ATARI®
XL ADDENDUM

ATARI HOME COMPUTER SYSTEM

OPERATING SYSTEM MANUAL

Supplement to ATARI 400/800™ Technical Reference Notes

ATARI®

Printed in U.S.A.

TABLE OF CONTENTS

1.0 INTRODUCTION

2.0 APPLICABLE DOCUMENTS

3.0 HOW THE 1200XL COMPARES TO THE A400/800

3.1 The Help Key
3.2 What the Function Keys Do

Cursor Left
Cursor Right
Cursor Up
Cursor Down
Home Cursor
Cursor to Lower Left Corner
Cursor to Beginning of Physical Line
Cursor to End of Physical Line
Keyboard Enable/Disable
Screen DMA Enable/Disable
Key-Click Enable/Disable
Domestic/International Char. Set Select

3.3 Key Redefinition

Contents of the Key Redefinition Table
Reassignment of the function keys only
Non-reassignable Keys and combinations

3.4 User-Alterable Keyboard Auto-Repeat Rate
3.5 Caps/Lowr Key Toggle Action
3.6 LED Initialization
3.7 Power-On Self-Test
3.8 Option Jumpers
3.9 Additional Hardware Screen Modes
3.10 Text Screen Fine Scrolling
3.11 Disk Communications Enhancements
3.12 Power-On Display Enhancement
3.13 Deleted Features

4.0 MEMORY MAP OF THE 1200XL

5.0 ENHANCEMENTS TO THE A400/800 REV. B OPERATING SYSTEM INCORPORATED IN THE 1200XL

Peripheral Handler Additions
General Improvements

6.0 OTHER CHANGES/GENERAL INFORMATION

Improved Handling of OS Database Variables
NTSC/PAL Timing Provisions
1200XL OS ROM Identification and Checksum

APPENDIX A — An Example of Keyboard Reassignment

APPENDIX B — Suggestions for the Construction of a New Character Set for the New Graphics Modes 12-13, and details of memory use and data interpretation for modes 12-15.

APPENDIX C — OS DataBase Changes from REV. B to 1200

1.0 INTRODUCTION

This manual is designed to serve as a supplement to the ATARI 400™ and ATARI 800™ OPERATING SYSTEM MANUAL.

The 1200XL, as shown in sections 3-5, is a technical upgrade of the A800. The operating system for the 1200XL has been written to maintain, as much as possible, compatibility with application programs which have already been developed for the A400/800.

Since the basic hardware which controls the user interface and the display is, for the most part, compatible with the earlier designs, the operating system, except for the enhancements or changes described here, has remained largely the same. Therefore the data contained in the OS manual for the A400/800 is still valid.

This manual has been written to provide the user with data regarding usage of the added features of the 1200XL operating system, with some details about the characteristics of the peripheral devices with which it will operate. Programmers or peripheral developers who require a greater level of detail regarding the handling of peripheral devices should refer to the documents referenced in item 2 of section 2 below.

2.0 APPLICABLE DOCUMENTS

1. ATARI Home Computer Operating Systems Manual.

   Describes the OS for the A400 and A800, which is the basis for the enhancements described in this manual.

2. ATARI Home Computer Hardware Manual and 1200XL Supplement.

   The Hardware Manual covers the hardware registers which control the various functions of the A400 and A800. The supplement to the hardware manual covers the added features for control of the 1200XL Home Computer. Details that are appropriate to the OS handling of such hardware registers are contained in this OS manual. The user who has need for other hardware-related data should refer to the hardware manual for more information.

3. DE RE ATARI

   This document provides the user with an introduction to the effective use of the ATARI Home Computer hardware. Although written to cover the A400/800, the data contained therein is valid for the 1200XL as well.

3.0 HOW THE 1200XL COMPARES TO THE A400/800

The following is a list of the features and functions which will be discussed in this chapter. Each will be explained in a separate section.

In this chapter, you will learn about:

1. The HELP Key
2. The Function Keys
3. How key codes are redefined and which ones cannot be redefined
4. How to alter the key repeat rate
5. The action of the Caps/Lowr Key
6. How the OS initializes the LED's on the keyboard
7. What happens when a cartridge is installed or removed
8. What happens during power-on self-test
9. What the option jumper assignments mean
10. What new screen modes the 1200XL can use
11. How to enable fine scrolling of the text screen
12. How the disk handler has been changed for improved operation
13. What kind of display is now produced at power-up
14. What features have been deleted as compared to the A400 or A800

Each of the items enumerated above corresponds to the paragraph number in this section which follows. For example, item 1 above is covered in paragraph 3.1, item 2 in paragraph 3.2 and so forth.

3.1 The HELP Key

The operating system, while watching the keyboard, will recognize the pressing of the HELP key as a request to set a flag in the OS database. This flag can be read by whichever application program is in control at the time and react accordingly.

The OS treats the help flag in the same way as the BREAK key in that no ATASCII code is produced but a database variable is set. Therefore, if your program is expecting the HELP key to be pressed, you must not only read the keyboard FIFO (hex location O2FC) for incoming ATASCII codes other than Help, but also occasionally check ("poll") the contents of the HELPFG (help flag) database variable to see if Help was requested.

After reading the database location, and deciding what to do, you must "clear" it for the next time the key will be pressed. The OS does not clear it for you. The Help Flag is cleared by storing a zero in its database variable.

The location of this variable is $O2DC. The conditions to which it responds are listed below, along with the codes which will be stored in HELPFG.

Hex value	Condition represented
OO	The Help flag is cleared. This flag is cleared at initial power-up reset and subsequently, if set, must be cleared by the application program.
11	HELP key alone was pressed.
51	SHIFT-HELP key combination was pressed.
91	CTRL-HELP key combination was pressed.

The HELP key can be used during the power-on display and during the self test feature. See those sections for more information.

3.2 What The FUNCTION Keys Do

NOTE: This section only applies to XL computers with function keys.

The 1200XL is provided with a set of four function keys. You may redefine the ATASCII values which these keys produce if you desire. As a matter of fact, the entire keyboard ATASCII output may be redefined as will be seen later. This section shows the normal definition of the F1-F4 keys, their functions and the ATASCII codes which they produce (if any) as a result of the power-on reset assignment. All values in the table below are given in hexadecimal.

FUNCTION KEY ASSIGNMENT SUMMARY

Key	If pressed alone
F1	Produces the Cursor-up function, returns ATASCII 1C
F2	Produces the Cursor-down function, returns ATASCII 1D
F3	Produces the Cursor-left function, returns ATASCII 1E
F4	Produces the Cursor-right function, returns ATASCII 1F

Key	If pressed with SHIFT
F1	See HOME CURSOR below
F2	See CURSOR TO LOWER LEFT CORNER below
F3	See CURSOR TO BEGINNING OF PHYSICAL LINE below
F4	See CURSOR TO FAR RIGHT OF PHYSICAL LINE below

Key	If pressed with CTRL
F1	See KEYBOARD ENABLE/DISABLE below
F2	See SCREEN DMA ENABLE/DISABLE below
F3	See KEY-CLICK ENABLE/DISABLE below
F4	See DOMESTIC/INTERNATIONAL CHARACTER SET below

Key	If pressed with CTRL and SHIFT
F1	Ignored
F2	Ignored
F3	Ignored
F4	Ignored

HOME CURSOR FUNCTION

SHIFT-F1 causes the cursor to move to the home position of the screen as well as producing the default ATASCII code 1C. The default function is reassignable.

CURSOR TO LOWER LEFT CORNER

SHIFT-F2 causes the cursor to move to the lower left corner of the screen as well as producing the default ATASCII code 1D. The default function is reassignable.

CURSOR TO BEGINNING OF PHYSICAL LINE

SHIFT-F3 causes the cursor to move to the far left of the physical line on which it is located (note, not the logical line which, in the screen editor, could be as many as 3 physical lines). This function is performed by the screen editor as well as generating the default ATASCII code 1E. The default function is reassignable.

CURSOR TO FAR RIGHT WITHIN PHYSICAL LINE

SHIFT-F4 causes the cursor to move to the far right side of the physical line on which it is located. This function is performed by the screen editor as well as generating the default ATASCII code 1F. The default function is reassignable.

KEYBOARD ENABLE/DISABLE

CTRL-F1 controls the keyboard enable/disable function. It produces no ATASCII code. This key combination affects the operating system handling of the keyboard and is not reassignable.

CTRL-F1 disables and re-enables all keyboard functions except for the following:

RESET	is the 6502 RESET key, and cannot be disabled
OPTION START SELECT	keys are not controlled by the operating system

Each time you press CTRL-F1, the operating system changes the enabled/disabled status to the opposite of what it was when you pressed this combination. In other words, if the OS had disabled the keyboard, LED 1 would be on. If, at that time, you press CTRL-F1, the OS would re-enable the keyboard and turn LED 1 off. The second press of this combination would reverse the process, disabling the keyboard again.

You may monitor or control the keyboard enable or disable function under software control by reading or writing the OS database variable called KEYDIS (hex location 026D). A value of 0 in this location means the keyboard is enabled, and a value of hex FF here means the keyboard is disabled.

SCREEN DMA ENABLE/DISABLE

CTRL-F2 controls the Screen Enable/Disable Direct Memory Access (DMA). It produces no ATASCII code. This key combination affects the operating system handling of the display function. This key combination is not reassignable.

The 1200XL on power-up, always enables the screen DMA. What this means is that the system will always initialize itself to display anything which has been defined for the screen display during power up. This same screen DMA enable will also occur if you touch any keyboard key other than the CTRL-F2 combination.

Various types of programs which you write may be heavily involved in arithmetic computations. To speed up the processing in the A400 or A800, you may disable the screen DMA. When it is disabled, the ANTIC processor does not steal memory cycles from the 6502 to get its data for the screen. Therefore during disable mode, the screen remains blank. When it is enabled, the full display which you have defined is visible; however, the processor is slowed down by anywhere from 10 to 40 percent as explained in the section on ANTIC DMA in the Atari Hardware Manual.

On the 1200XL, to start the higher speed/ no display function, press the CTRL-F2 key combination. The display will go blank. To restore the display again at any time, you can press any other key.

During your arithmetic calculations, you may be in continuous process of updating the memory area where the display data is contained. You can then get a status of the operation in process at any time simply by pressing any key other than CTRL-F2, then again press CTRL-F2 to re-enter the higher speed mode.

Your program, then, on completion of the calculation, could exercise direct program control over the ANTIC DMA variable to restore the display when the arithmetic intensive part is over.

The DMA control database variable SMDCTL contains status bits for display list memory access as well as player missile data access. When the combination CTRL-F2 is pressed, the OS will save this value, (if it is not already zero) in database variable location DMASAV$02DD). Then the variable SMDCTL will be set to zero. When the combination is pressed again, the original value is restored to SMDCTL from DMASAV, thereby restoring the display. Your program could perform the same process.

KEY-CLICK ENABLE/DISABLE

CTRL-F3 controls the Key-Click enable/disable function. If pressed once, it disables the audible feedback on keystrokes. Pressed again reenables it. This function only affects an OS database variable and produces no ATASCII code. It is not reassignable.

You may control the key click enable/disable from your program. All that needs to be done is to change the same flag which the operating system uses to indicate whether a key click is required. This flag is called NOCLIK. It is one of the OS database variables, contained at location $O2DB.

On power up and reset, the operating system initializes this variable to a value of OO, meaning that key click is enabled. This location, when it contains the value $FF, indicates that no key click is desired. The key combination CTRL-F3 toggles it between the values OO and FF.

DOMESTIC/INTERNATIONAL CHARACTER SELECTION

CTRL-F4 controls the domestic/international character selection. Default is domestic. It affects an OS database variable only and produces no ATASCII code. It is not reassignable. It toggles the display of character sets, changing between the two each time the key combination is pressed. When the international character set is selected, LED number 2 will be lit.

The international version of the character set is located in the ROM beginning at location $CC00 You can cause the international character set to be selected by storing the constant $CC to location $O2F4. This is the location CHBAS. The normal character set is located in the ROM starting at $EOOO. If a program stores $EO to CHBAS, it selects the display of the normal characters.

If you have defined your own character set, however, pressing CTRL-F4 will display the international character set. This is because the operating system will test CHBAS and find that the value $CC is not there. Therefore $CC must be the next value which is to be used (selects int'l. set). When it tests CHBAS and finds $CC stored there, it knows that $EO is the next value to use during the toggle between character sets.

Two variables are used to control the character set selection: CHBAS (O2F4) and CHSALT (O26B). The Screen Editor (E:) and the Display Handler (S:) initialize variables CHBAS and CHSALT at every OPEN command which you issue to either one. CHBAS is initialized to a value of hex EO and CHSALT is initialized to a value of hex CC.

When you press CTRL-F4, the operating system swaps the values of CHBAS and CHSALT using the OS variable TEMP as the temporary holding point. Once it completes the swap, if CHBAS is equal to CC, it will light LED 2, indicating that the international character set is selected.

3.3 KEY REDEFINITION

You may redefine most of the 1200XL console keys if desired. The redefinition process consists of setting up a pair of tables which can be referenced by the operating system when it translates your keystroke into an ATASCII value.

The two tables are the KEY Definition Table and the Function Key Definition Table. The operating system has a pair of data tables from which the normal definitions are made. You may define your own set of tables however, then simply tell the operating systen where they are located in memory.

One such use of key redefinition might be to experiment with other, possibly more efficient keyboard layouts, such as perhaps the Dvorak keyboard. An example is given in Appendix A of a keyboard redefinition to allow you to do such an experiment. (Over the years, the QWERTY key layout has been the accepted standard, though many people have found DVORAK to be more efficient. This would allow you to try it for yourself.)

CONTENTS OF THE KEY DEFINITION TABLE

This table allows most of the keys of the 1200XL to generate any desired ATASCII code or special internal function. The exceptions to this are listed at the end of this section. To redefine the keys, it is necessary first to define an area in memory where a 192 byte table may be stored.

Into this table, you will store the definitions of the keys which you desire. Later you will tell the operating system where this table is located so that future references may be made to it instead of the standard definition table.

The organization of this table is as follows:

Lower case convert. Group of 64 bytes	KEYTABLE _ START (Starts at user defined address) Table of lower case conversions
Shift plus key Group of 64 bytes	Table of uppercase conversions
CTRL plus key Group of 64 bytes	Table of control key conversions
	KEYTABLE _ START + 191.

The reason that each of the subdivisions of the table has 64 bytes in it is that the hardware can generate a total of 64 hardware keycodes. These codes, numbered 00-63 decimal (00-3F hexadecimal) are used to index directly into one of the three keycode tables. Which table is referenced depends on whether the CTRL or SHIFT keys is pressed.

Note that there is no table for the combination of both CTRL and SHIFT. This combination is invalid and is ignored by the operating system.

Each of the three 64 byte subsections of the table has the form:

00 code	Byte 0	contains conversion for key code 00 for key alone, key with CTRL, or key plus SHIFT. Depends on which table is accessed per which keys pressed.
01 code	Byte 1	contains conversion for key code 01
= =		
3F code	Byte 3F	contains conversion for key code 3F

The codes which you place in your table will either generate an ATASCII code (for direct character translation) or they will tell the system to perform a specific function. Specifically any code in the range of 80 to 91 hexadecimal will be treated as special by the system. This is illustrated in the table below.

CODES AND THEIR EFFECT ON THE SYSTEM AFTER TRANSLATION

CODE	EFFECT (if any)
OO thru 7F	Used as the ATASCII code only.
92 thru FF	Used as the ATASCII code only.
8O	Ignore, invalid key combination.
81	Invert the video output to the screen.
82	Alpha lock/Lower case toggle.
83	Alpha lock
84	Control Lock
85	End of file
86	ATASCII code
87	ATASCII code
88	ATASCII code
89	Key click on/off
8A	Function 1 *
8B	Function 2 *
8C	Function 3 *
8D	Function 4 *

* NOTE: When it sees these keycode translations, it is told to DO the function which is described in the Function Key descriptions. The ATASCII coded generation for the normal and shifted function keys is handled in a different table, whose description follows that for the keycode hardware translate table.

8E	Cursor to home
8F	Cursor to bottom
9O	Cursor to the left margin
91	Cursor to the right margin

The table below shows the key cap corresponding to each key code. The physical position of each key switch within the table determines the hardware code which it will generate. To determine what code it is, take the row address of the cap, and add it to the column address. The result is the hexadecimal value returned to the operating system (range OO-3F) for use in the table lookup for that key.

KEYCODE DEFINITIONS TABLE

	0	1	2	3	4	5	6	7
00	L	J	;	F1	F2	K	+	*
08	O		P	U	RET	I	—	=
10	V	HLP	C	F3	F4	B	X	Z
18	4		3	6	ESC	5	2	1
20	,	SPACE	.	N		M	/	)|(
28	R		E	Y	TAB	T	W	Q
30	9		O	7	BACKS	8		
38	F	H	D		CAPS	G	S	A

As an example the key cap "C" is in the table in row 10, column 2. This means that the hardware generates a hardware code 10 + 2 or 12 hexadecimal. Therefore, in the translation tables shown above, the function code or ATASCII code for this character will be stored in the key definition table position $12 for each of the three types of "C" which are valid (c alone, Shifted C, or Control C). You may cause each of these to perform a separate function or generate a separate ATASCII code by revising the tables.

When you have decided on how you want your keys to be redefined, you tell the operating system where it may find the definitions by storing the address of those definitions in locations 79 and 7A hexadecimal. The low byte of the hexadecimal address where you have stored the keys should be placed in location 79, the high byte is location 7A. This is defined as one of the system vectors, called KEYDEF. It will point to the default, or original key definition table at power-on reset time.

REASSIGNMENT OF THE FUNCTION KEYS ONLY

There may be times when you only want to redefine the function keys and not redefine the rest of the keyboard. The 1200XL operating system allows you to redefine only the function keys by setting up an 8-byte table in place of the 192 byte table which would have otherwise been required. The format of this table is as follows:

F1	← Lowest memory location of the table
F2	
F3	
F4	
SHIFT-F1	
SHIFT-F2	
SHIFT-F3	
SHIFT-F4	← Highest memory location of the table

When you have decided what functions each combination must perform and have built the table, change the system vector FKDEF to point to the lowest address of your table. This vector is located at memory locations 60 and 61 hexadecimal. Location 60 gets the low byte of the hex address, location 61 gets the high byte.

The same codes described in the section titled "CODES AND THEIR EFFECT ON THE SYSTEM AFTER TRANSLATION" are used in this table. However, DO NOT assign codes 8A through 8D to the same function as the key itself. In other words, do not specify that the key F1 should perform function F1, etc. since this would result in an infinite loop. (F1 sensed by the OS sends it to the function key table, which tells it to look up and perform the F1 function, which sends it to the table, and so on, with no possible exit.)

