5445 IPDEN	77F0 ISDEN	77F8
ITCONT	679C IXDIR	881E IXDIRB	6838 IXDIRBB	8853 IXDIROO	8854 IXSAVX	B3F9
IXSAVY	8800 JUMPMEM	A7E4 JUMPRT	A7FF KBORU	3801 KBOR1	3802 KBOR2	3804
KBDR3	3808 KBDR4	3810 KBDR5	3820 KBDR6	3840 KBDR7	3880 KCATLP	8508
KCATTGO	8503 KEXT	85F1 KEY	508E KEYBRO	408C KEYTABL	4093 K13	5143
KICAT	8650 KIGO	510E KILCATC	8594 KILDISK	005E KILDISK	0071 KILEXT	8165
KILLCAT	8585 KILLCT	863A KILLIT	8142 KILLLP	8156 KILLOP	814F KILLPGM	44C0
KILLSYM	8658 KILLW	8160 KILRET	0086 KIM	5312 KIMCHK	7453 KNEXT	8602
KSYM8	860C KSYM8G0	8635 LARR	0050 LBLS7	6E8E LO10R	8840 LEN256	A2E4
LENFLAG	A2C5 LF	0004 LINEC	812E LINFEED	4F1A LKSEC	5B4B LOADCHK	891B
LOADCKD	B9E1 LOADCON	0581 LOADDM	892F LOADPAT	062F LOCSYNC	A033 LOGO	068F
LOGODSP	06A3 LOGOLPA	0649 LOGOUT	0695 LOKOF	970A LOKOL	97D1 LOKFIN	97E8
LOGRAN	97C9 LOKIT	9886 LOKITWO	960E LOSEC	67AB LOSECL	67B0 LOWI	89C0
LOW2	B9C9 LOWJ	89D9 LS01R	6771 LS1B7TE	6EB0 LSTKEY	50E8 LXH3	516C
LXH4	5178 LXHS	5180 LXH6	5186 M00	6A97 M2SECD	AAC0 M2SECL	AAB3
MAKEDEC	8E28 MAKEGAT	833D MAKEHIT	8483 MAKGATL	834A MAKTRSD	8E36 MALDDDD	600E
MASTER	4015 MAXBO	0000 MCPL	608A MCOMM	60AF MODLCONT	6CF8 MOONE	6888
MDONEY	6886 MEM2PUT	A852 MENZSEC	A9A0 MENTB1D	A874 MEM2TRK	A848 MEMORY	4868
MEMMMR	4CCC MENU	4588 MENU00	5932 MENUHMR	46A6 MEXEX	6878 MEXXE	687C
MFLAG	4073 MGG1H	8062 MGG1JJ	B0EC MIDMEM	4037 MIDPNT	E400 MKHI	8404
MKHINT	8402 MKT1	5120 MKT2	5136 MLADD	60C6 MNOTH	6B9E MNTDES	6247
MNTSRC	6236 MNTSYS	6250 MNYGRN1	80C7 MNYGRN3	80B8 MNYGRN8	B0E4 MNYGRN9	80F6
MNYGRNS	8043 MOCOMM	626C M003FIL	405F MODCONT	6A30 MODO	6065 MODDAT	69F0
MODE	6060 MODDEX	6070 MODDIR	7030 MODDIR	704F MODDJMP	706B MODDRV	69E1
MODDX	6079 MODE	401C MODELN	4FC4 MODFIX	6A84 MODI	0000 MODIFY	67FF
MODI111	FFFF MODIRBK	702C MODIRD	7E0C MODIRL	700B MODIRLB	70E7 MODIRLN	70E9
MODIRR	70B8 MODIRRN	70CE MODIRU	70F3 MODIRXT	70CC MODKILL	70A2 MODL	6CEE
MODEL	6012 MODELEX	6015 MODELS	601E MODEMM	69E4 MODR	6C93 MODRCNT	6C9D
MOORE	6C62 MODREST	70AE MOREX	6C55 MODRX	6CC6 MODSEC	6FED MODTBL	6A5C
MODTBLO	6444 MODTOP	6CB7 MODTOPL	6CBE MODTRK	69E7 MODTRU	67EA MODTYP	69D2
MODU	603A MODUE	6042 MODUEX	6045 MODUX	604E MODUT	6A0B MOTORF	AF69
MOTORO	A5F0 MOUNTC	612A MOVBAK	58CA MOVLK	8610 MOVLBK1	861F MOVLK5	862F
MOVBLK8	B61A MOVCOMM	550B MOVDIRO	8C10 MOVDIR1	8C47 MOVDIR2	80DC MOVDIR3	8022
MOVDRR0	8C41 MOVE	58C0 MOVEDIR	8C04 MOVEIN	93B1 MOVEFILE	85CD MOVEHEAD	5520
MOVMEM	A739 MOVMEML	A742 MOVUT	5525 MREAD	60A5 MSERL	7485 MSERN	7493
MST1	74C4 MSTDDON	749E MSTMTN	9C15 MSTRSAM	74A0 MSTRSER	7474 MU13	8408
MULT	589F MVERIFY	609F MWRITE	60AB MXODE	69FC N000	6A9A NACCESS	8797
NAMDATE	7A0C NAPASS	8747 NBACESS	6782 NBLP	6997 NBPASS	8769 NCPCPF	CE17
NCPASS	8775 NEWBTP	8C04 NEWDATE	7F00 NEWDOME	A606 NEWDOME	A616 NEWOTS	6708
NEWEMEM	A60F NEWENT	8947 NEWEXT	C349 NEWLOAD	8936 NEWS	670E NEWST	5082
NEWSLT	5086 NEWTS	6720 NEXGLX	97EE NEXGLY	97F6 NEXGLZ	97F8 NEXGRAN	97E8
NEWROW	5116 NEXSEC	5A43 NEXTUI	5A4E NMOTNO	7A55 NMIRET	550D NMIVECT	4049

## SLI Plus 3.1a - Symbol Table - Model III

NNAME	8710	NOFILE	83C5	NOGOOD	9E6E	NONSTP	5E7A	NOPUT	95F4	NOSPT	SE09
NOTABF	6033	NOTAXT	59F6	NOTCCP	CES1	NOTDIR	7EBC	NOTDIRH	7EB3	NOTDOP	836B
NOTOSD	CAED	NOTINT	59B0	NOTREG	CAE3	NOTIT	8AEA	NOTIU	910E	NPASS	8735
NSBOOT	9610	NSBOOTH	9632	NSBOOTL	002E	NSBOOTM	9636	NSBOOTP	9627	NTFSS	9788
NTOOTH	7749	NUMBER	8EDE	NUMHAW	6FEF	NUMON	6F00	NUMTYPE	4043	NUNUSED	7086
NXDIR	6762	NXGGZ	9809	NXGZ	9801	NXIGRAN	8A5E	NXUI	940B	O34IX	B12F
OCTIT	6796	OKM	7110	OFFAOK	4F55	OFFSEI	8BFE	OFFVIO	4FAC	OFFFILE	BBF7
OLDDIR	8CC1	OLDENT	8BC3	ONEDR	586D	ONEDRV	586A	ONEKEY	6285	OPENGAT	845F
OPENREL	8471	ORDER00	9467	ORDER01	9479	ORDER02	9488	ORDER08	94A0	ORDERS	945D
ORDERSB	9470	ORDNEV	95F0	ORDNEV1	960C	ORDNEW2	9610	OUTPORT	AA69	OUTPR1B	AA6C
OUTRNG	8548	OUTSPAC	C398	OUTSPC	C3A3	OUTSPC	C3B7	OVERURT	BF34	P103C	C40B
PASSB	BA54	PASSWD	416A	PAT1	0654	PAT1B	0676	PAT1H	068A	PATIL	D657
PAT12	0660	PATCAN	0680	PATCHEN	02FF	PATCHES	0131	PATLD	0667	PATLP	D63B
PATREM	0640	PATPUT	938E	PATTUSE	938A	PAUFLG	613F	PAUSE	613E	PAUSED	6144
PAUSE1	6151	PAUSE2	6158	PAUSEC	6166	PAUSES	6167	PER64	5376	PER84A	5378
PERB1D	7420	PERCOPY	C25A	PERHDEC	6C83	PERMLXP	6C8B	PERTRC	44E4	PGMEND	0300
PIN1	8787	POCONT	879E	POSA	4076	POSENO	5C40	POSEXT	8839	POSIL	5888
POSIT	B700	POSITO	B7F5	POSITBC	0051	POSPLD	8817	POSPLD0	8810	POSPL1	8823
POSND	5C49	POSNOPE	87CC	POSSHOW	8759	POSXXX	8742	POSYYY	874F	POUT	5214