NON-REASSIGNABLE KEYS AND KEY COMBINATIONS

The following keys or key combinations are either specifically wired for special functions or are subjected to special handling by the operating system.

Even though there might be a hardware-generated key code shown in the table above, and a corresponding space in the translate tables, there is no way to reassign these functions. This is because the operating system traps the hardware code directly to perform the specified function and it never gets to the translate mode. These keys or combinations are as follows:

BREAK	—	This function is fixed as a special case in the operating system. It is sensed by the hardware.
SHIFT	—	This key is an integral part of the hardware encoding of any key function.
CTRL	—	This key in an integral part of the hardware encoding of any key function.
OPTION SELECT START	]—	All of these are directly wired to and are sensed by the GTIA circuitry.
RESET	—	Directly wired to the 6502 reset line.
HELP	—	Function is fixed by the operating system. The help function handling is described elsewhere in this manual.
CTRL-1		Controls the screen output start/stop function.
CTRL-F1		See KEYBOARD ENABLE/DISABLE above. As noted there, this function is not reassignable.
CTRL-F2		See SCREEN DMA CONTROL above. As noted there, this function is not reassignable.
CTRL-F3		See KEY-CLICK ENABLE/DISABLE above. As noted there, this function is not reassignable.
CTRL-F4		See DOMESTIC/INTERNATIONAL CHARACTER SET above.

3.4 USER-ALTERABLE KEY AUTO-REPEAT RATE

The 1200XL operating system allows you to control the rate at which a key, continuously held down, will repeat its entry to the system. This change can be done by modifying the OS database variable KEYREP, located at hex address 02DA.

This variable determines the repetition rate by counting the number of VBLANK (vertical blanking) intervals which occur. For the NTSC (60 Hz) system, the initial value of this variable is 6; for PAL systems, the value is 5. This assures a uniform repeat rate of 10 characters per second for either system. The key repeat rate equals the VBLANK rate (60 or 50 per second) divided by the KEYREP value.

Under control of this variable, the maximum "controllable" key repeat rate would be 50 characters per second on the PAL and 60 characters per second on the NTSC (screen refresh rate). This would occur with a value of 1 in this variable.

You may control the rate at which occurs before the key repeat starts. The OS database variable which controls this is called KRPDEL. Its hex address is 02D9.

It controls the number of VBLANKs which must occur between the sensing of the key pressed until the first repeat occurs. From that time on, the repeat rate is controlled as described above. The initial values used by the OS provide a 0.8 second initial delay for either NTSC (count = 48) or PAL (count = 40) systems.

3.5 CAPS/LOWR KEY TOGGLE ACTION

The CAPS/LOWR key on the 1200XL functions as shown in the chart below:

KEY COMBINATION	CURRENT STATE	NEW STATE
CAPS	Control Lock	Lower Case
CAPS	Alpha Lock	Lower Case
CAPS	Lower Case	Alpha Lock
SHIFT-CAPS	— any —	Alpha Lock
CTRL-CAPS	— any —	Control Lock
CTRL-SHIFT-CAPS	— any —	— no change —

The meaning of the terms is as follows:

Lower Case	—	All key caps respond in lower case mode
Alpha Lock	—	All alphabetic keys (A-Z) respond in upper case mode, all others lower case
Control Lock	—	All alphabetic keys (A-Z) respond as though the control key is being held down as well as the selected key

3.6 LED INITIALIZATION

The 1200XL has two LED's on the front panel called LED 1, and LED 2. LED 1, when lit, indicates that the Keyboard is disabled. LED 2, when lit, indicates that the international character set is selected. The operating system enables the keyboard and selects the domestic character set on power up and reset. Therefore these LED's will both be off.

3.7 POWER-ON SELF-TEST

During the initial power-on, the 1200XL operating system will perform the following quick check of the integrity of the system RAM and ROM:

a. Is it possible to write $FF (all ones) to all RAM locations?

b. Is it possible to write $00 (all zeros) to all RAM locations?

c. Does a checksum of the two ROM's compare to that stored within each ROM?

If any of these tests fail, the operating system will transfer control to the self-test memory test routine. Here a more thorough test of both RAM and ROM can take place.

3.8 OPTION JUMPERS

The 1200XL is provided with a set of four hardware jumpers which are designed to tell the operating system how the system is configured. As of the date of this writing, only one of the four jumpers has been assigned, specifically J1. This is specified in the table below. During the power-on sequence, the 1200XL operating system reads the state of these jumpers and stores this state in the OS database variable JMPERS, location 030E.

The bit assignments for each of the four jumpers is as specified below. The bits are all active low, meaning that if a line reads a digital zero, the jumper is installed.

BIT	FUNCTION	HARDWARE NAME
0	Self test enable (will run self test if low)	J1 (pot 4)
1-3	Reserved for future use	
4-7	Unused	

3.9 ADDITIONAL HARDWARE SCREEN MODES

The 1200XL adds direct access to the remaining special purpose display processor operating modes. The table below shows the current mapping which has been provided for the A400 and A800. The table which follows thereafter shows the added modes and the numbers which the software can use to access the extra modes.

Mode mapping common to A400/A800:

Software Mode		ANTIC MODE		GTIA MODE
0	($00)	2	($02)	0
1	($01)	6	($06)	0
2	($02)	7	($07)	0
3	($03)	8	($08)	0
4	($04)	9	($09)	0
5	($05)	10	($0A)	0
6	($06)	11	($0B)	0
7	($07)	13	($0D)	0
8	($08)	15	($0F)	0
9	($09)	15	($0F)	1
10	($0A)	15	($0F)	2
11	($0B)	15	($0F)	3

Mode mapping for 1200XL (additional):

Software Mode		ANTIC MODE		GTIA MODE
12	($0C)	4	($04)	0 (note 1)
13	($0D)	5	($05)	0 (note 1)
14	($0E)	12	($0C)	0
15	($0F)	14	($0E)	0

Note 1: The existing character sets will not provide recognizable characters for these new modes. Therefore you will have to provide the character set if you use these modes. This is done by defining the full character set, then modifying the OS database variable CHBAS to point to the most significant byte of the address at which the character set starts. CHBAS is located at $2F4.

Appendix B of this manual contains some suggestions on the method for designing a new character set to support those added modes.

3.10 TEXT SCREEN FINE SCROLLING

The screen editor (E:) supports fine scrolling of the text screen data as an option. This fine scrolling option will be enabled if the database variable FINE (hex location 026E) is set nonzero prior to issuing the OPEN command to the screen editor. Likewise, the feature will be disabled if this location is set to 00 before issuing the OPEN.

There are only two allowed values for FINE = 0 and hex FF. Other values may produce undesirable results.

During an OPEN command to the Screen Editor (E:), if FINE (026E) is hex FF, then a fine scrolling display list is created. This display list will be one byte larger than a coarse scrolling display list. In addition, the OS places the address of a display list interrupt routine into the display list vector VDSLST (0200) replacing an other vector which you might have already stored there.

When fine scrolling is enabled, the Screen Editor's display list interrupt service routine modifies the content of color register COLPF1 (D017) for the very last visible line of the screen.

When a CLOSE command is issued for the Screen Editor, if FINE is hex FF, then the address of an RTI is placed into the display list vector VDSLST (0200). For OS versions 11 and beyond, FINE is set to zero again, and the screen is reopened with a coarse scrolling display list.

The recommended manner for enabling and disabling fine scrolling is shown below:

a. Set FINE to hex FF

b. OPEN E: using an IOCB number

c. Use E: as usual; fine scrolling is enabled

d. CLOSE E:

e. If the IOCB is now open, then you are finished, otherwise continue with the next step

f. Set FINE to zero

g. OPEN E:

3.11 DISK COMMUNICATIONS ENHANCEMENTS

The 1200XL adds the capability for the resident disk handler to read and write disk sectors having variable length from 1 to 65536 bytes. The default length, as is used on the A400 and A800 currently, is 128 bytes. Both at power-on and RESET (warm start), the 128 byte sector length is established. Your program can alter this length by modifying the OS database variable DSCTLN. The location of this two-byte variable is 02D5 and 02D6 (lo byte in 02D5, hi in 02D6).

In addition to the capability to read and write variable length sectors, the 1200XL also adds the capability to write a sector to the disk without a read-verify operation always following it. This is the command 'P' which was specifically excluded in the previous releases of the operating system.

With this capability added, you have a choice of either using the verify, for system integrity (always read after write). Or you can take a chance of writing a bad sector on rare occasions but increasing your average speed of disk usage by some value related to the verify time. You may want to experiment with some of your programs with and without verify to see the results.

3.12 POWER-ON DISPLAY ENHANCEMENT

In place of the original power-on memo pad display used by the A400 and A800 (in the absence of a cartridge or disk), the 1200XL displays a dynamic ATARI rainbow. If you press the HELP key while the rainbow is displayed, the 1200XL will enter the self-test mode.

4.0 MEMORY MAP OF THE 1200XL

The following table shows how the 6502 processor perceives the various address spaces which it can access. The maximum allowable address range, with the 16 bit address of the 6502 is hexadecimal 0000-FFFF. This address range is split, by the hardware memory management circuitry, as follows:

(Note: The 1200XL uses 64K RAM's as the main system writeable memory. Addresses within those RAM's, which would normally have filled the entire memory access space of 0000-FFFF of the processor, are prevented from access by the memory manager. This allows ROM's, cartridge memory, and peripherals to occupy a part of the memory space as is noted below.)

1200XL MEMORY MAP

HEX ADDRESS	WHAT IS ACCESSED THERE	NOTES
FFFF-D800	OS-ROM or RAM if ROM disabled	1
D7FF-D000	The special purpose chips respond to the address ranges shown in the listing below D000-D0FF GTIA D200-D2FF POKEY D300-D3FF PIA D400-D4FF ANTIC D500-D5FF Any read or write to an address in this range enables the cartridge control line CCNTL on the cartridge interface (same as A400/A800. D100-D1FF, D600-D6FF, and D700-D7FF are reserved for future use. OS-ROM physically present, but cannot be accessed here.	2
CFFF-C000	OS-ROM or RAM if ROM is disabled	1
BFFF-A000	RAM, or cartridge interface	3
9FFF-8000	RAM, or cartridge interface	3
7FFF-5800	RAM	
57FF-5000	RAM, unless in self-test mode	2
4FFF-0000	RAM	

NOTES: 1. Access to the OS ROM may be disabled by wirting a zero to port B of the PIA, bit O. Access is normally enabled, with a 1 present in this bit. (When changing this bit in the register, other bits should not be changed.)

2. The self-test ROM code is physically present in the OS ROM at actual address DOOO-D7FF.. However, this area is used for the access to the memory mapped I/O devices. When the self-test feature is invoked, the RAM located from 5OOO-57FF is disabled. The memory manager re-maps the memory access such that the OS ROM physical addresses DOOO-D7FF are accessed at 5OOO-57FF. The memory manager uses port B of the PIA, bit to determine whether to access RAM or ROM in the region 5OOO-57F. If bit 7 is high, RAM is accessed. If bit 7 is low, the OS-ROM is accessed instead. (When changing this bit in the register, other bits should not be changed.)

(Port B was used in the A4OO/8OO to service the game ports 3 and 4. The use of the remaining bits of ths port are specified in Section 6 of this manual.)

3. ROM will be selected in these regions if control lines RD4 or RD5 are pulled up to +5V by the cartridge. RD4 controls ROM select in the region 8OOO-9FFF. RD5 controls ROM select in the region AOOO-BFFF.

5.0 ENHANCEMENTS TO THE A400/800 REV. B OPERATING SYSTEM INCORPORATED IN THE 1200XL

This section describes a set of enhancements which include new methods of handling peripheral products and, in a separate section, improvements in basic operations of the system.The latter might be referred to as "bug fixes".

PERIPHERAL HANDLER ADDITIONS

To accommodate a new class of peripheral devices, the operating system now includes a relocating loader, used to upload peripheral handlers through the serial (I/O interface).

In the A400/800, device handlers for the peripherals were uploaded as fixed location (absolute) object code. These handlers were loaded using a set of device inquires, or polls, known as types 0, 1 and 2. Information on types 0, 1 and 2 Poll Commands is available from Atari Customer Service.

The 1200XL adds two other types of polls to its operating system. One poll, known as type 3, is issued at power-on or reset time. The other, type 4, can be issued as a result of an OPEN command by an application program.

Type 3 Poll Command

The type 3 poll command itself is used as an "Are You There?" type of command. Associated with the type 3 poll are two other types, specifically the:

a) Poll Reset
and b) Null Poll

Poll Reset consists of the following SIO command byte sequence (refer to the SIO document for further explanation of the byte types):

Byte Position	Value (hex)
Device Address	4F
Command Byte	40
AUX1	4F
AUX2	4F
Command Checksum	Normal (checked by peripheral)

The 4F in AUX1 and AUX2 define this sequence to all peripherals as a poll reset.

After responding to a type 3 poll by sending a handler to the system, a peripheral is not supposed to respond again to a type 3 poll. The Poll Reset command, at power-up, resets all type 3 peripherals, freeing them to respond to the poll request. However, no serial bus device sends back any data as a result of a poll reset command.

Type 3 Poll (Are you there?)

There may be several types of peripherals which can respond to a type 3 poll. In types O, 1 and 2, the device address sent on the serial line specifies which exact device is being called. In the type 3 poll processing, however, the address remains fixed (4F) and the devices each respond after a specific number of poll 3 retries. In other words, during poll 3 operations, the computer doesn't know which peripherals are actually attached, but will keep asking "is anybody there" until it has reached its last retry and no peripheral has responded.

Each peripheral which does respond to the type 3 poll must be designed to count the number of retries of type 3 polls, then to respond as described below on its own specified retry slot. Each time it sees a command other than a type 3 poll, these peripherals must reset their retry counters. This allows the computer to load the handler for each peripheral which responds, then restart its poll 3 sequence (original retry number restored) to look for another poll 3 response from the next peripheral (if any).

Since each peripheral responds only once (after a poll reset), a second request at a specific retry slot causes no peripheral response and allows the next retry slot to be polled.

This poll ("are you there?") is sent as follows:

Byte Position	**Value (hex)**
Device address	4F
Command Byte	40
AUX1	OO
AUX2	OO
Command checksum	Normal (checked by peripheral)

When, after checking the retry count, it is a peripheral's turn to respond, it sends back the following data to the computer on the serial interface:

a) An ACK response byte, and

b) 1. Low byte of handler size in bytes (must be EVEN)
   2. High byte of handler size
   3. Device Serial I/O Address to be used for loading
   4. Peripheral Revision Number

These four bytes, if sent by the peripheral, will be stored in OS variables DVSTAT (O2EA hex) through DVSTAT+3. If there is a successful return to the OS (not a timeout or other problem), it indicates that there is a handler to be loaded. The loading is performed, then the type 3 poll is repeated until all retries are exhausted and no peripheral responds.

Once the device address data is received from the peripheral during this type 3 poll, it can thereafter be referenced directly on the serial bus by its address in place of the original poll address 4F.

Specific details of the actions taken by the OS after receiving an answer from a peripheral may be found in Appendix C.

Null Poll Command

This command is used as a serial bus no-operation. If any error should occur during loading of a peripheral handler or by the relocator, the system should be free to "back out" of the linking of the faulty loader and tell the peripherals that it is ready for the next one to be loaded. Since this null poll is a non-type-3 poll, all peripherals will have reset their retry counters and should be ready for another sequence of retries, looking for their own response retry slot. This maintains synchronization between the computer and the peripherals.

The structure of the Null Poll is as follows:

Byte Position	Value (hex)
Device Address	4F
Command Byte	40
AUX1	4E
AUX2	4E
Command Checksum	Normal (peripherals check it)

Type 4 Polling

This type of poll is sent out on the serial bus as a result of an application initiated request. During an OPEN command, a device which responds to a type 4 poll may conditionally or unconditionally be polled to determine if it is online and may or may not have its handler uploaded and linked to the system under control of the OS. Detailed information regarding the handling of the device under various operating conditions may be found in Appendix C.

The Type 4 Poll is a serial port command structured as follows:

- Device address of 4F hex (peripherals looking for Type 4 Poll may ignore the device address and look only for the poll command '@'; however, the device address will always be 4F hex and the peripheral may check this);

- Command is '@' (40 hex) (peripherals looking for this poll will always look for the '@' command);

- AUX1 contains the device name, which is an ATASCII upper-case letter (range 41 hex through 5A hex) (the peripheral must be assigned that device name in order to legally answer the poll);

- AUX2 contains the device number, which is an ATASCII digit (range ATASCII 1 through 9, 31 hex through 39 hex) (the peripheral may optionally use this information in deciding whethe or not to answer the poll);

- Standard command checksum (peripheral checks this).

This poll differs from the Type 3 Poll in that the device name and number is included in the poll. Therefore the peripheral need not count retries of the type 4 poll and should answer the poll as soon as the poll command is recognized. There is no limitation on the type 4 poll; the peripheral should answer its type 4 poll each time it is issued.

The peripheral response to a type 4 poll is the same as for the type 3 poll. The four response bytes are placed, by the computer, into DVSTAT through DVSTAT+3 (02EA through 02ED hex).

GENERAL ENHANCEMENTS TO THE REV. B OS FUNCTIONS

The following functions which are supported by the A400/800 Rev. B Operating System have been further enhanced by the addition of the following features:

Printer CLOSE with data in the buffer —

The printer handler will insert an EOL(end-of-line) character in the printer buffer, if one is not there, before sending the buffer to the printer on a CLOSE. This assures that the last line will be printed immediately rather than having the printer forced offline to output the final line.

Printer Unit Number Handling —

The printer handler has been changed so that it will process the unit number in the IOCB, allowing separate addressing for printers P1 through P8.

CIO Handling of Truncated Records on Read —

The CIO now places an EOL in the user's input buffer on the occurrence of either a record longer than the buffer being read or an EOF being encountered during the read attempt. This assures that all records are accessible, even if the user has not provided a sufficient buffer size, he will at least get as much of the record as he has provided for.

CIO Error Handling With Zero Length Buffer —

The CIO will return a buffer length of zero (in the 6502 A-register) when there is a handler error while effecting a zero length buffer transfer. (See CIO section in the OS manual.)

Display Handler Cursor Handling —

The display handler now accepts a screen clear code no matter what value is in the cursor X and Y coordinates.

Display Handler/Screen Editor Memory Clearing —

The Display handler and Screen editor will not clear memory beyond the end of memory as indicated by RAMTOP. Now it is possible for the user to specify the top of memory to be used by the system and to store device handlers or personal machine code in the memory area above the display. Changing display graphics modes, then, will not erase any data which has been placed in the RAM area above that assigned for use by the display or screen editor.

Rework of the Floating Point Package —

The 1200XL operating system corrects a bug in the Rev. B OS. It now produces an error status when an attempt is made to calculate the LOG or LOG10 of zero.