POUTD	5227	POUT11	5236	POUT12	5230	POUT13	5249	POUT14	5256	PPPOO	6795
PRADDO	403F	PRBUFF	4030	PRESS	5A2B	PROMPT	005F	PRSIZE	403B	PRTAKO	4041
PSYS	8801	PTFW	8447	PURGE	4801	PURWR	490C	PUT10	95E0	PUTABC	87CA
PUTABC	8701	PUTADE	8845	PUTBBC	8809	PUTCONF	CFBA	PUTDATA	6E0A	PUTDOR	7E25
PUTD1RB	7E32	PUTEOF5	84E4	PUTHEX	6210	PUTHUT	8515	PUTLD	8A07	PUTLDO	BAEB
PUTD1D1	8AF2	PUTLD2	8812	PUTLD4	88CD	PUTLOX	8800	PUTOFF	8A89	PUTOFF1	8A89
PUTOF2	8A8B	PUTPER	5018	PUTPSW	C4F5	PUTSNY	AEE4	PUTSYNC	AEC0	PVIS	87F1
PZ013	73E9	RAND1	9AC2	RAND2	9AC9	RAND3	9ACF	RANDOM	9ABF	RARR	005E
RAT5IP	00B5	RO3F IX	57C5	RODIR	7A80	RODIR0	7B1F	RODIRD	7B13	ROD1R1	7AEB
ROD1R0	7A0A	ROINT	00E0	ROPR11	8898	ROPR12	8893	ROPR1	00F0	ROTYPE	4079
ROWINS	5340	READ0	5348	READ1	5416	READ10	541A	READIX	5421	READNR	665F
READNS	5370	READTR	5838	READTRL	5840	READURT	53B8	REGF1X	5F04	RELGRN	0379
RELGRNS	83E7	RELSEC	8739	RELTKS	56A1	REMORIV	9C6F	REOGFND	88AC	REORGIT	8890
REORGLP	8864	REPAIR	44A0	REPBOOT	8913	REPCOMM	8916	REPGRAT	8909	REPHIT	890E
REPLEN	4082	REPRSO	89E4	REPRSX	898C	REPRWR	4B56	REREAD	6E00	RESA	8780
RESB	8774	RESECT	8161	RESCUE	8173	RESELEC	55FD	RESFDC	5928	RESLLLP	8170
REST0P	8176	RESTORE	54F6	RESULT	4044	RET	80E6	RETADO	5A09	RETNI	6244
REIN2	6284	REINH1	404C	RETRY	5E78	RETRY1	5E86	RETRYC	5A54	RETURN	5988
REVDE	7687	REVKEY	5164	REVMEM	A7A6	REVMEMD	A7B0	REVMEML	A7C1	REVSEC	767E
REVSECL	7693	REVSECT	76AC	RFIF1	5789	RFIF2	57C1	RINGBUF	5205	RNG	8592
ROG	8868	RGRLGP	8898	ROWS	0010	RPASS2	82FB	RPASSL	82DC	RPASSU	8280
RPASSWC	82CC	RPD1R	8879	RPD1RC	8898	RPD1R0	8892	RSECT1	4077	RSSS5	8798
RSSSST	8771	RST08	4000	RST10	4003	RST10	4006	RST120	4009	RST128	400C
RS130	4010	RST38	4012	RSTLP	0545	RSYSDN	8263	RSYSG0	8221	RSYST	81FB
RSYS12	825C	RSYSTC	820A	RSYSTLP	8292	RSYSTM	8243	RSYSTM	829F	RSYSTRS	8285
RT1	6579	RTAPE	A067	RTAPEL	A0C2	RTAPEL	A0B2	RTRIES	5851	RUENT	SE74
RVCNT1	SE60	RVCL1	53CC	RVCL2	53E4	RVCL3	53FD	RXFER	5784	RXPBOOT	89A1
RXX1	5868	RXX2	5872	RXX3	587A	S2MEMO	AB13	S2MEML	AAE8	SAVECON	CF5F
SAVEREG	SEF9	SAVM0V	5514	SBC01	73C8	SBC02	73E0	SBC03	7305	SBREAK	0002
SBUILD	A230	SCAN	5020	SCAN2	503C	SCAN8	5084	SCANB0	5059	SCANF	5080
SCANFLG	4070	SCANXY	5050	SCDON	5088	SCLEAR	0000	SCOPY	9A04	SCOPYC	9B0E