New ROM Vectors —

The following fixed entry point vectors have been added to the 1200XL ROM set:

```
E480  JMP PUPDIS        entry to power-on display
E483  JMP SLFTST        entry to the self-test pgm.
E486  JMP PHENTR        entry to uploaded handler enter.
E489  JMP PHULNK        entry to uploaded handler unlink.
E48C  JMP PHINIS        entry to uploaded handler inti.
```

6.0 OTHER CHANGES/GENERAL INFORMATION

This section deals with items which involve operating system changes, but which do not easily fit into any other category.

IMPROVED HANDLING OF OS DATABASE VARIABLES

During normal power-on sequence (cold start), the OS database variables from $03ED-$03FF are set to zero. During a RESET (warm start), they are NOT changed by the OS. This means that an enhanced version of the operating system in the future will be able to make use of these locations without reloading them after any RESET operation.

These bytes are all reserved for use in future OS revisions.

NTSC/PAL VERSION TIMING PROVISIONS

There are various timing differences between the NTSC (60 hz) and the PAL (50 hz) versions. To eliminate the necessity for providing a special operating system ROM set for each one, the specific timing adjustment values are handled within the single ROM set.

To determine which type of system the ROM is operating on, the operating system checks a flag within the GTIA chip and adjusts all timings accordingly. This was possible because the GTIA must be different to handle the modified display format for the 50 Hz version. By making certain timings a function of the state of this flag, it was possible to make external timings independent of the NTSC or PAL system itself.

The timing values relate to the handling of the 115 Volt cassette player (Atari 410) and the console auto-repeat rate as shown in the table below:

CASSETTE TIMINGS NOW INDEPENDENT	TIMING
Write Inter-record gap (long)	3.0 sec.
Read IRG delay (long)	2.0 sec.
Write IRG (short)	0.25 sec.
Read IRG delay (short)	0.16 sec.
Write File leader	19.2 sec.
Read Leader delay	9.6 sec.
Beep cue duration	0.5 sec.
Beep cue separation	0.16 sec.

AUTO-REPEAT FUNCTIONS NOW INDEPENDENT	TIMING
Initial delay for auto-repeat	0.8 sec.
Repeat rate	10.0 char/sec.

1200XL OS ROM IDENTIFICATION AND CHECKSUM DATA

Each of the two ROM's in which the 1200XL operating system is contained has a capacity of 64K bits organized as 8K by 8. Within each of the ROM's is a block of data organized as shown in the diagram below, to identify the ROM and to give its checksum. The checksum is tested by the operating system as part of the power up sequence.

The format of the block for the C000-DFFF ROM is as follows:

ROM Cksum (lo)		C000	Checksum which is the sum of all bytes in ROM except checksum bytes themselves.
ROM Cksum (hi)		C001	
D1	D1	C002	Revision date having the form DDMMYY where D = day digit M = month digit. Y = year digit. Each a 4 bit BCD digit. Reserved, contains $00 for the 1200XL.
M1	M2	C003	
Y1	Y2	C004	
Option byte		C005	
A1		C006	Part number having the form AANNNNNN, where A's represent ASCII characters, N are BCD digits.
A2		C007	
N1	N2	C008	
N3	N4	C009	
N5	N6	C00A	
Revision No.		C00B	

The format of the identification block for the E000-FFFF ROM is as follows:

D1	D2	FFEE	Revision date having the form DDMMYY where D = day digit M = month digit. Y = year digit Each a 4 bit BCD digit.
M1	M2	FFEF	
Y1	Y2	FFF0	
Option byte		FFF1	Hardware product identifier, will be used by Atari to identify Home Computer products, for 1200XL = $01.
A1		FFF2	Part number having the form AANNNNNN, where A's represent ASCII characters, N are BCD digits.
A2		FFF3	
N 1	N 2	FFF4	
N3	N4	FFF5	
N5	N6	FFF6	
Revision No.		FFF7	
ROM Cksum (lo)		FFF8	Checksum which is the sum of all bytes in ROM except for checksum bytes themselves.
ROM Cksum (hi)		FFF9	
vector table for for NMI, RES and IRQ		FFFA-FFFF	This area reserved for power-on reset vectors, NMI and IRQ vectors.

PORT B CHANGES

Port B of the PIA is a read/write port which no longer is connected to game I/O ports. Instead, its bits control various functions which include control of LED 1, LED 2, read enable of the the OS ROM's and other functions. To change only one single bit at a time within that port, the following technique should be used.

```
    Clear A Bit (bit b)
        LDA PORTB
        AND # $FF-b
        STA PORTB ;clears only bit b in the port
    Set A Bit (bit b)
        LDA PORTB
        ORA # b
        STA PORTB ;sets only bit b in the port
```

XL PORTB ($D301) BIT ASSIGNMENTS

BIT	VALUE	USE
0	0	OS ROM DISABLED, RAM ENABLED
	1	OS ROM ENABLED
		The memory region mapped to the OS ROM is from $C000 to $FFFF except for the region from $D000 to $D7FF which is always mapped to the hardware I/O chips (GTIA, POKEY, PIA, ANTIC).
1	0	BASIC ENABLED
	1	BASIC DISABLED, RAM ENABLED
		The memory region mapped to BASIC is from $BFFF.
2	0	LED #1 ON
	1	LED #1 OFF
3	0	LED #2 ON
	1	LED #2 OFF
4		RESERVED FOR FUTURE USE
5		RESERVED FOR FUTURE USE
6		RESERVED FOR FUTURE USE
7	0	SELF TEST ROM ENABLED
	1	SELF TEST ROM DISABLED, RAM ENABLED
		The memory region mapped to the self test ROM is from $5000 to $57FF.

NOTE: The OS VBLANK process copies the port A joystick and paddle values into the Port B shadows. Thus, stick 0 affects both 0 and 2, stick 1 affects both 1 and 3.

REV-LEVEL DETERMINATION

To allow program products to determine which Atari Home Computer and Operating System Revision level it is operating with, the following tests are recommended:

If location $FCD8 = $A2, then product is an A400/A800 wherein:

> If location $FFF8 = $DD and $FFF9 = $57
>   then OS is NTSC rev A.
> If location $FFF8 = $D6 and $FFF9 = $57
>   then OS is PAL rev A.
> If location $FFF8 = $F3 and $FFF9 = $E6
>   then OS is NTSC rev B.
> If location $FFF8 = $22 and $FFF9 = $58
>   then OS is PAL rev B.
> Otherwise, it is some future A400/A800 OS.

If location $FCD8 not $A2, then product is a 1200XL or other future home computer product, wherein:

> If location $FFF1 = $01, then OS is 1200XL
>   and location $FFF7 will be the
>   internal rev number for the 1200XL OS.
>
> Otherwise, location $FFF1 = product code for
>   future Atari Home Computer product
>   and location $FFF7 contains OS rev
>   level for this product.

APPENDIX A — AN EXAMPLE OF KEYBOARD REASSIGNMENT

As suggested earlier in this document, the keyboard functions may be reassigned. The table below gives the corresponding keys for the Dvorak (also known as the American Simplified) Keyboard. When the typewriter was first invented in 1867, Christopher L. Sholes chose a layout for the keys which would slow down the good typists of his day and thereby prevent his machine from jamming. This keyboard has endured to this day.

In 1932, August Dvorak invented this key layout which places the most often used characters, including the vowels, on the "home" key line and also redistributes the keystrokes from a 60-70% left-hand activity to an almost 50/50 activity. Certain manufacturers currently offer this key layout as an option. Now you can try it for yourself if you wish. Only the list of key correspondence is given here. It is left to the reader to compose the key function table using the data contained earlier in this manual.

TOP ROW OF KEYBOARD		CENTER ROW		BOTTOM ROW	
Current	Dvorak	Current	Dvorak	Current	Dvorak
Q	?	A	A	Z	'
q	/			z	;
W	,	S	O	X	Q
w	,				
E	.	D	E	C	J
e	.				
R	P	F	U	V	K
T	Y	G	I	B	X
Y	F	H	D	N	B
U	G	J	H	M	M
I	C	K	T	"	W
				"	w
O	R	L	N	.	V
				.	v
P	L	:	S	?	Z
		;	s	/	z
¼	"	"	_(underline)		
½	'	'	—		

APPENDIX B — SUGGESTIONS FOR THE CONSTRUCTION OF A NEW CHARACTER SET FOR THE NEW GRAPHICS MODES

This appendix covers the new graphics modes 12, 13, 14 and 15 now provided on the 1200XL. Modes 14 and 15 are pure graphics modes with resolutions of 160 by 20 and 160 by 40 respectively. Since these are not character modes, the discussion below will be limited only to modes 12 and 13.

Graphics 12 and 13 do not produce recognizable characters, for the most part using the standard character set. One will understand why this is true by examining the following comparison between Graphics mode 0 to 12 and 13.

> Mode 0 is a 40 character mode. Each character is formed from an 8 wide by 8 high pixel matrix. Each pixel is one bit wide in memory and is ½ of a color clock wide on the screen.
>
> Modes 12 and 13 are also 40 character modes. However, each character is formed from a 4 wide by 8 high pixel matrix, with each pixel 2 bits wide in memory and one color clock wide on the screen. This forces the character to be the same width as that used in Graphics mode 0, but cannot convey the same information within 4 pixels as with 8 as far as character recognition is concerned. (It is difficult to form a recognizable character in a four by eight dot matrix).

Let's examine how the 4-pixel character is formed, again comparing the way the 8-pixel character is formed in mode 0:

> Mode 0 has a choice of two colors for each pixel (the hardware manual says 1 ½ colors, but it is actually either the hue and luminance of playfield 2 if there is a zero bit in the selected pixel position, or the hue from playfield 2 with the luminance of playfield 1 if there is a 1 bit in the selected pixel position. Therefore, each single bit in the character definition byte for a given line occupies a single ½-color-clock-wide pixel position. The character set built into the OS defines the characters in an 8 by 8 matrix.
>
> Mode 12 also uses 8 scan lines per character. However, it uses the character bytes in a different manner. Each of the character bytes retrieved by the ANTIC is treated as a set of four two-bit quantities, where each bit pair describes the color which is to be applied to one of the 4 single color-clock-wide pixels which are part of the character. Mode 13 is the same in its treatment of the data bytes, but each of the characters is double-height (16 scan lines instead of 8) and each data byte is used twice which effectively doubles the height of the character.

Let's look at a typical character, for example a W. The bits which form a W in the default character set are similar to the following:

```
1  0  0  0  0  0  0  1          display:
1  0  0  0  0  0  0  1
1  0  0  1  1  0  0  1
1  0  0  1  1  0  0  1
1  0  1  0  0  1  0  1
1  1  0  0  0  0  1  1
1  1  0  0  0  0  1  1
1  0  0  0  0  0  0  1
```

(NOTE: This is not the exact representation, but is used as an example of correct interpretation in mode 0 and incorrect interpretation in modes 12 and 13.)

If you view the sample set of bytes, each at consecutive addresses within the defined character set, it actually looks like a W when you trace the outline formed by the 1's in the byte set, as shown in the display example to the right of the byte representation.

In this mode 0 display, each of the 1's would be one color, and each of the zeros would be another color, assuring a readable display.

For the modes 12 and 13, the four (not 8) pixels are controlled as follows:

If two-bit value is:	Then the pixel color is:
00	the background color
01	the playfield 0 color
10	the playfield 1 color
11	the playfield 2 color (if bit 7 of char = 0)
11	the playfield 3 color (if bit 7 of char = 1)

For the example shown, then, the 4th line from the bottom would display a 10 10 01 01 or 4 pixels of playfield colors 1, 1, 0, 0 in a row, if the standard character set is used. And the bottom-most line would display playfield colors 1, BAK, BAK, 0 in a row. As may be imagined, it is difficult to recognize such a character. (This character is a mirror image left to right — nonsymmetric characters would be even more difficult to recognize.)

To build a character set for these modes 12 and 13, then, it is suggested that you build each character as double wide, to allow a total of 8 pixels (by 8 lines) to define the character. This would also mean assigning two character set locations for each character and treating each character printed in these modes as two characters to be printed. For the example of the W, the character set might look like this:

Byte set 1:

10	00	00	00
10	00	00	00
10	00	00	10
10	00	00	10
10	00	10	00
10	10	00	00
10	10	00	00
10	00	00	00

Byte set 2:

00	00	00	10
00	00	00	10
10	00	00	10
10	00	00	10
00	10	00	10
00	00	10	10
00	00	10	10
00	00	00	10

Byte set 1 may represent ATASCII value hex 57 within the new character set table, and set 2 may be at ATASCII value hex D7 (hex 57 plus hex 80) if desired. You may feel free, of course, to assign your character sets in any manner you desire.

Therefore, if you would print these two characters side by side on the screen, it would become effectively a 20 character per line mode, with the resultant 10-combination treated as the 1-bit in the mode 0 example and the 00-combination as the 0-bit in the mode 0 example, forming a recognizable W in the process.

Note also that you may want to design these new character sets in a 7 by 7 matrix starting the upper left hand corner of the bit-pair set to allow at least one blank row and column between each of the new characters. (This was not done in the example.)

Thus many combinations of colorful characters may be formed using this technique, allowing the user of the 1200XL additional program flexibility.

MEMORY REQUIREMENTS FOR NEW SCREEN MODES

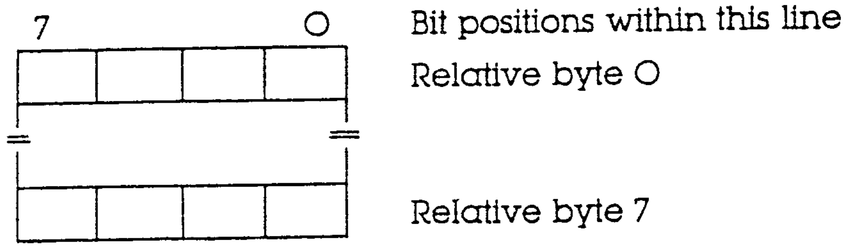
The following table summarizes the memory utilization for the new modes 12 through 15.

Mode No.	Horiz. Posit.	Vert. w/o split screen	Vert. with split screen	Colors	Data Value range	Color Reg. used	Memory Required (split screen)	Memory Required (full screen)
12	40	24	20	5	00-7F	*	1154	1152
13	40	12	10	5	00-7F	*	664	660
14	160	192	160	2	0 1	BAK PF0	4270	4296
15	160	192	160	4	0 1 2 3	BAK PF0 PF1 PF2	8112	8138

*Note: See character definition format for modes 12 and 13.

CHARACTER DEFINITION FORMAT FOR MODES 12 AND 13

The following chart shows the layout for a single character of the character set which would be used for forming characters in modes 12 and 13. As explained above, the value of each of the bit pairs specifies what color will appear on a full-width color clock when this character mode is selected.



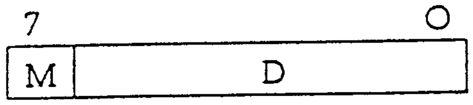
Each 2 bit color specification in the character definition maps to the color registers as follows:

If the bits have the value:	**Then the color register used to select the color of the pixel is:**
O	BAK
1	PFO
2	PF1
3	PF2 if Bit-7 (the color modifier) equals a O, or PF3 if Bit-7 (the color modifier) equals a 1.

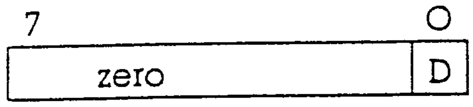
The meaning of the color modifier is shown in the following tables, which show the formats for the data bytes which are used to produce the display itself. As a reminder, the data which is to be displayed on the screen is located somewhere in memory. The data is located by the address provided in the display list. The data bytes themselves in these locations will be interpreted according to the following table.

TABLE of DATA FORMATS used for GET CHARACTER/PUT CHARACTER for MODES 12 through 15.

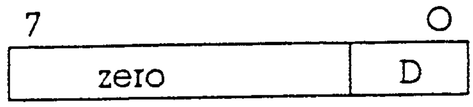
Modes 12, 13 — M = color modifier bit; D = truncated ATASCII



Mode 14 — D = color



Mode 15 — D = color



APPENDIX C — DATA BASE CHANGES FROM REV. B TO 1200

This appendix lists the difference in memory usage between the Rev. B operating system of the A400/800 and the operating system for the 1200XL.

LOCATION	REV. B USE	1200XL USE		
0000	reserved	LNFLG	—	for inhouse debugger
0001	reserved	NGFLAG	—	for power-up self test
001C	PTIMOT moved (0314)	ABUFPT	—	reserved
001D	PBPNT moved (02DE)	ABUFPT	—	reserved
001E	PBUFSZ moved (02DF)	ABUFPT	—	reserved
001F	PTEMP (deleted)	ABUFPT	—	reserved
0036	CRETRY moved (029C)	LTEMP	—	loader temp.
0037	DRETRY moved (02BD)	LTEMP	—	loader temp.
004A	CKEY moved (03E9)	ZCHAIN	—	handler loader temp.
004B	CASSBT moved (03E9)	ZCHAIN	—	handler loader temp.
0060	NEWROW moved (02F5)	FKDEF	—	func. key def. ptr.
0061	NEWCOL moved (02F6)	FKDEF	—	func. key def. ptr.
0062	NEWCOL moved (02F7)	PALNTS	—	PAL/NTSC flag.
0079	ROWINC moved (02F8)	KEYDEF	—	key def. pointer
007A	COLINC moved (02F9	KEYDEF	—	key def. pointer
0233	reserved	LCOUNT	—	loader temp.
0238-0239	reserved	RELADR	—	loader
0245	reserved	RECLEN	—	loader
0247	LINBUF (deleted)	reserved		
0248-026A	LINBUF (deleted)	reserved		
026B	LINBUF (deleted)	CHSALT	—	character set ptr.
026C	LINBUF (deleted)	VSFLAG	—	fine scroll temp.
026D	LINBUF (deleted)	KEYDIS	—	keyboard disable
026E	LINBUF (deleted)	FINE	—	fine scrolling flag
0288	CSTAT (deleted)	HIBYTE	—	loader
028E	reserved	NEWADR	—	loader
029C	TMPX1 (deleted)	CRETRY	—	from 0036
02BD	HOLD5 (deleted)	DRETRY	—	from0037
02C9-02CA	reserved	RUNADR	—	loader
02CB-02CC	reserved	HIUSED	—	loader
02CD-02CE	reserved	ZHIUSE	—	loader
02CF-02D0	reserved	GBYTEA	—	loader
02D1-02D2	reserved	LOADAD	—	loader
02D3-02D4	reserved	ZLOADA	—	loader
02D5-02D6	reserved	DSCTLN	—	disk sector size
02D7-02D8	reserved	ACMISR	—	reserved
02D9	reserved	KRPDEL	—	auto key delay
02DA	reserved	KEYREP	—	auto key rate
02DB	reserved	NOCLIK	—	key click disable
02DC	reserved	HELPFG	—	HELP key flag
02DD	reserved	DMASAV	—	DMA state save
02DE	reserved	PBPNT	—	from 001D
02DF	reserved	PBUFSZ	—	from 001E

O2E9	reserved
O2F5	reserved
O2F6-O2F7	reserved
O2F8	reserved
O2F9	reserved
O3OE	ADDCOR (deleted)
O314	TEMP2 moved (O313)
O33D	reserved
O33E	reserved
O33F	reserved
O3E8	reserved
O3E9	reserved
O3EA	reserved
O3EB	reserved
O3ED-O3F8	reserved
O3F9	reserved
O3FA	reserved
O3FB-O3FC	reserved

HNDLOD	—	handler loader flag
NEWROW	—	from OO6O
NEWCOL	—	from OO61
ROWINC	—	from OO79
COLINC	—	from OO7A
JMPERS	—	option jumpers
PTIMOT	—	from OO1C
PUPBT1	—	power-up/reset
PUPBT2	—	power-up/reset
PUPBT3	—	power-up/reset
SUPERF	—	screen editor
CKEY	—	from OO4A
CASSBT	—	from OO4B
CARTCK	—	cart checksum
ACMVAR	—	reserved
MINTLK	—	reserved
GINTLK	—	cart interlock
CHLINK	—	handler chain