SCOPYKO	9C06	SCOPYLP	9807	SCOPYO	9844	SCPYBAK	98C0	SCRIL	4F&F	SCRNPRT	51CC
SCROL	4F6A	SDARR	001A	SDBLDL	A26A	SDBUILD	A254	SEDO1	73F1	SDHLN	A2EE
SOLPD	A27C	SDRIVE	4071	SOKSP	A2FA	SEC2MEM	A008	SECALLO	8840	SEC001	67F1
SECDATA	7359	SECDAT	7379	SECDIN	7435	SECDOM	7348	SECON	67F8	SECPOT	730C
SECOLO	7384	SECOL1	73C1	SECLP	934C	SECPAS	4070	SECSER	76F1	SECSR1	777A
SECSERL	7731	SECSFIN	7756	SECTOP	67EE	SECTOR	401E	SECUP	67F4	SEEK	546C
SEEKWH	56C8	SECKNT	5478	SELECT	5489	SELLOOP	548F	SELVRT	5444	SENTER	000E
SER005	43EC	SER87X	433E	SERCNT	4478	SERIAL1	4561	SERIALS	AC9E	SERIAL1	AC9A
SERIALU	0021	SERIALV	A408	SERIALV	A1A1	SERSAV2	A518	SERSAVE	84A3	SERSTRI	4347
SETB	6423	SETBSS	972F	SETCON	645E	SETD	63E1	SETDON	5564	SETDEC	6BFF

## SU Plus 3.1a - Symbol Table - Model III

SETOET	8EAB SETDLP	555F SETDRV	5551 SETOSS	973D SETGAT	96F6 SETL	63C9
SETLSS	977D SETM	63F7 SETMDF	974B SETHDFL	0008 SETMSS	9750 SETN	6400
SETNM1	5587 SETNM1P	5584 SETNS10	8EAT SETNSS	973C SETSD0	CA99 SETSIDA	568F
SETS1DB	549E SETS1DC	5498 SETSIDE	567C SETS1DX	56A6 SETT	63AB SETU	63A2
SETUPS	599F SETX	6433 SETYES	C888 SFMT	8E54 SFMT2	BE61 SFMT3	968B
SFM14	948E SFM15	8E76 SFMTW	8E41 SHAD0	7CDE SHCON	7B01 SHMOV	73FB
SHOWRT	7411 SHFLG	40A6 SHL01	73BF SHL02	73F9 SH0001	5F83 SH0002	5F87
SH0003	5F91 SHOACNT	6006 SHOBREM	A9C7 SHOCFG	C4A4 SHOCNT	6008 SHODIS	6E74
SH001M	6E78 SHOEXTS	8289 SHOFMT	5F84 SHOFT1	8078 SHOFT2	8068 SHOFT3	8088
SHOFTKS	8038 SHOFTL	805C SHOFTL	8056 SHOK0	7BF3 SHOMNY	9FCC SHOMDE	688C
SHONUMB	67AA SHONUML	6986 SHOREAD	5F12 SHOSYS1	81B8 SHOSYS3	8100 SHOSYX	81AF
SHOVERF	5F5C SHOVERX	5FA7 SHOW	682C SHOWDIR	7B4C SHOWFL	845B SHOWFL1	8466
SHOWFL2	B46F SHOWFL3	8470 SHOWFL4	8486 SHOWFLS	848A SHOWIT	7B97 SHOWL1	7BC3
SHOWFL	6805 SHOWPL	6870 SHOWLX1	7C11 SHOWOFF	898C SHOWPL	6870 SHOWPL1	687B
SHOWPP	68A5 SHOWRIT	F537 SHOWST	5C04 SHOWSY2	8104 SHOWSY4	810B SHOWSYS	81A5
SHOWWH	5005 SHOX1	B33A SHOX2	B341 SHOXTL	B2CD SHOYY1	8C86 SHOYY2	8C8C
SHRTT	684F SHRUCLM	5F6E SHUGH	68A1 SHVERF	5F93 SHUTLL	7C23 SIT	6E49
SJRTT	6866 SKD	920F SKERR	551B SKKUWT	8035 SKPD0	927F SKPD0D	92CE
SKPFLSG	4451 SKPRST	530F SKPZU	6089 SKVER	9C75 SLARR	D018 SMOUNT	6120
SOLOOP	9874 SORDERD	9388 SORDERB	9427 SPACE	0020 SPACES	6C73 SPACOUT	C3AC