247-26A = free ram
cB65-cBFF = free os

```
0000                          LIST    -F,-M


                   ***        Copyright 1984 ATARI.  Unauthorized reproduction, a:
                   *          distribution, performance or display of this comput:
                   *          or the associated audiovisual work is strictly proh:



                   ***        OS - Operating System
                   *
                   *          NOTES
                   *                   This represents an attempt to bring the OS :
                   *                   into conformance with the Atari Internal So:
                   *                   Standards as defined in the Software Develo:
                   *                   Committee Report on Procedures and Standard:
                   *                   (10/27/81).  Due to time constraints, the e:
                   *                   source could not be brought up to the stand:
                   *                   particularly in the area of subroutine head:
                   *                   documentation (ENTRY, EXIT, CHANGES and CAL:
                   *                   More complete and consistent conformance to:
                   *                   standard is planned for the next revision o:
                   *                   Operating System (Revision 3).
                   *
                   *          MODS
                   *                   Revision A (400/800)
                   *                   D. Crane/A. Miller/L. Kaplan/R. Whitehead
                   *
                   *                   Revision B (400/800)
                   *                   Fix several problems.
                   *                   M. Mahar/R. S. Scheiman
                   *
                   *                   Revision 10 (1200XL)
                   *                   Support 1200XL, add new features.
                   *                   H. Stewart/L. Winner/R. S. Scheiman/
                   *                   Y. M. Chen/M. W. Colburn       10/26/82
                   *
                   *                   Revision 11 (1200XL)
                   *                   Fix several problems.
                   *                   R. S. Scheiman  12/23/82
                   *
                   *                   Revision 1 (600XL/800XL)        ~~C061598B~~  C062024
                   *                   Support PBI and on-board BASIC.
                   *                   R. S. Scheiman/R. K. Nordin/Y. M. Chen  03/:
                   *
                   *                   Revision 2 (600XL/800XL)  C061598B
                   *                   Fix several problems.
                   *                   R. S. Scheiman  05/10/83
                   *                   Bring closer to Coding Standard (object unc:
                   *                   R. K. Nordin    11/01/83
```

```
**      Program Structure
*
*       The sections of the OS appear in the following orde:
*       corresponding subtitles:
*
*       Equates and Definitions
*
*               System Symbol Equates
*               System Address Equates
*               Miscellaneous Address Equates
*               Macro Definitions
*
*       Code and Data
*
*               First 8K ROM Identification and Checksum
*
*               Interrupt Handler
*               Initialization
*               Disk Input/Ouput
*               Relocating Loader
*               Self-test, Part 1
*               Parallel Input/Output
*               Peripheral Handler Loading Facility, Part 1
*               Self-test, Part 2
*               Peripheral Handler Loading Facility, Part 2
*
*               International Character Set
*
*               Self-test, Part 3
*               Floating Point Package
*
*               Domestic Character Set
*
*               Device Handler Vector Tables
*               Jump vectors
*               Generic Parallel Device Handler Vector Tabl:
*
*               $E4C0 Patch
*               Central Input/Output
*               Peripheral Handler Loading Facility, Part 3
*               $E912 Patch
*               Peripheral Handler Loading Facility, Part 4
*               $E959 Patch
*               Serial Input/Output
*               Keyboard, Editor and Screen Handler, Part 1
*               Peripheral Handler Loading Facility, Part 5
*               $EF6B Patch
*               Keyboard, Editor and Screen Handler, Part 2
*               $F223 Patch
*               Keyboard, Editor and Screen Handler, Part 3
*               $FCD8 Patch
*               Cassette Handler
*               Printer Handler
*               Self-test, Part 4
*
*               Second 8K ROM Identification and Checksum
*               6502 Machine Vectors
```

```
0000                 **      Assembly Option Equates


      = 0000         FALSE   EQU      0
      = FFFF         TRUE    EQU      not FALSE

      = FFFF         VGC     SET      TRUE     ;virtual game controllers
      = 0000         RAMSYS  SET      FALSE    ;not RAM based system
      = 0000         LNBUG   SET      FALSE    ;no LNBUG interface
      = 0000         ACM1    SET      FALSE    ;no asynchronous communications mod:



                     **      Identification Equates


      = 0002         IDREV   EQU      $02      ;identification revision number
      = 0010         IDDAY   EQU      $10      ;identification day
      = 0005         IDMON   EQU      $05      ;identification month
      = 0083         IDYEAR  EQU      $83      ;identification year
      = 0002         IDCPU   EQU      $02      ;identification CPU series
      = 0042         IDPN1   EQU      'B'      ;identification part number field 1
      = 0042         IDPN2   EQU      'B'      ;identification part number field 2
      = 0000         IDPN3   EQU      $00      ;identification part number field 3
      = 0000         IDPN4   EQU      $00      ;identification part number field 4
      = 0001         IDPN5   EQU      $01      ;identification part number field 5



                     **      Configuration Equates
                     *
                     *       NOTES
                     *                Problem: last byte of HATABS (as defined by:
                     *                overlaps first power-up validation byte.


      = 0021         MAXDEV  EQU      33       ;offset to last possible entry of H:
      = 0010         IOCBSZ  EQU      16       ;length of IOCB

      = 0000         SEIOCB  EQU      0*IOCBSZ          ;screen editor IOCB index
      = 0080         MAXIOC  EQU      8*IOCBSZ          ;first invalid IOCB index

      = 0080         DSCTSZ  EQU      128      ;disk sector size

      = 0002         LEDGE   EQU      2        ;left edge
      = 0027         REDGE   EQU      39       ;right edge

      = 0700         INIML   EQU      $0700    ;initial MEMLO

      = CC00         ICSORG  EQU      $CC00    ;international character set origin
      = E000         DCSORG  EQU      $E000    ;domestic character set origin
```

```
                **      IOCB Command Code Equates


  = 0003        OPEN    EQU     $03     ;open
  = 0005        GETREC  EQU     $05     ;get record
  = 0007        GETCHR  EQU     $07     ;get character(s)
  = 0009        PUTREC  EQU     $09     ;put record
  = 000B        PUTCHR  EQU     $0B     ;put character(s)
  = 000C        CLOSE   EQU     $0C     ;close
  = 000D        STATIS  EQU     $0D     ;status
  = 000E        SPECIL  EQU     $0E     ;special



                **      Special Entry Command Equates

                ;       Screen Commands

  = 0011        DRAWLN  EQU     $11     ;draw line
  = 0012        FILLIN  EQU     $12     ;draw line with right fill



                **      ICAX1 Auxiliary Byte 1 Equates


  = 0001        APPEND  EQU     $01     ;open write append (D:) or screen r:
  = 0002        DIRECT  EQU     $02     ;open for directory access (D:)
  = 0004        OPNIN   EQU     $04     ;open for input (all devices)
  = 0008        OPNOT   EQU     $08     ;open for output (all devices)
  = 0010        MXDMOD  EQU     $10     ;open for mixed mode (E:, S:)
  = 0020        INSCLR  EQU     $20     ;open for input without clearing sc:



                **      Device Code Equates


  = 0043        CASSET  EQU     'C'     ;cassette
  = 0044        DISK    EQU     'D'     ;disk
  = 0045        SCREDT  EQU     'E'     ;screen editor
  = 004B        KBD     EQU     'K'     ;keyboard
  = 0050        PRINTR  EQU     'P'     ;printer
  = 0053        DISPLY  EQU     'S'     ;screen display
```

```
                **      Character and Key Code Equates


    = 007D      CLS     EQU     $7D     ;clear screen
    = 009B      EOL     EQU     $9B     ;end of line (RETURN)

    = 0011      HELP    EQU     $11     ;key code for HELP
    = 0083      CNTLF1  EQU     $83     ;key code for CTRL-F1
    = 0084      CNTLF2  EQU     $84     ;key code for CTRL-F2
    = 0093      CNTLF3  EQU     $93     ;key code for CTRL-F3
    = 0094      CNTLF4  EQU     $94     ;key code for CTRL-F4
    = 009F      CNTL1   EQU     $9F     ;key code for CTRL-1



                **      Status Code Equates


    = 0001      SUCCES  EQU     1       ;successful operation

    = 0080      BRKABT  EQU     128     ;BREAK key abort
    = 0081      PRVOPN  EQU     129     ;IOCB already open error
    = 0082      NONDEV  EQU     130     ;nonexistent device error
    = 0083      WRONLY  EQU     131     ;IOCB opened for write only error
    = 0084      NVALID  EQU     132     ;invalid command error
    = 0085      NOTOPN  EQU     133     ;device/file not open error
    = 0086      BADIOC  EQU     134     ;invalid IOCB index error
    = 0087      RDONLY  EQU     135     ;IOCB opened for read only error
    = 0088      EOFERR  EQU     136     ;end of file error
    = 0089      TRNRCD  EQU     137     ;truncated record error
    = 008A      TIMOUT  EQU     138     ;peripheral device timeout error
    = 008B      DNACK   EQU     139     ;device does not acknowledge comman:
    = 008C      FRMERR  EQU     140     ;serial bus framing error
    = 008D      CRSROR  EQU     141     ;cursor overrange error
    = 008E      OVRRUN  EQU     142     ;serial bus data overrun error
    = 008F      CHKERR  EQU     143     ;serial bus checksum error
    = 0090      DERROR  EQU     144     ;device done (operation incomplete):
    = 0091      BADMOD  EQU     145     ;bad screen mode number error
    = 0092      FNCNOT  EQU     146     ;function not implemented in handle:
    = 0093      SCRMEM  EQU     147     ;insufficient memory for screen mod:

                **      DCB Device Bus ID Equates


    = 0031      DISKID  EQU     $31     ;disk bus ID
    = 0040      PDEVN   EQU     $40     ;printer bus ID
    = 0060      CASET   EQU     $60     ;cassette bus ID
```

```
                    **      Bus Command Equates


      = 0021        FOMAT   EQU     'I'     ;format command
      = 0050        PUTSEC  EQU     'P'     ;put sector command
      = 0052        READ    EQU     'R'     ;read command
      = 0053        STATC   EQU     'S'     ;status command
      = 0057        WRITE   EQU     'W'     ;write command



                    **      Command Auxiliary Byte Equates


      = 0044        DOUBLE  EQU     'D'     ;print 20 characters double width
      = 004E        NORMAL  EQU     'N'     ;print 40 characters normally
      = 0050        PLOT    EQU     'P'     ;plot
      = 0053        SIDWAY  EQU     'S'     ;print 16 characters sideways



                    **      Bus Response Equates


      = 0041        ACK     EQU     'A'     ;device acknowledged
      = 0043        COMPLT  EQU     'C'     ;device successfully completed oper:
      = 0045        ERROR   EQU     'E'     ;device incurred error in attempted:
      = 004E        NACK    EQU     'N'     ;device did not understand



                    **      Floating Point Package Miscellaneous Equates


      = 0006        FPREC   EQU     6       ;precision

      = 0005        FMPREC  EQU     FPREC-1 ;length of mantissa



                    **      Cassette Record Type Equates


      = 00FB        HDR     EQU     $FB     ;header
      = 00FC        DTA     EQU     $FC     ;data record
      = 00FA        DT1     EQU     $FA     ;last data record
      = 00FE        EOT     EQU     $FE     ;end of tape (file)

      = 0002        TONE1   EQU     2       ;record
      = 0001        TONE2   EQU     1       ;playback
```

```
                        **      Cassette Timing Equates


       = 0480           WLEADN  EQU     1152    ;NTSC 19.2 second WRITE file leader
       = 0240           RLEADN  EQU     576     ;NTSC 9.6 second READ file leader
       = 00B4           WIRGLN  EQU     180     ;NTSC 3.0 second WRITE IRG
       = 0078           RIRGLN  EQU     120     ;NTSC 2.0 second READ IRG
       = 000F           WSIRGN  EQU     15      ;NTSC 0.25 second WRITE short IRG
       = 000A           RSIRGN  EQU     10      ;NTSC 0.16 second READ short IRG
       = 001E           BEEPNN  EQU     30      ;NTSC 0.5 second beep duration
       = 000A           BEEPFN  EQU     10      ;NTSC 0.16 second beep separation

       = 03C0           WLEADP  EQU     960     ;PAL 19.2 second WRITE file leader
       = 01E0           RLEADP  EQU     480     ;PAL 9.6 second READ file leader
       = 0096           WIRGLP  EQU     150     ;PAL 3.0 second WRITE IRG
       = 0064           RIRGLP  EQU     100     ;PAL 2.0 second READ IRG
       = 000D           WSIRGP  EQU     13      ;PAL 0.25 second WRITE short IRG
       = 0008           RSIRGP  EQU     8       ;PAL 0.16 second READ short IRG
       = 0019           BEEPNP  EQU     25      ;PAL 0.5 second beep duration
       = 0008           BEEPFP  EQU     8       ;PAL 0.16 second beep separation

       = 0000           WIRGHI  EQU     0       ;high WRITE IRG
       = 0000           RIRGHI  EQU     0       ;high READ IRG


                        **      Power-up Validation Byte Value Equates


       = 005C           PUPVL1  EQU     $5C     ;power-up validation value 1
       = 0093           PUPVL2  EQU     $93     ;power-up validation value 2
       = 0025           PUPVL3  EQU     $25     ;power-up validation value 3


                        **      Relocating Loader Miscellaneous Equates


       = 009C           DATAER  EQU     156     ;end of record appears before END r:
       = 009D           MEMERR  EQU     157     ;memory insufficient for load error
```

```
                  **       Miscellaneous Equates


   = 00FF         IOCFRE  EQU     $FF       ;IOCB free indicator

   = 0028         B19200  EQU     $0028     ;19200 baud POKEY counter value
   = 05CC         B00600  EQU     $05CC     ;600 baud POKEY counter value

   = 0005         HITONE  EQU     $05       ;FSK high freq. POKEY counter value;
   = 0007         LOTONE  EQU     $07       ;FSK low freq. POKEY counter value ;

   = 0034         NCOMLO  EQU     $34       ;PIA lower NOT COMMAND line command
   = 003C         NCOMHI  EQU     $3C       ;PIA raise NOT COMMAND line command

   = 0034         MOTRGO  EQU     $34       ;PIA cassette motor ON command
   = 003C         MOTRST  EQU     $3C       ;PIA cassette motor OFF command

   = 0000         NODAT   EQU     $00       ;SIO immediate operation
   = 0040         GETDAT  EQU     $40       ;SIO read data frame
   = 0080         PUTDAT  EQU     $80       ;SIO write data frame

   = 000D         CRETRI  EQU     13        ;number of command frame retries
   = 0001         DRETRI  EQU     1         ;number of device retries
   = 0002         CTIM    EQU     2         ;command frame ACK timeout

   = 0028         NBUFSZ  EQU     40        ;print normal buffer size
   = 0014         DBUFSZ  EQU     20        ;print double buffer size
   = 001D         SBUFSZ  EQU     29        ;print sideways buffer size
```

```
0000                **       Page Zero Address Equates

       = 0000       LNFLG    EQU      $0000    ;1-byte LNBUG flag (0 = not LNBUG)
       = 0001       NGFLAG   EQU      $0001    ;1-byte memory status (0 = failure)

                    ;        Not Cleared

       = 0002       CASINI   EQU      $0002    ;2-byte cassette program initializa:
       = 0004       RAMLO    EQU      $0004    ;2-byte RAM address for memory test
       = 0006       TRAMSZ   EQU      $0006    ;1-byte RAM size temporary
       = 0007       CMCMD    EQU      $0007    ;1-byte command communications

                    ;        Cleared upon Coldstart Only

       = 0008       WARMST   EQU      $0008    ;1-byte warmstart flag (0 = coldsta:
       = 0009       BOOT?    EQU      $0009    ;1-byte successful boot flags
       = 000A       DOSVEC   EQU      $000A    ;2-byte disk program start vector
       = 000C       DOSINI   EQU      $000C    ;2-byte disk program initialization:
       = 000E       APPMHI   EQU      $000E    ;2-byte applications memory high li:

                    ;        Cleared upon Coldstart or Warmstart

       = 0010       INTZBS   EQU      $0010    ;first page zero location to clear

       = 0010       POKMSK   EQU      $0010    ;1-byte IRQEN shadow
       = 0011       BRKKEY   EQU      $0011    ;1-byte BREAK key flag (0 = no BREA:
       = 0012       RTCLOK   EQU      $0012    ;3-byte real time clock (16 millise:
       = 0015       BUFADR   EQU      $0015    ;2-byte disk interface buffer addre:
       = 0017       ICCOMT   EQU      $0017    ;1-byte CIO command table index
       = 0018       DSKFMS   EQU      $0018    ;2-byte DOS File Management System :
       = 001A       DSKUTL   EQU      $001A    ;2-byte DOS utility pointer
       = 001C       ABUFPT   EQU      $001C    ;4-byte ACMI buffer pointer area

       = 0020       ZIOCB    EQU      $0020    ;address of page zero IOCB
       = 0020       IOCBAS   EQU      $0020    ;16-byte page zero IOCB
       = 0020       ICHIDZ   EQU      $0020    ;1-byte handler ID ($FF = IOCB free:
       = 0021       ICDNOZ   EQU      $0021    ;1-byte device number
       = 0022       ICCOMZ   EQU      $0022    ;1-byte command code
       = 0023       ICSTAZ   EQU      $0023    ;1-byte status of last action
       = 0024       ICBALZ   EQU      $0024    ;1-byte low buffer address
       = 0025       ICBAHZ   EQU      $0025    ;1-byte high buffer address
       = 0026       ICPTLZ   EQU      $0026    ;1-byte low PUT-BYTE routine addres:
       = 0027       ICPTHZ   EQU      $0027    ;1-byte high PUT-BYTE routine addre:
       = 0028       ICBLLZ   EQU      $0028    ;1-byte low buffer length
       = 0029       ICBLHZ   EQU      $0029    ;1-byte high buffer length
       = 002A       ICAX1Z   EQU      $002A    ;1-byte first auxiliary information
       = 002B       ICAX2Z   EQU      $002B    ;1-byte second auxiliary informatio:
       = 002C       ICSPRZ   EQU      $002C    ;4-byte spares

       = 002C       ENTVEC   EQU      $002C    ;2-byte (not used)
       = 002E       ICIDNO   EQU      $002E    ;1-byte IOCB index (IOCB number tim:
       = 002F       CIOCHR   EQU      $002F    ;1-byte character for current CIO o:

       = 0030       STATUS   EQU      $0030    ;1-byte SIO operation status
       = 0031       CHKSUM   EQU      $0031    ;1-byte checksum (single byte sum w:
       = 0032       BUFRLO   EQU      $0032    ;1-byte low data buffer address
= 0033      BUFRHI  EQU     $0033   ;1-byte high data buffer address
= 0034      BFENLO  EQU     $0034   ;1-byte low data buffer end address
= 0035      BFENHI  EQU     $0035   ;1-byte high data buffer end addres:
= 0036      LTEMP   EQU     $0036   ;2-byte relocating loader temporary
= 0038      BUFRFL  EQU     $0038   ;1-byte data buffer full flag (0 = :
= 0039      RECVDN  EQU     $0039   ;1-byte receive-frame done flag (0 :
= 003A      XMTDON  EQU     $003A   ;1-byte transmit-frame done flag (0:
= 003B      CHKSNT  EQU     $003B   ;1-byte checksum sent flag (0 = not:
= 003C      NOCKSM  EQU     $003C   ;1-byte no checksum follows data fl:
= 003D      BPTR    EQU     $003D   ;1-byte cassette buffer pointer
= 003E      FTYPE   EQU     $003E   ;1-byte cassette IRG type (neg. = c:
= 003F      FEOF    EQU     $003F   ;1-byte cassette EOF flag (0 = no E:
= 0040      FREQ    EQU     $0040   ;1-byte cassette beep counter
= 0041      SOUNDR  EQU     $0041   ;1-byte noisy I/O flag (0 = quiet)

= 0042      CRITIC  EQU     $0042   ;1-byte critical section flag (0 = :

= 0043      FMSZPG  EQU     $0043   ;7-byte reserved for DOS File Manag:

= 004A      ZCHAIN  EQU     $004A   ;2-byte handler linkage chain point:
= 004C      DSTAT   EQU     $004C   ;1-byte display status
= 004D      ATRACT  EQU     $004D   ;1-byte attract-mode timer and flag
= 004E      DRKMSK  EQU     $004E   ;1-byte attract-mode dark (luminanc:
= 004F      COLRSH  EQU     $004F   ;1-byte attract-mode color shift
= 0050      TMPCHR  EQU     $0050   ;1-byte temporary character
= 0051      HOLD1   EQU     $0051   ;1-byte temporary
= 0052      LMARGN  EQU *   $0052   ;1-byte text column left margin
= 0053      RMARGN  EQU *   $0053   ;1-byte text column right margin
= 0054      ROWCRS  EQU *   $0054   ;1-byte cursor row                   '_' for
= 0055      COLCRS  EQU *   $0055   ;2-byte cursor column                  (m
= 0057      DINDEX  EQU *   $0057   ;1-byte display mode                    a
= 0058      SAVMSC  EQU *   $0058   ;2-byte saved memory scan counter - leave incase ML u
= 005A      OLDROW  EQU *   $005A   ;1-byte prior row
= 005B      OLDCOL  EQU *   $005B   ;2-byte prior column
= 005D      OLDCHR  EQU *   $005D   ;1-byte saved character under curso:
= 005E      OLDADR  EQU *   $005E   ;2-byte saved cursor memory address
= 0060    - FKDEF   EQU *   $0060   ;2-byte function key definition tab:
= 0062    - PALNTS  EQU     $0062   ;1-byte PAL/NTSC indicator (0 = NTS:
= 0063      LOGCOL  EQU *   $0063   ;1-byte logical line cursor column
= 0064      ADRESS  EQU *   $0064   ;2-byte temporary address

= 0066      MLTTMP  EQU     $0066   ;1-byte temporary
= 0066      OPNTMP  EQU     $0066   ;1-byte open temporary
= 0066      TOADR   EQU     $0066   ;2-byte destination address    temps

= 0068      SAVADR  EQU     $0068   ;2-byte saved address
= 0068      FRMADR  EQU     $0068   ;2-byte source address

= 006A    - RAMTOP  EQU     $006A   ;1-byte RAM size
= 006B      BUFCNT  EQU *   $006B   ;1-byte buffer count (logical line :
= 006C      BUFSTR  EQU *   $006C   ;2-byte buffer start pointer
= 006E      BITMSK  EQU *   $006E   ;1-byte bit mask for bit map operat:
= 006F      SHFAMT  EQU *   $006F   ;1-byte shift amount for pixel just:
= 0070      ROWAC   EQU *   $0070   ;2-byte draw working row
= 0072      COLAC   EQU *   $0072   ;2-byte draw working column
= 0074      ENDPT   EQU *   $0074   ;2-byte end point
= 0076      DELTAR  EQU *   $0076   ;1-byte row difference
```

```
    = 0077          DELTAC  EQU     $0077   ;2-byte column difference
    = 0079          KEYDEF  EQU     $0079   ;2-byte key definition table addres:
    = 007B          SWPFLG  EQU     $007B   ;1-byte split screen swap flag (0 =:
    = 007C          HOLDCH  EQU     $007C   ;1-byte temporary character
    = 007D          INSDAT  EQU     $007D   ;1-byte temporary
    = 007E          COUNTR  EQU     $007E   ;2-byte draw iteration count

                    ;       Reserved for Application and Floating Point Package

                    ;       EQU     $0080   ;128 bytes reserved for application:



                    **      Floating Point Package Page Zero Address Equates

    = 00D4          FR0     EQU     $00D4   ;6-byte register 0
    = 00D5          FR0M    EQU     $00D5   ;5-byte register 0 mantissa
    = 00D9          QTEMP   EQU     $00D9   ;1-byte temporary

    = 00DA          FRE     EQU     $00DA   ;6-byte (internal) register E

    = 00E0          FR1     EQU     $00E0   ;6-byte register 1
    = 00E1          FR1M    EQU     $00E1   ;5-byte register 1 mantissa

    = 00E6          FR2     EQU     $00E6   ;6-byte (internal) register 2

    = 00EC          FRX     EQU     $00EC   ;1-byte temporary

    = 00ED          EEXP    EQU     $00ED   ;1-byte value of exponent

    = 00EE          FRSIGN  EQU     $00EE   ;1-byte floating point sign
    = 00EE          NSIGN   EQU     $00EE   ;1-byte sign of number

    = 00EF          PLYCNT  EQU     $00EF   ;1-byte polynomial degree
    = 00EF          ESIGN   EQU     $00EF   ;1-byte sign of exponent

    = 00F0          SGNFLG  EQU     $00F0   ;1-byte sign flag
    = 00F0          FCHFLG  EQU     $00F0   ;1-byte first character flag

    = 00F1          XFMFLG  EQU     $00F1   ;1-byte transform flag
    = 00F1          DIGRT   EQU     $00F1   ;1-byte number of digits after deci:

    = 00F2          CIX     EQU     $00F2   ;1-byte current input index
    = 00F3          INBUFF  EQU     $00F3   ;2-byte line input buffer

    = 00F5          ZTEMP1  EQU     $00F5   ;2-byte temporary
    = 00F7          ZTEMP4  EQU     $00F7   ;2-byte temporary
    = 00F9          ZTEMP3  EQU     $00F9   ;2-byte temporary

    = 00FC          FLPTR   EQU     $00FC   ;2-byte floating point number point:
    = 00FE          FPTR2   EQU     $00FE   ;2-byte floating point number point:
```

```
                    **      Page One (Stack) Address Equates


                    ;       EQU     $0100   ;256-byte stack



                    **      Page Two Address Equates


        = 0200      INTABS  EQU     $0200   ;42-byte interrupt handler table

        = 0200      VDSLST  EQU     $0200   ;2-byte display list NMI vector
        = 0202      VPRCED  EQU     $0202   ;2-byte serial I/O proceed line IRQ;
        = 0204      VINTER  EQU     $0204   ;2-byte serial I/O interrupt line I;
        = 0206      VBREAK  EQU     $0206   ;2-byte BRK instruction IRQ vector
        = 0208      VKEYBD  EQU     $0208   ;2-byte keyboard IRQ vector
        = 020A      VSERIN  EQU     $020A   ;2-byte serial input ready IRQ vect;
        = 020C      VSEROR  EQU     $020C   ;2-byte serial output ready IRQ vec;
        = 020E      VSEROC  EQU     $020E   ;2-byte serial output complete IRQ ;
        = 0210      VTIMR1  EQU     $0210   ;2-byte POKEY timer 1 IRQ vector
        = 0212      VTIMR2  EQU     $0212   ;2-byte POKEY timer 2 IRQ vector
        = 0214      VTIMR4  EQU     $0214   ;2-byte POKEY timer 4 IRQ vector
        = 0216      VIMIRQ  EQU     $0216   ;2-byte immediate IRQ vector
        = 0218      CDTMV1  EQU     $0218   ;2-byte countdown timer 1 value
        = 021A      CDTMV2  EQU     $021A   ;2-byte countdown timer 2 value
        = 021C      CDTMV3  EQU     $021C   ;2-byte countdown timer 3 value
        = 021E      CDTMV4  EQU     $021E   ;2-byte countdown timer 4 value
        = 0220      CDTMV5  EQU     $0220   ;2-byte countdown timer 5 value
        = 0222      VVBLKI  EQU     $0222   ;2-byte immediate VBLANK NMI vector
        = 0224      VVBLKD  EQU     $0224   ;2-byte deferred VBLANK NMI vector
        = 0226      CDTMA1  EQU     $0226   ;2-byte countdown timer 1 vector
        = 0228      CDTMA2  EQU     $0228   ;2-byte countdown timer 2 vector

        = 022A      CDTMF3  EQU     $022A   ;1-byte countdown timer 3 flag (0 =;
        = 022B      SRTIMR  EQU     $022B   ;1-byte software key repeat timer
        = 022C      CDTMF4  EQU     $022C   ;1-byte countdown timer 4 flag (0 =;
        = 022D      INTEMP  EQU     $022D   ;1-byte temporary
        = 022E      CDTMF5  EQU     $022E   ;1-byte countdown timer 5 flag (0 =;
        = 022F      SDMCTL  EQU     $022F   ;1-byte DMACTL shadow
        = 0230      SDLSTL  EQU     $0230   ;1-byte DLISTL shadow
        = 0231      SDLSTH  EQU     $0231   ;1-byte DLISTH shadow
        = 0232      SSKCTL  EQU     $0232   ;1-byte SKCTL shadow
        = 0233      LCOUNT  EQU     $0233   ;1-byte relocating loader record le;
        = 0234      LPENH   EQU     $0234   ;1-byte light pen horizontal value
        = 0235      LPENV   EQU     $0235   ;1-byte light pen vertical value
        = 0236      BRKKY   EQU     $0236   ;2-byte BREAK key vector
        = 0238      VPIRQ   EQU     $0238   ;2-byte parallel device IRQ vector
        = 023A      CDEVIC  EQU     $023A   ;1-byte command frame device ID
        = 023B      CCOMND  EQU     $023B   ;1-byte command frame command
        = 023C      CAUX1   EQU     $023C   ;1-byte command auxiliary 1
        = 023D      CAUX2   EQU     $023D   ;1-byte command auxiliary 2

        = 023E      TEMP    EQU     $023E   ;1-byte temporary

                            ASSERT  low TEMP<>$FF   ;may not be the last word o;
```

```
      = 023F      ERRFLG  EQU     $023F    ;1-byte I/O error flag (0 = no erro:

                          ASSERT  low ERRFLG<>$FF ;may not be the last word o:

      = 0240      DFLAGS  EQU     $0240    ;1-byte disk flags from sector 1
      = 0241      DBSECT  EQU     $0241    ;1-byte disk boot sector count
      = 0242      BOOTAD  EQU     $0242    ;2-byte disk boot memory address
      = 0244      COLDST  EQU     $0244    ;1-byte coldstart flag (0 = complet:
      = 0245      RECLEN  EQU     $0245    ;1-byte relocating loader record le:
      = 0246      DSKTIM  EQU     $0246    ;1-byte disk format timeout
      = 0247      PDVMSK  EQU     $0247    ;1-byte parallel device selection m:
      = 0248      SHPDVS  EQU     $0248    ;1-byte PDVS (parallel device selec:
      = 0249      PDIMSK  EQU     $0249    ;1-byte parallel device IRQ selecti:
      = 024A      RELADR  EQU     $024A    ;2-byte relocating loader relative :
      = 024C      PPTMPA  EQU     $024C    ;1-byte parallel device handler tem:
      = 024D      PPTMPX  EQU     $024D    ;1-byte parallel device handler tem:

                  ;       EQU     $024E    ;6 bytes reserved for Atari

                  ;       EQU     $0254    ;23 bytes reserved for Atari

      = 026B      CHSALT  EQU     $026B    ;1-byte character set alternate
      = 026C      VSFLAG  EQU     $026C    ;1-byte fine vertical scroll count
      = 026D      KEYDIS  EQU     $026D    ;1-byte keyboard disable
      = 026E      FINE    EQU     $026E    ;1-byte fine scrolling mode
      = 026F      GPRIOR  EQU     $026F    ;1-byte PRIOR shadow

      = 0270      PADDL0  EQU     $0270    ;1-byte potentiometer 0
      = 0271      PADDL1  EQU     $0271    ;1-byte potentiometer 1
      = 0272      PADDL2  EQU     $0272    ;1-byte potentiometer 2
      = 0273      PADDL3  EQU     $0273    ;1-byte potentiometer 3
      = 0274      PADDL4  EQU     $0274    ;1-byte potentiometer 4
      = 0275      PADDL5  EQU     $0275    ;1-byte potentiometer 5
      = 0276      PADDL6  EQU     $0276    ;1-byte potentiometer 6
      = 0277      PADDL7  EQU     $0277    ;1-byte potentiometer 7

      = 0278      STICK0  EQU     $0278    ;1-byte joystick 0
      = 0279      STICK1  EQU     $0279    ;1-byte joystick 1
      = 027A      STICK2  EQU     $027A    ;1-byte joystick 2
      = 027B      STICK3  EQU     $027B    ;1-byte joystick 3

      = 027C      PTRIG0  EQU     $027C    ;1-byte paddle trigger 0
      = 027D      PTRIG1  EQU     $027D    ;1-byte paddle trigger 1
      = 027E      PTRIG2  EQU     $027E    ;1-byte paddle trigger 2
      = 027F      PTRIG3  EQU     $027F    ;1-byte paddle trigger 3
      = 0280      PTRIG4  EQU     $0280    ;1-byte paddle trigger 4
      = 0281      PTRIG5  EQU     $0281    ;1-byte paddle trigger 5
      = 0282      PTRIG6  EQU     $0282    ;1-byte paddle trigger 6
      = 0283      PTRIG7  EQU     $0283    ;1-byte paddle trigger 7

      = 0284      STRIG0  EQU     $0284    ;1-byte joystick trigger 0
      = 0285      STRIG1  EQU     $0285    ;1-byte joystick trigger 1
      = 0286      STRIG2  EQU     $0286    ;1-byte joystick trigger 2
      = 0287      STRIG3  EQU     $0287    ;1-byte joystick trigger 3

      = 0288      HIBYTE  EQU     $0288    ;1-byte relocating loader high byte:
```

```
 = 0289         WMODE    EQU      $0289    ;1-byte cassette WRITE mode ($80 = :
 = 028A         BLIM     EQU      $028A    ;1-byte cassette buffer limit
 = 028B         IMASK    EQU      $028B    ;1-byte (not used)
 = 028C         JVECK    EQU      $028C    ;2-byte jump vector or temporary
 = 028E         NEWADR   EQU      $028E    ;2-byte relocating address
 = 0290         TXTROW   EQU +    $0290    ;1-byte split screen text cursor ro:
 = 0291         TXTCOL   EQU *    $0291    ;2-byte split screen text cursor co:
 = 0293         TINDEX   EQU *    $0293    ;1-byte split scree text mode
 = 0294         TXTMSC   EQU *    $0294    ;2-byte split screen memory scan co:
 = 0296         TXTOLD   EQU *    $0296    ;6-byte OLDROW, OLDCOL, OLDCHR, OLD:
 = 029C         CRETRY   EQU      $029C    ;1-byte number of command frame ret:
 = 029D         HOLD3    EQU      $029D    ;1-byte temporary
 = 029E         SUBTMP   EQU      $029E    ;1-byte temporary
 = 029F         HOLD2    EQU      $029F    ;1-byte (not used)
 = 02A0         DMASK    EQU      $02A0    ;1-byte display (pixel location) ma:
 = 02A1         TMPLBT   EQU      $02A1    ;1-byte (not used)          [illegible] flag.
 = 02A2         ESCFLG   EQU +    $02A2    ;1-byte escape flag ($80 = ESC dete:
 = 02A3         TABMAP   EQU *    $02A3    ;15-byte (120-bit) tab stop bit map    19 bytes
 = 02B2         LOGMAP   EQU *    $02B2    ;8-byte (32-bit) logical line bit m:
 = 02B6         INVFLG   EQU      $02B6    ;1-byte inverse video flag ($80 = i: (keyboard
 = 02B7         FILFLG   EQU      $02B7    ;1-byte right fill flag (0 = no fil:
 = 02B8         TMPROW   EQU      $02B8    ;1-byte temporary row
 = 02B9         TMPCOL   EQU      $02B9    ;2-byte temporary column
 = 02BB         SCRFLG   EQU      $02BB    ;1-byte scroll occurence flag (0 = :
 = 02BC         HOLD4    EQU      $02BC    ;1-byte temporary
 = 02BD         DRETRY   EQU      $02BD    ;1-byte number of device retries
 = 02BE         SHFLOK   EQU      $02BE    ;1-byte shift/control lock flags
 = 02BF         BOTSCR   EQU      $02BF    ;1-byte screen bottom (24 = normal,:

 = 02C0         PCOLR0   EQU      $02C0    ;1-byte player-missle 0 color/lumin:
 = 02C1         PCOLR1   EQU      $02C1    ;1-byte player-missle 1 color/lumin:
 = 02C2         PCOLR2   EQU      $02C2    ;1-byte player-missle 2 color/lumin:
 = 02C3         PCOLR3   EQU      $02C3    ;1-byte player-missle 3 color/lumin:

 = 02C4         COLOR0   EQU      $02C4    ;1-byte playfield 0 color/luminance
 = 02C5         COLOR1   EQU      $02C5    ;1-byte playfield 1 color/luminance
 = 02C6         COLOR2   EQU      $02C6    ;1-byte playfield 2 color/luminance
 = 02C7         COLOR3   EQU      $02C7    ;1-byte playfield 3 color/luminance

 = 02C8         COLOR4   EQU      $02C8    ;1-byte background color/luminance

 = 02C9         PARMBL   EQU      $02C9    ;6-byte relocating loader parameter:
 = 02C9         RUNADR   EQU      $02C9    ;2-byte run address
 = 02CB         HIUSED   EQU      $02CB    ;2-byte highest non-zero page addre:
 = 02CD         ZHIUSE   EQU      $02CD    ;2-byte highest zero page address

 = 02CF         OLDPAR   EQU      $02CF    ;6-byte relocating loader parameter:
 = 02CF         GBYTEA   EQU      $02CF    ;2-byte GET-BYTE routine address
 = 02D1         LOADAD   EQU      $02D1    ;2-byte non-zero page load address
 = 02D3         ZLOADA   EQU      $02D3    ;2-byte zero page load address

 = 02D5         DSCTLN   EQU      $02D5    ;2-byte disk sector length
 = 02D7         ACMISR   EQU      $02D7    ;2-byte ACMI interrupt service rout:
 = 02D9         KRPDEL   EQU      $02D9    ;1-byte auto-repeat delay
 = 02DA         KEYREP   EQU      $02DA    ;1-byte auto-repeat rate
 = 02DB         NOCLIK   EQU      $02DB    ;1-byte key click disable
 = 02DC         HELPFG   EQU      $02DC    ;1-byte HELP key flag (0 = no HELP)
```

```
      = 02DD        DMASAV  EQU     $02DD    ;1-byte SDMCTL save/restore
      = 02DE        PBPNT   EQU     $02DE    ;1-byte printer buffer pointer
      = 02DF        PBUFSZ  EQU     $02DF    ;1-byte printer buffer size

                    ;       EQU     $02E0    ;4 bytes reserved for DOS

      = 02E4        RAMSIZ  EQU     $02E4    ;1-byte high RAM size
      = 02E5        MEMTOP  EQU     $02E5    ;2-byte top of available user memor
      = 02E7        MEMLO   EQU     $02E7    ;2-byte bottom of available user me
      = 02E9        HNDLOD  EQU     $02E9    ;1-byte user load flag (0 = no hand
      = 02EA        DVSTAT  EQU     $02EA    ;4-byte device status buffer
      = 02EE        CBAUDL  EQU     $02EE    ;1-byte low cassette baud rate
      = 02EF        CBAUDH  EQU     $02EF    ;1-byte high cassette baud rate
      = 02F0        CRSINH  EQU     $02F0    ;1-byte cursor inhibit (0 = cursor
      = 02F1        KEYDEL  EQU     $02F1    ;1-byte key debounce delay timer
      = 02F2        CH1     EQU     $02F2    ;1-byte prior keyboard character
      = 02F3        CHACT   EQU     $02F3    ;1-byte CHACTL shadow
      = 02F4        CHBAS   EQU     $02F4    ;1-byte CHBASE shadow

      = 02F5        NEWROW  EQU     $02F5    ;1-byte draw destination row
      = 02F6        NEWCOL  EQU     $02F6    ;2-byte draw destination column
      = 02F8        ROWINC  EQU     $02F8    ;1-byte draw row increment
      = 02F9        COLINC  EQU     $02F9    ;1-byte draw column increment

      = 02FA        CHAR    EQU     $02FA    ;1-byte internal character
      = 02FB        ATACHR  EQU     $02FB    ;1-byte ATASCII character or plot p
      = 02FC        CH      EQU     $02FC    ;1-byte keyboard code (buffer)
      = 02FD        FILDAT  EQU     $02FD    ;1-byte right fill data
      = 02FE        DSPFLG  EQU     $02FE    ;1-byte control character display f
      = 02FF        SSFLAG  EQU     $02FF    ;1-byte start/stop flag (0 = not st


                    **      Page Three Address Equates


      = 0300        DCB     EQU     $0300    ;12-byte device control block
      = 0300        DDEVIC  EQU     $0300    ;1-byte unit 1 bus ID
      = 0301        DUNIT   EQU     $0301    ;1-byte unit number
      = 0302        DCOMND  EQU     $0302    ;1-byte bus command
      = 0303        DSTATS  EQU     $0303    ;1-byte command type/status return
      = 0304        DBUFLO  EQU     $0304    ;1-byte low data buffer address
      = 0305        DBUFHI  EQU     $0305    ;1-byte high data buffer address
      = 0306        DTIMLO  EQU     $0306    ;1-byte timeout (seconds)
      = 0307        DUNUSE  EQU     $0307    ;1-byte (not used)
      = 0308        DBYTLO  EQU     $0308    ;1-byte low number of bytes to tran
      = 0309        DBYTHI  EQU     $0309    ;1-byte high number of bytes to tra
      = 030A        DAUX1   EQU     $030A    ;1-byte first command auxiliary
      = 030B        DAUX2   EQU     $030B    ;1-byte second command auxiliary

      = 030C        TIMER1  EQU     $030C    ;2-byte initial baud rate timer val
      = 030E        JMPERS  EQU     $030E    ;1-byte jumper options
      = 030F        CASFLG  EQU     $030F    ;1-byte cassette I/O flag (0 = not
      = 0310        TIMER2  EQU     $0310    ;2-byte final baud rate timer value
      = 0312        TEMP1   EQU     $0312    ;2-byte temporary
      = 0313        TEMP2   EQU     $0313    ;1-byte temporary
```

```
    = 0314       PTIMOT  EQU     $0314    ;1-byte printer timeout
    = 0315       TEMP3   EQU     $0315    ;1-byte temporary
    = 0316       SAVIO   EQU     $0316    ;1-byte saved serial data input ind:
    = 0317       TIMFLG  EQU     $0317    ;1-byte timeout flag (0 = timeout)
    = 0318       STACKP  EQU     $0318    ;1-byte SIO saved stack pointer
    = 0319       TSTAT   EQU     $0319    ;1-byte temporary status

    = 031A       HATABS  EQU     $031A    ;35-byte handler address table

    = 033D       PUPBT1  EQU     $033D    ;1-byte power-up validation byte 1
    = 033E       PUPBT2  EQU     $033E    ;1-byte power-up validation byte 2
    = 033F       PUPBT3  EQU     $033F    ;1-byte power-up validation byte 3

    = 0340       IOCB    EQU     $0340    ;128-byte I/O control blocks area
    = 0340       ICHID   EQU     $0340    ;1-byte handler ID ($FF = free)
    = 0341       ICDNO   EQU     $0341    ;1-byte device number
    = 0342       ICCOM   EQU     $0342    ;1-byte command code
    = 0343       ICSTA   EQU     $0343    ;1-byte status of last action
    = 0344       ICBAL   EQU     $0344    ;1-byte low buffer address
    = 0345       ICBAH   EQU     $0345    ;1-byte high buffer address
    = 0346       ICPTL   EQU     $0346    ;1-byte low PUT-BYTE routine addres:
    = 0347       ICPTH   EQU     $0347    ;1-byte high PUT-BYTE routine addre:
    = 0348       ICBLL   EQU     $0348    ;1-byte low buffer length
    = 0349       ICBLH   EQU     $0349    ;1-byte high buffer length
    = 034A       ICAX1   EQU     $034A    ;1-byte first auxiliary information
    = 034B       ICAX2   EQU     $034B    ;1-byte second auxiliary informatio:
    = 034C       ICSPR   EQU     $034C    ;4-byte work area

    = 03C0       PRNBUF  EQU     $03C0    ;40-byte printer buffer
    = 03E8       SUPERF  EQU     $03E8    ;1-byte editor super function flag :
    = 03E9       CKEY    EQU     $03E9    ;1-byte cassette boot request flag :
    = 03EA       CASSBT  EQU     $03EA    ;1-byte cassette boot flag (0 = not:
    = 03EB       CARTCK  EQU     $03EB    ;1-byte cartridge equivalence check:
    = 03EC       DERRF   EQU     $03EC    ;1-byte screen OPEN error flag (0 =:

                 ;       Remainder of Page Three Not Cleared upon Reset

    = 03ED       ACMVAR  EQU     $03ED    ;11 bytes reserved for ACMI
    = 03F8       BASICF  EQU     $03F8    ;1-byte BASIC switch flag (0 = BASI:
    = 03F9       MINTLK  EQU     $03F9    ;1-byte ACMI module interlock
    = 03FA       GINTLK  EQU     $03FA    ;1-byte cartridge interlock
    = 03FB       CHLINK  EQU     $03FB    ;2-byte loaded handler chain link
    = 03FD       CASBUF  EQU     $03FD    ;3-byte first 3 bytes of cassette b:
```

```
                   **       Page Four Address Equates

                   ;        EQU      $0400   ;128-byte remainder of cassette buf:

                   ;        Reserved for Application

       = 0480      USAREA   EQU      $0480   ;128 bytes reserved for application


                   **       Page Five Address Equates

                   ;        Reserved for Application and Floating Point Package

                   ;        EQU      $0500   ;256 bytes reserved for application:


                   **       Floating Point Package Address Equates

       = 057E      LBPR1    EQU      $057E   ;1-byte LBUFF preamble
       = 057F      LBPR2    EQU      $057F   ;1-byte LBUFF preamble
       = 0580      LBUFF    EQU      $0580   ;128-byte line buffer

       = 05E0      PLYARG   EQU      $05E0   ;6-byte floating point polynomial a:
       = 05E6      FPSCR    EQU      $05E6   ;6-byte floating point temporary
       = 05EC      FPSCR1   EQU      $05EC   ;6-byte floating point temporary


                   **       Page Six Address Equates

                   ;        Reserved for Application

                   ;        EQU      $0600   ;256 bytes reserved for application
```

```
                    **       LNBUG Address Equates


      = 0000        LNBUG    IF       LNBUG
                    LNBUG    ENDIF



                    **       Cartridge Address Equates


      = BFFA        CARTCS   EQU      $BFFA    ;2-byte cartridge coldstart address
      = BFFC        CART     EQU      $BFFC    ;1-byte cartridge present indicator
      = BFFD        CARTFG   EQU      $BFFD    ;1-byte cartridge flags
      = BFFE        CARTAD   EQU      $BFFE    ;2-byte cartridge start vector



                    **       CTIA/GTIA Address Equates


      = D000        CTIA     EQU      $D000    ;CTIA/GTIA area

                    ;        Read/write Addresses

      = D01F        CONSOL   EQU      $D01F    ;console switches and speaker contr:

                    ;        Read Addresses

      = D000        M0PF     EQU      $D000    ;missle 0 and playfield collision
      = D001        M1PF     EQU      $D001    ;missle 1 and playfield collision
      = D002        M2PF     EQU      $D002    ;missle 2 and playfield collision
      = D003        M3PF     EQU      $D003    ;missle 3 and playfield collision

      = D004        P0PF     EQU      $D004    ;player 0 and playfield collision
      = D005        P1PF     EQU      $D005    ;player 1 and playfield collision
      = D006        P2PF     EQU      $D006    ;player 2 and playfield collision
      = D007        P3PF     EQU      $D007    ;player 3 and playfield collision

      = D008        M0PL     EQU      $D008    ;missle 0 and player collision
      = D009        M1PL     EQU      $D009    ;missle 1 and player collision
      = D00A        M2PL     EQU      $D00A    ;missle 2 and player collision
      = D00B        M3PL     EQU      $D00B    ;missle 3 and player collision

      = D00C        P0PL     EQU      $D00C    ;player 0 and player collision
      = D00D        P1PL     EQU      $D00D    ;player 1 and player collision
      = D00E        P2PL     EQU      $D00E    ;player 2 and player collision
      = D00F        P3PL     EQU      $D00F    ;player 3 and player collision

      = D010        TRIG0    EQU      $D010    ;joystick trigger 0
      = D011        TRIG1    EQU      $D011    ;joystick trigger 1

      = D012        TRIG2    EQU      $D012    ;cartridge interlock
      = D013        TRIG3    EQU      $D013    ;ACMI module interlock
      = D014         PAL      EQU      $D014    ;PAL/NTSC indicator

                     ;        Write Addresses

      = D000         HPOSP0   EQU      $D000    ;player 0 horizontal position
      = D001         HPOSP1   EQU      $D001    ;player 1 horizontal position
      = D002         HPOSP2   EQU      $D002    ;player 2 horizontal position
      = D003         HPOSP3   EQU      $D003    ;player 3 horizontal position

      = D004         HPOSM0   EQU      $D004    ;missle 0 horizontal position
      = D005         HPOSM1   EQU      $D005    ;missle 1 horizontal position
      = D006         HPOSM2   EQU      $D006    ;missle 2 horizontal position
      = D007         HPOSM3   EQU      $D007    ;missle 3 horizontal position

      = D008         SIZEP0   EQU      $D008    ;player 0 size
      = D009         SIZEP1   EQU      $D009    ;player 1 size
      = D00A         SIZEP2   EQU      $D00A    ;player 2 size
      = D00B         SIZEP3   EQU      $D00B    ;player 3 size

      = D00C         SIZEM    EQU      $D00C    ;missle sizes

      = D00D         GRAFP0   EQU      $D00D    ;player 0 graphics
      = D00E         GRAFP1   EQU      $D00E    ;player 1 graphics
      = D00F         GRAFP2   EQU      $D00F    ;player 2 graphics
      = D010         GRAFP3   EQU      $D010    ;player 3 graphics

      = D011         GRAFM    EQU      $D011    ;missle graphics