SPATTRN	9867 SPBLONI	A2E8 SPBL0OK	A2B8 SPBU0LD	A217 SPDOFF	6358 SP0ON	6361
SPFFLAG	A1D3 SPFM0	A35E SPFM1	A365 SPFM2	A371 SPFM3	A378 SPFM4	A395
SPFM5	A3A1 SPFM7	A337 SPOOL	527E SPOOL0	5284 SPOOLA	5286 SPOOLB	S28C
SPPOOLC	5297 SPOOL0	5242 SPRD1	979A SRARR	0019 SSEUP	61C4 SSSPACE	0080
STACK	415F ST2END0	6F68 STA0FF	6F66 STAT	5603 STAT2	56ED STEPIN	5531
STEPOUT	553C STPRAT	D0AA STRING	40AD STRINGF	7C01 STRLEN	40B1 STRMAT	763A
STRMX	764C STRNEX	7602 STRSER	745B STRSERC	75E3 STRSERD	7616 STRSERL	75E7
STRSERP	75F6 STRTRK	9080 SUARR	001B SUBMENU	4018 SWAP	7608 SWF0AM	7344
SWAPL	76E3 SUP0AMD	734F SYNC	5865 SYNCHAV	ABCF SYNCNO	ABEE TAB	0009
TABLE	597E TADOR	4088 TADDS	408A TAPE	4CEB TAPELDR	A04B TAPEND	AFF1
TAPESET	A025 TAPESH0	A059 TAPSH01	A079 TAPUHR	A054 TASK	42BC TASKCH	4497
TASKCHK	43E6 TASKCHL	43FB TASKCHM	430C TASKCHN	43F9 TASKON	4303 TASKORV	6340
TCOUNT	4084 TOIVD	5912 TDIVD1	5917 TDIVD2	5920 TDIVD3	5922 TEMPO	404E
TEMP1	4050 TEMP2	4052 TEMP3	4054 TEMP4	4056 TEMP5	4058 TEMP6	405A
TEMP7	405C TEMP8	405E TEMP9	4060 TEMPFF	407F TESTFIN	A9A2 TESTM1	A957
TESTM2	A961 TESTM3	A96A TESTM6	A906 TESTML	A748 TKBD	AC4B TKCKY	4487
TKE	AC2C TKS	AC23 THMSGG	A99A THMULT	58E8 THMULT1	58F6 THMULT2	5905
TOGALV	452F TOGALV2	4540 TOGINK	4546 TOGSER	4540 TOGSERV	4572 TOGTRA	4524
TOGVER	457C TOGVERV	4581 TOGH0	433B TOPG0B	8430 TOPGRAN	8400 TOPGS13	8430
TOPGSB	8433 TOPGSS	8416 TOPMEM	4035 TOPOFIL	B53D TRACADD	44E7 TRAK	401F
TRANGE	AF81 TRANGES	AF75 TREAD	53AB TRIFL0	8421 TRIFLUR	B308 TRK2HEM	A89E
TRKBOT	67EA TRKDN	67DC TRKEND	5A68 TRKEND0	5A72 TRKTOP	67E4 TRKUP	6701
TRNG1	AF8B TRNG2	AF4C TRNG3	AFE7 TRNG4	AF04 TRREAO	5854 TRUE	4020
TRWRITE	5808 TRYHAV	9529 TRYHAZV	9510 TRYHV6	954F TRYHV7	9563 TSDIR	6778
TSKCK1	44A7 TSKCNTX	4454 TSKCNXY	442F TSKCNTZ	4455 TSKDOME	44DB TSKNOT	44DE
TSKXX	4462 TSKXY	4470 TSKXZ	4494 TSUM	4086 TSYSFIX	847C TSYSFX	849E
TSYSFX1	8484 TT0	5718 TT1	5704 TT2	570C TT3	5714 TT88	8486
TURNPSO	6351 TURN1	5384 TYPE	401D TYUT	68EC U9Z3	C32C UARR	0058
UCASE	62F2 UCD	A07E UCDD	A043 UCDR	A043 UCFA	A0B5 UCH	A0B3
UCL	A099 UCM	A071 UCN	A043 UCNO	A0BB UCNOT	A0AE UCOPY	9E70
UCS	A08E UCSD	A05A UCT	A07A UCTBL	A460 UCTC	A045 UCTR	A051
UFADDR	9A2E UFADDS	9A49 UFCMM	99CB UFCMX	95E4 UFCPUT	99E2 UFCRC	9A10