      = D012         COLPM0   EQU      $D012    ;player-missle 0 color/luminance
      = D013         COLPM1   EQU      $D013    ;player-missle 1 color/luminance
      = D014         COLPM2   EQU      $D014    ;player-missle 2 color/luminance
      = D015         COLPM3   EQU      $D015    ;player-missle 3 color/luminance

      = D016         COLPF0   EQU      $D016    ;playfield 0 color/luminance
      = D017         COLPF1   EQU      $D017    ;playfield 1 color/luminance
      = D018         COLPF2   EQU      $D018    ;playfield 2 color/luminance
      = D019         COLPF3   EQU      $D019    ;playfield 3 color/luminance

      = D01A         COLBK    EQU      $D01A    ;background color/luminance

      = D01B         PRIOR    EQU      $D01B    ;priority select
      = D01C         VDELAY   EQU      $D01C    ;vertical delay
      = D01D         GRACTL   EQU      $D01D    ;graphic control
      = D01E         HITCLR   EQU      $D01E    ;collision clear
```

```
                  **      PBI Address Equates

   = D100         PBI     EQU     $D100    ;parallel bus interface area

                  ;       Read Addresses

   = D1FF         PDVI    EQU     $D1FF    ;parallel device IRQ status

                  ;       Write Addresses

   = D1FF         PDVS    EQU     $D1FF    ;parallel device select



                  **      POKEY Address Equates

   = D200         POKEY   EQU     $D200    ;POKEY area

                  ;       Read Addresses

   = D200         POT0    EQU     $D200    ;potentiometer 0
   = D201         POT1    EQU     $D201    ;potentiometer 1
   = D202         POT2    EQU     $D202    ;potentiometer 2
   = D203         POT3    EQU     $D203    ;potentiometer 3
   = D204         POT4    EQU     $D204    ;potentiometer 4
   = D205         POT5    EQU     $D205    ;potentiometer 5
   = D206         POT6    EQU     $D206    ;potentiometer 6
   = D207         POT7    EQU     $D207    ;potentiometer 7

   = D208         ALLPOT  EQU     $D208    ;potentiometer port state
   = D209         KBCODE  EQU     $D209    ;keyboard code
   = D20A         RANDOM  EQU     $D20A    ;random number generator
   = D20D         SERIN   EQU     $D20D    ;serial port input
   = D20E         IRQST   EQU     $D20E    ;IRQ interrupt status
   = D20F         SKSTAT  EQU     $D20F    ;serial port and keyboard status

                  ;       Write Addresses

   = D200         AUDF1   EQU     $D200    ;channel 1 audio frequency
   = D201         AUDC1   EQU     $D201    ;channel 1 audio control

   = D202         AUDF2   EQU     $D202    ;channel 2 audio frequency
   = D203         AUDC2   EQU     $D203    ;channel 2 audio control

   = D204         AUDF3   EQU     $D204    ;channel 3 audio frequency
   = D205         AUDC3   EQU     $D205    ;channel 3 audio control

   = D206         AUDF4   EQU     $D206    ;channel 4 audio frequency
   = D207         AUDC4   EQU     $D207    ;channel 4 audio control

   = D208         AUDCTL  EQU     $D208    ;audio control
   = D209         STIMER  EQU     $D209    ;start timers
   = D20A         SKRES   EQU     $D20A    ;reset SKSTAT status
   = D20B         POTGO   EQU     $D20B    ;start potentiometer scan sequence
```

```
        = D20D       SEROUT  EQU    $D20D   ;serial port output
        = D20E       IRQEN   EQU    $D20E   ;IRQ interrupt enable
        = D20F       SKCTL   EQU    $D20F   ;serial port and keyboard control



                     **      PIA Address Equates


        = D300       PIA     EQU    $D300   ;PIA area

                     ;       Read/Write Addresses

        = D300       PORTA   EQU    $D300   ;port A direction register or jacks:
        = D301       PORTB   EQU    $D301   ;port B direction register or memor:

        = D302       PACTL   EQU    $D302   ;port A control
        = D303       PBCTL   EQU    $D303   ;port B control



                     **      ANTIC Address Equates


        = D400       ANTIC   EQU    $D400   ;ANTIC area

                     ;       Read Addresses

        = D40B       VCOUNT  EQU    $D40B   ;vertical line counter
        = D40C       PENH    EQU    $D40C   ;light pen horizontal position
        = D40D       PENV    EQU    $D40D   ;light pen vertical position
        = D40F       NMIST   EQU    $D40F   ;NMI interrupt status

                     ;       Write Addresses

        = D400       DMACTL  EQU    $D400   ;DMA control
        = D401       CHACTL  EQU    $D401   ;character control
        = D402       DLISTL  EQU    $D402   ;low display list address
        = D403       DLISTH  EQU    $D403   ;high disply list address
        = D404       HSCROL  EQU    $D404   ;horizontal scroll
        = D405       VSCROL  EQU    $D405   ;vertical scroll
        = D407       PMBASE  EQU    $D407   ;player-missle base address
        = D409       CHBASE  EQU    $D409   ;character base address
        = D40A       WSYNC   EQU    $D40A   ;wait for HBLANK synchronization
        = D40E       NMIEN   EQU    $D40E   ;NMI enable
        = D40F       NMIRES  EQU    $D40F   ;NMI interrupt status reset
```

```
                        **      PBI RAM Address Equates


      = D600            PBIRAM  EQU     $D600   ;parallel bus interface RAM area



                        **      ACMI Address Equates


      = 0000            ACMI    IF      ACMI
                        ACMI    ENDIF



                        **      Floating Point Package Address Equates

      = D800            AFP     EQU     $D800   ;convert ASCII to floating point
      = D8E6            FASC    EQU     $D8E6   ;convert floating point to ASCII
      = D9AA            IFP     EQU     $D9AA   ;convert integer to floating point
      = D9D2            FPI     EQU     $D9D2   ;convert floating point to integer
      = DA44            ZFR0    EQU     $DA44   ;zero FR0
      = DA46            ZF1     EQU     $DA46   ;zero floating point number
      = DA60            FSUB    EQU     $DA60   ;subtract floating point numbers
      = DA66            FADD    EQU     $DA66   ;add floating point numbers
      = DADB            FMUL    EQU     $DADB   ;multiply floating point numbers
      = DB28            FDIV    EQU     $DB28   ;divide floating point numbers
      = DD40            PLYEVL  EQU     $DD40   ;evaluate floating point polynomial
      = DD89            FLD0R   EQU     $DD89   ;load floating point number
      = DD8D            FLD0P   EQU     $DD8D   ;load floating point number
      = DD98            FLD1R   EQU     $DD98   ;load floating point number
      = DD9C            FLD1P   EQU     $DD9C   ;load floating point number
      = DDA7            FST0R   EQU     $DDA7   ;store floating point number
      = DDAB            FST0P   EQU     $DDAB   ;store floating point number
      = DDB6            FMOVE   EQU     $DDB6   ;move floating point number
      = DECD            LOG     EQU     $DECD   ;calculate floating point logarithm
      = DED1            LOG10   EQU     $DED1   ;calculate floating point base 10 l:
      = DDC0            EXP     EQU     $DDC0   ;calculate floating point exponenti:
      = DDCC            EXP10   EQU     $DDCC   ;calculate floating point base 10 e:
```

```
                **      Parallel Device Address Equates


   = D803       PDID1    EQU      $D803     ;parallel device ID 1
   = D805       PDIOV    EQU      $D805     ;parallel device I/O vector
   = D808       PDIRQV   EQU      $D808     ;parallel device IRQ vector
   = D80B       PDID2    EQU      $D80B     ;parallel device ID 2
   = D80D       PDVV     EQU      $D80D     ;parallel device vector table



                **      Device Handler Vector Table Address Equates


   = E400       EDITRV   EQU      $E400     ;editor handler vector table
   = E410       SCRENV   EQU      $E410     ;screen handler vector table
   = E420       KEYBDV   EQU      $E420     ;keyboard handler vector table
   = E430       PRINTV   EQU      $E430     ;printer handler vector table
   = E440       CASETV   EQU      $E440     ;cassette handler vector table



                **      Jump Vector Address Equates


   = E450       DINITV   EQU      $E450     ;vector to initialize DIO
   = E453       DSKINV   EQU      $E453     ;vector to DIO
   = E456       CIOV     EQU      $E456     ;vector to CIO
   = E459       SIOV     EQU      $E459     ;vector to SIO
   = E45C       SETVBV   EQU      $E45C     ;vector to set VBLANK parameters
   = E45F       SYSVBV   EQU      $E45F     ;vector to process immediate VBLANK:
   = E462       XITVBV   EQU      $E462     ;vector to process deferred VBLANK :
   = E465       SIOINV   EQU      $E465     ;vector to initialize SIO
   = E468       SENDEV   EQU      $E468     ;vector to enable SEND
   = E46B       INTINV   EQU      $E46B     ;vector to initialize interrupt han:
   = E46E       CIOINV   EQU      $E46E     ;vector to initialize CIO
   = E471       BLKBDV   EQU      $E471     ;vector to power-up display (former:
   = E474       WARMSV   EQU      $E474     ;vector to warmstart
   = E477       COLDSV   EQU      $E477     ;vector to coldstart
   = E47A       RBLOKV   EQU      $E47A     ;vector to read cassette block
   = E47D       CSOPIV   EQU      $E47D     ;vector to open cassette for input
   = E480       PUPDIV   EQU      $E480     ;vector to power-up display
   = E483       SLFTSV   EQU      $E483     ;vector to self-test
   = E486       PHENTV   EQU      $E486     ;vector to enter peripheral handler
   = E489       PHUNLV   EQU      $E489     ;vector to unlink peripheral handle:
   = E48C       PHINIV   EQU      $E48C     ;vector to initialize peripheral ha:
```

```
                    **      Generic Parallel Device Handler Vector Table Addres:

      = E48F        GPDVV   EQU     $E48F   ;generic parallel device handler ve:
```

```
0000                **      Self-test Page Zero Address Equates


        = 0080      STTIME  EQU     $0080     ;2-byte main screen timeout timer
        = 0082      STAUT   EQU     $0082     ;1-byte auto-mode flag
        = 0083      STJMP   EQU     $0083     ;3-byte ANTIC jump instruction
        = 0086      STSEL   EQU     $0086     ;1-byte selection
        = 0087      STPASS  EQU     $0087     ;1-byte pass
        = 0088      STSPP   EQU     $0088     ;1-byte SELECT previously pressed f:
                    ;       EQU     $0089     ;1-byte (not used)
        = 008A      STKST   EQU     $008A     ;1-byte keyboard self-test flag (0 :
        = 008B      STCHK   EQU     $008B     ;2-byte checksum
        = 008D      STSMM   EQU     $008D     ;1-byte screen memory mask
        = 008E      STSMP   EQU     $008E     ;1-byte screen memory pointer
        = 008F      ST1K    EQU     $008F     ;1-byte current 1K of memory to tes:
        = 0090      STPAG   EQU     $0090     ;2-byte current page to test
        = 0092      STPC    EQU     $0092     ;1-byte page count
        = 0093      STMVAL  EQU     $0093     ;1-byte correct value for memory te:
        = 0094      STSKP   EQU     $0094     ;1-byte simulated keypress index
        = 0095      STTMP1  EQU     $0095     ;2-byte temporary
        = 0097      STVOC   EQU     $0097     ;1-byte current voice indicator
        = 0098      STNOT   EQU     $0098     ;1-byte current note counter
        = 0099      STCDI   EQU     $0099     ;1-byte cleft display pointer
        = 009A      STCDA   EQU     $009A     ;1-byte cleft data pointer
        = 009B      STTMP2  EQU     $009B     ;2-byte temporary
        = 009D      STTMP3  EQU     $009D     ;1-byte temporary
        = 009E      STADR1  EQU     $009E     ;2-byte temporary address
        = 00A0      STADR2  EQU     $00A0     ;2-byte temporary address
        = 00A2      STBL    EQU     $00A2     ;1-byte blink counter
        = 00A3      STTMP4  EQU     $00A3     ;1-byte temporary
        = 00A4      STLM    EQU     $00A4     ;1-byte LED mask
        = 00A5      STTMP5  EQU     $00A5     ;1-byte temporary



                    **      Self-test Address Equates


        = 3000      ST3000  EQU     $3000     ;screen memory
        = 3002      ST3002  EQU     $3002     ;cleft display
        = 3004      ST3004  EQU     $3004     ;"VOICE #" text display
        = 300B      ST300B  EQU     $300B     ;voice number display
        = 301C      ST301C  EQU     $301C     ;START key display
        = 301E      ST301E  EQU     $301E     ;SELECT key display
        = 3020      ST3020  EQU     $3020     ;OPTION key display, first 8K ROM d:
        = 3021      ST3021  EQU     $3021     ;keyboard character display
        = 3022      ST3022  EQU     $3022     ;keyboard text display
        = 3024      ST3024  EQU     $3024     ;second 8K ROM display
        = 3028      ST3028  EQU     $3028     ;"RAM" text display
        = 3038      ST3038  EQU     $3038     ;RAM display
        = 303C      ST303C  EQU     $303C     ;fifth note display
        = 304C      ST304C  EQU     $304C     ;"B S" text display
        = 3052      ST3052  EQU     $3052     ;tab key display
        = 3062      ST3062  EQU     $3062     ;cleft display
        = 306D      ST306D  EQU     $306D     ;return key display
        = 3072      ST3072  EQU     $3072     ;control key display
```

```
      = 3092        ST3092  EQU      $3092   ;"SH" text display
      = 309E        ST309E  EQU      $309E   ;sixth note display
      = 30AB        ST30AB  EQU      $30AB   ;"SH" text display
      = 30B7        ST30B7  EQU      $30B7   ;"S P A C E   B A R" text display
      = 30C1        ST30C1  EQU      $30C1   ;cleft display
      = 30C2        ST30C2  EQU      $30C2   ;cleft display
      = 30C7        ST30C7  EQU      $30C7   ;third note display
      = 30CA        ST30CA  EQU      $30CA   ;fourth note display
      = 30F8        ST30F8  EQU      $30F8   ;third note display
      = 3100        ST3100  EQU      $3100   ;screen memory
      = 3121        ST3121  EQU      $3121   ;cleft display
      = 3122        ST3122  EQU      $3122   ;cleft display
      = 313C        ST313C  EQU      $313C   ;fifth note display
      = 3150        ST3150  EQU      $3150   ;first line of staff display
      = 3154        ST3154  EQU      $3154   ;first note display
      = 3181        ST3181  EQU      $3181   ;cleft display
      = 3182        ST3182  EQU      $3182   ;cleft display
      = 3186        ST3186  EQU      $3186   ;second note display
      = 318C        ST318C  EQU      $318C   ;fifth note display
      = 31B0        ST31B0  EQU      $31B0   ;second line of staff display
      = 31C2        ST31C2  EQU      $31C2   ;cleft display
      = 31CA        ST31CA  EQU      $31CA   ;fourth note display.
      = 31EE        ST31EE  EQU      $31EE   ;sixth note display
      = 31F1        ST31F1  EQU      $31F1   ;cleft display
      = 3210        ST3210  EQU      $3210   ;third line of staff display
      = 321A        ST321A  EQU      $321A   ;fourth note display
      = 3248        ST3248  EQU      $3248   ;third note display
      = 3270        ST3270  EQU      $3270   ;fourth line of staff display
      = 32D0        ST32D0  EQU      $32D0   ;fifth line of staff display
```

```
0000                **      FIX - Fix Address
                    *
                    *       FIX sets the origin counter to the value specified :
                    *       argument.  If the current origin counter is less th:
                    *       argument, FIX fills the intervening bytes with zero:
                    *       issues a message to document the location and numbe:
                    *       bytes that are zero filled.
                    *
                    *       ENTRY   FIX     address
                    *
                    *
                    *       EXIT
                    *               Origin counter set to specified address.
                    *               Message issued if zero fill required.
                    *
                    *       CHANGES
                    *               -none-
                    *
                    *       CALLS
                    *               -none-
                    *
                    *       NOTES
                    *               Due to ECHO limitiation of 255 iterations, :
                    *               recursive.
                    *               If the current origin counter value is beyo:
                    *               argument, FIX generates an error.
                    *
                    *       MODS
                    *               R. K. Nordin    11/01/83


                    FIX     MACRO   address
                            IF      %1 <> *0
                            IF      %1 > *0
                            IF      %1 - *0 < 256
                            MSG     '$',%1 - *0,' free bytes from $',*0,' to $':
                            ECHO    %1 - *0
                            DB      0
                            ENDM
                            ELSE
                            FIX     *0 + 255
                            FIX     %1
                            ENDIF
                            ELSE
                            ERR             ;%1 precedes current origin counter:
                            ENDIF
                            ENDIF
                            ORG     *0
                            ENDM
```

```
 `00  = C000              ORG     $C000



                 **       First 8K ROM Identification and Checksum

C000  0000                Dw      $0000                         ;reserved f:
C002  100583              DB      IDDAY,IDMON,IDYEAR            ;date (day,:
C005  00                  DB      $00                           ;not used
C006  4242000001          DB      IDPN1,IDPN2,IDPN3,IDPN4,IDPN5 ;part numbe:
C008  02                  DB      IDREV                         ;revision n:
```

```
C00C                   **      IIH - Initialize Interrupt Handler
                       *
                       *       ENTRY   JSR     IIH
                       *               TRIG3 = ACMI module interlock
                       *               TRIG2 = cartridge interlock
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83

        = C00C         IIH     =       *       ;entry

C00C  A940                     LDA     #$40
C00E  8D0ED4                   STA     NMIEN   ;disable DLI and enable VBLANK NMI

C011  AD13D0                   LDA     TRIG3   ;cartridge interlock
C014  8DFA03                   STA     GINTLK  ;cartridge interlock status

        = 0000         ACMI    IF      ACMI
                       ACMI    ENDIF

C017  60                       RTS             ;return



                       **      NMI - Process NMI
                       *
                       *       ENTRY   JMP     NMI
                       *
                       *       EXIT
                       *               Exits via appropriate vector to process NMI
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83

        = C018         NMI     =       *               ;entry

                               ASSERT  $C0=high NMI    ;for compatibility with LNB:

                       ;       Check for display list NMI.

C018  2C0FD4                   BIT     NMIST
C01B  1003 ^C020               BPL     NMI1            ;if not display list NMI

C01D  6C0002                   JMP     (VDSLST)        ;process display list NMI, :

                       ;       Initialize.

C020  D8               NMI1    CLD

                       ;       Save registers.
```

```
  21  48                    PHA                           ;save A
C022  8A                    TXA
C023  48                    PHA                           ;save X
C024  98                    TYA
C025  48                    PHA                           ;save Y

                  ;         Reset NMI status.

C026  8D0FD4                STA      NMIRES               ;reset NMI status

                  ;         Process NMI.

      = 0000      LNBUG     IF       LNBUG
                  LNBUG     ELSE
C029  6C2202                JMP      (VVBLKI)             ;process immediate VBLANK N:
                  LNBUG     ENDIF


                  **        IRQ - Process IRQ
                  *
                  *         ENTRY    JMP      IRQ
                  *
                  *         EXIT
                  *                  Exits via VIMIRQ vector
                  *
                  *         MODS
                  *                  Original Author Unknown
                  *                  1. Bring closer to Coding Standard (object :
                  *                     R. K. Nordin-11/01/83

      = C02C      IRQ       =        *                    ;entry

                  ;         Initialize.

C02C  D8                    CLD

                  ;         Process IRQ.

      = 0000      LNBUG     IF       LNBUG
                  LNBUG     ELSE
C02D  6C1602                JMP      (VIMIRQ)             ;process immediate IRQ, ret:
                  LNBUG     ENDIF
```

```
                        **      IIR - Process Immediate IRQ
                        *
                        *       ENTRY   JMP     IIR
                        *
                        *       EXIT
                        *               Exits via appropriate vector to process IRQ
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C030          IIR     =       *               ;entry

                        ;       Initialize.

 C030  48                       PHA                     ;save A

                        ;       Check for serial input ready IRQ.

 C031  AD0ED2                   LDA     IRQST           ;IRQ status
 C034  2920                     AND     #$20            ;serial input ready
 C036  D00D ^C045               BNE     IIR1            ;if not serial input ready

                        ;       Process serial input IRQ.

 C038  A9DF                     LDA     #not $20        ;all other interrupts
 C03A  8D0ED2                   STA     IRQEN           ;enable all other interrupt:
 C03D  A510                     LDA     POKMSK
 C03F  8D0ED2                   STA     IRQEN
 C042  6C0A02                   JMP     (VSERIN)        ;process serial input ready:

                        ;       Process possible ACMI IRQ.

 C045                   IIR1
        = 0000          ACMI    IF      ACMI
                        ACMI    ENDIF

                        ;       Initialize further.

 C045  8A                       TXA
 C046  48                       PHA                     ;save X

                        ;       Check for parallel device IRQ.

 C047  ADFFD1                   LDA     PDVI            ;parallel device IRQ status:
 C04A  2D4902                   AND     PDIMSK          ;select desired IRQ statuse:
 C04D  F003 ^C052               BEQ     IIR2            ;if no desired IRQ

                        ;       Process parallel device IRQ.

 C04F  6C3802                   JMP     (VPIRQ)         ;process parallel device IR:

                        ;       Check other types of IRQ.

 C052  A206             IIR2    LDX     #TIRQL-1-1      ;offset to next to last ent:
```

```
C054  BDCFC0         IIR3    LDA     TIRQ,X           ;IRQ type
C057  E005                   CPX     #5               ;offset to serial out compl:
C059  D004 ^C05F             BNE     IIR4             ;if not serial out complete