UFCRND	99DE UFDEN	9A64 UFEED	0008 UFHEAD	99C1 UF18M	99E6 UFLENG	99C9
UFHT	9887 UFNON1	99FB UFAND	9A72 UFSD	9947 UFSDF	993F UFSECT	99C5
UFSS	993A UFSSF	9932 UFT0	988A UFT2	980E UFT3	98F0 UFTBL	999F
UFTINV	9980 UFTNS	9A49 UFTNS1	9A84 UFTNS2	9A87 UFTNT	9A8C UFTRAK	998E
UFTT	9920 UILP	C580 ULNE	005F UNSE0	81ED UOLP	C57E UPARSE	9977
UPDATE	6091 UPDATE1	60C1 UPDIRX	9C07 UPDIRY	9CFA UPDIRZ	9011 UPDVEC	7676
UPFED0	5008 UPFILE	8528 UR94	C38C URHA	71CD URPOIR	8088 US0IR	6786
USEDIX	8830 USEGR	C197 USEGROK	C20A UTAD00	9FBA UTAD05	9F87 UTCK1	9F3E

## SU Plus 3.1a - Symbol Table - Model III

UTCOPY	A3A9 UTCPY1	A3C4 UTCPY2	A438 UTCPYX	A43B UTMFA	A1B8 UTMFB	A1C4
UTFMC	A1B2 UTMFD	A1AA UTMFRE	A0F7 UTMFT	A0F0 UTMFTD	A106 UTMFTI	A175
UTFH12	A1FF UTMFTS	A1FC UTMFT6	A1E9 UTMFT7	A1EE UTLD0	A4C4 UTLD5	A402
UTLDSV	A499 UTLD5V	A4DC UTLD5W	A512 UTLD7	A4CD UTLOAD	A478 UTLOOP	9EC7
UTL00P1	9F2A UTL00P2	9F45 UTL00P3	9F4A UTL00P4	9F53 UTLOOPS	9F69 UTLP1	9EEE
UTLP2	9F07 UTLP3	9F15 UTSAVE	A4B8 UTSCALL	A50B VALLP	5C13 VALMAX	5C2E
VALTRM	5C1F VALUE	5B03 VECTOR0	A0A2 VECTOR1	A064 VECTOR2	A066 VECTOR3	A068
VECTOR4	406A VERSEC	6FFF VERSECL	7014 VERSEQ0	701E VERSKP	9B5A VID6	51F4
VIOBK	5212 VIDEO1	5101 VIDEO3	5109 VIDENO	4000 VIDEO	3C00 VIO1T	51E9
VIDIT1	51EA VIDEX	5206 VTAPE	AE0E VTAPE1	AE4E VTAPEO	A6E1 VTAPEL	AE3F
VIAPENO	A00C WAINOT	B975 WAIOFF	B978 WAIOK	B973 WDVV	6202 WE41	C54C
WF1	57CC WEF2	5709 WHATDEC	8E02 WHATTRS	8E18 WHDIRZ	7A81 WHDR	5DF6
WHED01	7A86 WHEDIR	7A89 WHERE	4018 WHDIRY	7A88 WHDIRZ	7AA2 WHSC	SE08
WHTR	50FF WIFC1	C121 WR3FIX	5700 WRBUILD	A83F WRDIR	7B2A WRDIRT	7B20
WRFLAG	B458 WRFLN	658A WRINT	00E5 WRITE	5387 WRITE1	5440 WRITE15	5451
WRITEX	545E WRITENS	5399 WRITETR	57E1 WRITSEC	C06C WRPR	D0F8 WRREST	9262
WRTHT	921B WRTYPE	407A WTAPE	AE81 WTAPED	AEB0 WTAPEL	AEAB WTK3	43E2
WTRIE	57EC WTRIES	5802 WTTC	90C2 WX1	581E WX2	5829 VX3	5834
WXFER	57C9 XACOUNT	40AB XBADRO	6664 XBADROX	6627 XBOK1	663F XCOUNT	4DA9
XIDMARK	794E XREAD	5384 ZAP	46C3 ZAPFLAG	40A6 ZAPN	6672 ZAPO	666C
ZAPUHR	47E8 ZBUFF	620F ZEGRAN	8A40 ZEGRANI	8A45 ZERFIN	71D7 ZEROFF	6F22
ZERSEC	71A2 ZERSEC1	71C4 ZUGRNS	89F8 ZUNGILP	8A2A ZUNGO	8A04 ZUNGOLP	8A1C
ZUNUSE	805A ZUNUSEC	8066 ZUNUSEL	81E0			