C05B  2510                   AND     POKMSK           ;and with POKEY IRQ enable
C05D  F005 ^C064             BEQ     IIR5             ;if serial out complete not:

C05F  2C0ED2         IIR4    BIT     IRQST            ;IRQ interrupt status
C062  F006 ^C06A             BEQ     IIR6             ;if interrupt found

C064  CA             IIR5    DEX
C065  10ED ^C054             BPL     IIR3             ;if not done

                     ;       Coninue IRQ processing.

C067  4CA0C0                 JMP     CIR              ;continue IRQ processing, r:

                     ;       Enable other interrupts.

C06A  49FF           IIR6    EOR     #$FF             ;complement mask
C06C  8D0ED2                 STA     IRQEN            ;enable all others
C06F  A510                   LDA     POKMSK           ;POKEY IRQ mask
C071  8D0ED2                 STA     IRQEN            ;enable indicated IRQ's

                     ;       Check for BREAK key IRQ.

C074  E000                   CPX     #0
C076  D005 ^C07D             BNE     IIR7             ;if not BREAK key IRQ

                     ;       Check for keyboard disabled.

C078  AD6D02                 LDA     KEYDIS
C07B  D023 ^C0A0             BNE     CIR              ;if keyboard disabled, cont:

                     ;       Process IRQ.

C07D  BDD7C0         IIR7    LDA     TOIH,X           ;offset to interrupt handle:
C080  AA                     TAX
C081  BD0002                 LDA     INTABS,X         ;interrupt handler address
C084  8D8C02                 STA     JVECK
C087  BD0102                 LDA     INTABS+1,X
C08A  8D8D02                 STA     JVECK+1
C08D  68                     PLA
C08E  AA                     TAX                      ;restore X
C08F  6C8C02                 JMP     (JVECK)          ;process interrupt, return
```

```
                        **      BIR - Process BREAK Key IRQ
                        *
                        *       ENTRY   JMP     BIR
                        *
                        *       EXIT
                        *               Exits via RTI
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C092          BIR     =       *               ;entry

                        ;       Process BREAK.

 C092  A900                     LDA     #0
 C094  8511                     STA     BRKKEY  ;clear BREAK key flag
 C096  8DFF02                   STA     SSFLAG  ;clear start/stop flag
 C099  8DF002                   STA     CRSINH  ;enable cursor
 C09C  854D                     STA     ATRACT  ;turn off attract-mode

                        ;       Exit.

 C09E  68               BIR1    PLA             ;restore A
 C09F  40                       RTI             ;return


                        **      CIR - Continue IRQ Processing
                        *
                        *       ENTRY   JMP     CIR
                        *
                        *       EXIT
                        *               Exits via appropriate vector to process IRQ:
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C0A0          CIR     =       *                       ;entry

                        ;       Initialize.

 C0A0  68                       PLA
 C0A1  AA                       TAX

                        ;       Check for port A interrupt.

 C0A2  2C02D3                   BIT     PACTL           ;port A control
 C0A5  1006 ^C0AD               BPL     CIR1            ;if not port A interrupt

                        ;       Process proceed line IRQ.
```

```
C0A7  AD00D3                 LDA     PORTA           ;clear interrupt status bit
C0AA  6C0202                 JMP     (VPRCED)        ;process proceed line IRQ, :

                      ;      Check for port B interrupt.

C0AD  2C03D3          CIR1   BIT     PBCTL           ;port B control
C0B0  1006 ^C0B8             BPL     CIR2            ;if not port B interrupt

                      ;      Process interrupt line IRQ.

C0B2  AD01D3                 LDA     PORTB           ;clear interrupt status bit
C0B5  6C0402                 JMP     (VINTER)        ;process interrupt line IRQ:

                      ;      Check for BRK instruction IRQ.

C0B8  68              CIR2   PLA
C0B9  8D8C02                 STA     JVECK

C0BC  68                     PLA                     ;saved P
C0BD  48                     PHA                     ;resave P
C0BE  2910                   AND     #$10            ;B bit of P register
C0C0  F007 ^C0C9             BEQ     CIR3            ;if not BRK instruction IRQ

                      ;      Process BRK instruction IRQ.

C0C2  AD8C02                 LDA     JVECK
C0C5  48                     PHA
C0C6  6C0602                 JMP     (VBREAK)        ;process BRK instruction IR:

                      ;      Exit IRQ processing.

C0C9  AD8C02          CIR3   LDA     JVECK
C0CC  48                     PHA
                      ;      JMP     XIR             ;exit IRQ processing, retur:


                      **     XIR - Exit IRQ Processing
                      *
                      *      ENTRY   JMP     XIR
                      *
                      *      EXIT
                      *              Exits to RIR
                      *
                      *      MODS
                      *              Original Author Unknown
                      *              1. Bring closer to Coding Standard (object :
                      *                 R. K. Nordin 11/01/83


      = C0CD          XIR    =       *       ;entry
C0CD  68                     PLA             ;restore A
                      ;      JMP     RIR     ;return from interrupt
```

```
                        **      RIR - Return from Interrupt
                        *
                        *       ENTRY   JMP     RIR
                        *
                        *       EXIT
                        *               Exits via RTI
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = C0CE            RIR     =       *               ;entry
 C0CE  40                       RTI                     ;return


                        **      AIR - Process ACMI IRQ
                        *
                        *       ENTRY   JSR     AIR
                        *
                        *       EXIT    Exits via ACMISR vector
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = 0000            ACMI    IF      ACMI
                        ACMI    ENDIF


                        **      TIRQ - Table of IRQ Types
                        *
                        *       Entry n is the interrupt indicator of priority n (0:
                        *
                        *       NOTES
                        *               Problem: entry 7 (serial input ready) not u:

 C0CF  80               TIRQ    DB      $80     ;0 - BREAK key IRQ
 C0D0  40                       DB      $40     ;1 - keyboard IRQ
 C0D1  04                       DB      $04     ;2 - timer 4 IRQ
 C0D2  02                       DB      $02     ;3 - timer 2 IRQ
 C0D3  01                       DB      $01     ;4 - timer 1 IRQ
 C0D4  08                       DB      $08     ;5 - serial output complete IRQ
 C0D5  10                       DB      $10     ;6 - serial output ready IRQ
 C0D6  20                       DB      $20     ;7 - serial input ready IRQ

      = 0008            TIRQL   =       *-TIRQ  ;length
```

```
                        **      TOIH - Table of Offsets to Interrupt Handlers
                        *
                        *       Entry n is the offset to the interrupt handler vect:
                        *       corresponding to entry n of TIRQ.
                        *
                        *       NOTES
                        *               Problem: entry 7 (serial input ready) not u:

C0D7  36                TOIH    DB      BRKKY-INTABS    ;0 - BREAK key IRQ
C0D8  08                        DB      VKEYBD-INTABS   ;1 - keyboard IRQ
C0D9  14                        DB      VTIMR4-INTABS   ;2 - timer 4 IRQ
C0DA  12                        DB      VTIMR2-INTABS   ;3 - timer 2 IRQ
C0DB  10                        DB      VTIMR1-INTABS   ;4 - timer 1 IRQ
C0DC  0E                        DB      VSEROC-INTABS   ;5 - serial output complete:
C0DD  0C                        DB      VSEROR-INTABS   ;6 - serial output ready IR:
C0DE  0A                        DB      VSERIN-INTABS   ;7 - serial input ready IRQ


                        **      WFR - Wait for RESET
                        *
                        *       WFR loops forever.
                        *
                        *       ENTRY   JMP     WFR
                        *
                        *       EXIT
                        *               Does not exit
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = C0DF            WFR     =       *       ;entry

                        ;       Loop forever, waiting for RESET.

C0DF  4CDFC0            WFR1    JMP     WFR1    ;loop


                        **      IVNM - Process Immediate VBLANK NMI
                        *
                        *       ENTRY   JMP     IVNM
                        *
                        *       EXIT
                        *               Exits to DVNM or via VVBLKD vector
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83
```

```
          = C0E2       IVNM     =       *               ;entry

                       ;        Increment frame counter and attract-mode counter.

C0E2  E614                      INC     RTCLOK+2        ;increment low frame counte:
C0E4  D008 ^C0EE                BNE     IVN1            ;if low counter not zero

C0E6  E64D                      INC     ATRACT          ;increment attract-mode cou:
C0E8  E613                      INC     RTCLOK+1        ;increment middle frame cou:
C0EA  D002 ^C0EE                BNE     IVN1            ;if middle counter not zero

C0EC  E612                      INC     RTCLOK          ;increment high frame count:

                       ;        Set attract-mode effects.

C0EE  A9FE             IVN1     LDA     #$FE            ;select no luminance change
C0F0  A200                      LDX     #0              ;select no color shift
C0F2  A44D                      LDY     ATRACT          ;attract-mode timer/flag
C0F4  1006 ^C0FC                BPL     IVN2            ;if not attract-mode

C0F6  854D                      STA     ATRACT          ;ensure continued attract-m:
C0F8  A613                      LDX     RTCLOK+1        ;select color shift
C0FA  A9F6                      LDA     #$F6            ;select lower luminance

C0FC  854E             IVN2     STA     DRKMSK          ;attract-mode luminance
C0FE  864F                      STX     COLRSH          ;attract-mode color shift

                       ;        Update COLPF1 (in case fine scrolling and critical :

C100  ADC502                    LDA     COLOR1          ;playfield 1 color
C103  454F                      EOR     COLRSH          ;modify color with attract-:
C105  254E                      AND     DRKMSK          ;modify with attract-mode l:
C107  8D17D0                    STA     COLPF1          ;set playfield 1 color/lumi:

                       ;        Process countdown timer 1.

C10A  A200                      LDX     #0              ;indicate countdown timer 1
C10C  2055C2                    JSR     DCT             ;decrement countdown timer
C10F  D003 ^C114                BNE     IVN3            ;if timer not expired

C111  204FC2                    JSR     PTO             ;process countdown timer 1 :

                       ;        Check for critical section.

C114  A542             IVN3     LDA     CRITIC
C116  D008 ^C120                BNE     IVN4            ;if critical section

                       ;        Check for IRQ enabled.

C118  BA                        TSX                     ;stack pointer
C119  BD0401                    LDA     $0104,X         ;stacked P
C11C  2904                      AND     #$04            ;I (IRQ disable) bit
C11E  F003 ^C123                BEQ     IVN5            ;if IRQ enabled

                       ;        Exit.
```

```
C120  4C8AC2        IVN4    JMP     DVNM            ;process deferred VBLANK NM;

                    ;       Process IRQ enabled non-critical section.

C123                IVN5

                    ;       Check for ACMI module change.

      = 0000        ACMI    IF      ACMI
                    ACMI    ENDIF

                    ;       Check for cartridge change.

C123  AD13D0                LDA     TRIG3           ;cartridge interlock
C126  CDFA03                CMP     GINTLK          ;previous cartridge interlo:
C129  D0B4 ^C0DF            BNE     WFR             ;if cartridge change, wait :

                    ;       Set hardware registers from shadows.

C12B  AD0DD4                LDA     PENV
C12E  8D3502                STA     LPENV           ;light pen vertical positio:
C131  AD0CD4                LDA     PENH
C134  8D3402                STA     LPENH           ;light pen vertical positio:
C137  AD3102                LDA     SDLSTH
C13A  8D03D4                STA     DLISTH          ;high display list address
C13D  AD3002                LDA     SDLSTL
C140  8D02D4                STA     DLISTL          ;low display list address
C143  AD2F02                LDA     SDMCTL
C146  8D00D4                STA     DMACTL          ;DMA control
C149  AD6F02                LDA     GPRIOR
C14C  8D1BD0                STA     PRIOR           ;priority select

                    ;       Check for vertical scroll enabled.

C14F  AD6C02                LDA     VSFLAG          ;vertical scroll count
C152  F00E ^C162            BEQ     IVN6            ;if vertical scroll not ena:

                    ;       Scroll one line.

C154  CE6C02                DEC     VSFLAG          ;decrement vertical scroll :
C157  A908                  LDA     #8              ;scroll one line
C159  38                    SEC
C15A  ED6C02                SBC     VSFLAG          ;subtract vertical scroll c:
C15D  2907                  AND     #07
C15F  8D05D4                STA     VSCROL          ;set vertical scroll

                    ;       Turn off speaker.

C162  A208          IVN6    LDX     #$08            ;speaker off
C164  8E1FD0                STX     CONSOL          ;set speaker control

                    ;       Set color registers from shadows.

                    ;       LDX     #8              ;offset to background color

C167  58            IVN7    CLI
C168  BDC002                LDA     PCOLR0,X        ;color register shadow
```

```
C16B  454F                    EOR     COLRSH          ;modify with attract-mode c:
C16D  254E                    AND     DRKMSK          ;modify with attract-mode l:
C16F  9D12D0                  STA     COLPM0,X        ;set color register
C172  CA                      DEX
C173  10F2 ^C167              BPL     IVN7            ;if not done

                      ;       Set character set control.

C175  ADF402                  LDA     CHBAS
C178  8D09D4                  STA     CHBASE
C17B  ADF302                  LDA     CHACT
C17E  8D01D4                  STA     CHACTL

                      ;       Process countdown timer 2.

C181  A202                    LDX     #2              ;indicate countdown timer 2
C183  2055C2                  JSR     DCT             ;decrement countdown timer
C186  D003 ^C18B              BNE     IVN8            ;if timer not expired

C188  2052C2                  JSR     PTT             ;process countdown timer 2 :

                      ;       Process timers 3, 4 and 5.

C18B  A202            IVN8    LDX     #2              ;preset offset to timer 2

C18D  E8              IVN9    INX
C18E  E8                      INX                     ;offset to countdown timer
C18F  BD1802                  LDA     CDTMV3-4,X      ;countdown timer
C192  1D1902                  ORA     CDTMV3+1-4,X
C195  F006 ^C19D              BEQ     IVN10           ;if countdown timer already:

C197  2055C2                  JSR     DCT             ;decrement countdown timer
C19A  9D2602                  STA     CDTMF3-4,X      ;indicate timer expiration :

C19D  E008            IVN10   CPX     #8              ;offset to timer 5
C19F  D0EC ^C18D              BNE     IVN9            ;if all timers not done

                      ;       Check debounce counter.

C1A1  AD0FD2                  LDA     SKSTAT          ;keyboard status
C1A4  2904                    AND     #$04            ;key down indicator
C1A6  F008 ^C1B0              BEQ     IVN11           ;if key down

                      ;       Process key up.

C1A8  ADF102                  LDA     KEYDEL          ;key delay counter
C1AB  F003 ^C1B0              BEQ     IVN11           ;if counted down already

C1AD  CEF102                  DEC     KEYDEL          ;decrement key delay counte:

                      ;       Check software key repeat timer.

C1B0  AD2B02          IVN11   LDA     SRTIMR          ;key repeat timer
C1B3  F03E ^C1F3              BEQ     IVN13           ;if key repeat timer expire:

C1B5  AD0FD2                  LDA     SKSTAT          ;keyboard status
C1B8  2904                    AND     #$04            ;key down indicator
```

```
C1BA  D032 ^C1EE            BNE     IVN12           ;if key no longer down

C1BC  CE2B02                DEC     SRTIMR          ;decrement key repeat timer
C1BF  D032 ^C1F3            BNE     IVN13           ;if key repeat timer not ex:

                    ;       Process key repeat timer expiration.

C1C1  AD6D02                LDA     KEYDIS          ;keyboard disable flag
C1C4  D02D ^C1F3            BNE     IVN13           ;if keyboard disabled, no r:

C1C6  ADDA02                LDA     KEYREP          ;initial timer value
C1C9  8D2B02                STA     SRTIMR          ;reset key repeat timer
C1CC  AD09D2                LDA     KBCODE          ;key code

                    ;       Check for hidden codes.

C1CF  C99F                  CMP     #CNTL1
C1D1  F020 ^C1F3            BEQ     IVN13           ;if CTRL-1

C1D3  C983                  CMP     #CNTLF1
C1D5  F01C ^C1F3            BEQ     IVN13           ;if CTRL-F1

C1D7  C984                  CMP     #CNTLF2
C1D9  F018 ^C1F3            BEQ     IVN13           ;if CTRL-F2

C1DB  C994                  CMP     #CNTLF4
C1DD  F014 ^C1F3            BEQ     IVN13           ;if CTRL-F4

C1DF  293F                  AND     #$3F
C1E1  C911                  CMP     #HELP
C1E3  F00E ^C1F3            BEQ     IVN13           ;if HELP

                    ;       Set key code.

C1E5  AD09D2                LDA     KBCODE          ;key code
C1E8  8DFC02                STA     CH              ;set key code
C1EB  4CF3C1                JMP     IVN13           ;continue

                    ;       Zero key repeat timer.

C1EE  A900          IVN12   LDA     #0
C1F0  8D2B02                STA     SRTIMR          ;zero key repeat timer

                    ;       Read joysticks.

C1F3  AD00D3        IVN13   LDA     PORTA           ;joystick readings
C1F6  4A                    LSR     A
C1F7  4A                    LSR     A
C1F8  4A                    LSR     A
C1F9  4A                    LSR     A               ;joystick 1 reading
C1FA  8D7902                STA     STICK1          ;set joystick 1 reading
      = FFFF        VGC     IF      VGC
C1FD  8D7B02                STA     STICK3          ;simulate joystick 3 readin:
                    VGC     ENDIF
C200  AD00D3                LDA     PORTA           ;joystick readings
C203  290F                  AND     #$0F            ;joystick 0 reading
C205  8D7802                STA     STICK0          ;set joystick 0 reading
```

```
        = FFFF      VGC     IF      VGC
 C208   8D7A02              STA     STICK2              ;simulate joystick 2 readin:
                    VGC     ENDIF

                    ;       Read joystick triggers.

 C20B   AD10D0              LDA     TRIG0               ;trigger 0 indicator
 C20E   8D8402              STA     STRIG0              ;set trigger 0 indicator
        = FFFF      VGC     IF      VGC
 C211   8D8602              STA     STRIG2              ;simulate trigger 2 indicat:
                    VGC     ENDIF
 C214   AD11D0              LDA     TRIG1               ;trigger 1 indicator
 C217   8D8502              STA     STRIG1              ;set trigger 1 indicator
        = FFFF      VGC     IF      VGC
 C21A   8D8702              STA     STRIG3              ;simulate trigger 3 indicat:
                    VGC     ENDIF

                    ;       Read potentiometers.

 C21D   A203                LDX     #3                  ;offset to last potentiomet:

 C21F   BD00D2      IVN14   LDA     POT0,X              ;potentiometer reading
 C222   9D7002              STA     PADDL0,X            ;set potentiometer reading
        = FFFF      VGC     IF      VGC
 C225   9D7402              STA     PADDL4,X            ;simulate potentiometer rea:
                    VGC     ENDIF
 C228   CA                  DEX
 C229   10F4 ^C21F          BPL     IVN14               ;if not done

                    ;       Start potentiometers for next time.

 C22B   8D0BD2              STA     POTGO               ;start potentiometers

                    ;       Read paddle triggers.

 C22E   A202                LDX     #2                  ;offset to paddle trigger r:
 C230   A001                LDY     #1                  ;offset to joystick reading

 C232   B97802      IVN15   LDA     STICK0,Y            ;joystick reading
 C235   4A                  LSR     A
 C236   4A                  LSR     A
 C237   4A                  LSR     A                   ;paddle trigger reading
 C238   9D7D02              STA     PTRIG1,X            ;set paddle trigger reading
        = FFFF      VGC     IF      VGC
 C23B   9D8102              STA     PTRIG5,X            ;simulate paddle trigger re:
                    VGC     ENDIF

 C23E   A900                LDA     #0
 C240   2A                  ROL     A                   ;paddle trigger reading
 C241   9D7C02              STA     PTRIG0,X            ;set paddle trigger reading
        = FFFF      VGC     IF      VGC
 C244   9D8002              STA     PTRIG4,X            ;simulate paddle trigger re:
                    VGC     ENDIF
 C247   CA                  DEX
 C248   CA                  DEX
 C249   88                  DEY
 C24A   10E6 ^C232          BPL     IVN15               ;if not done
```

```
                          ;       Exit.

 C24C  6C2402                     JMP     (VVBLKD)        ;process deferred VBLANK NM:


                          **      PTO - Process Countdown Timer One Expiration
                          *
                          *       ENTRY   JSR     PTO
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

      = C24F        PTO           =       *               ;entry
 C24F  6C2602                     JMP     (CDTMA1)        ;process countdown timer 1 :


                          **      PTT - Process Countdown Timer Two Expiration
                          *
                          *       ENTRY   JSR     PTT
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

      = C252        PTT           =       *               ;entry
 C252  6C2802                     JMP     (CDTMA2)        ;process countdown timer 2 :


                          **      DCT - Decrement Countdown Timer
                          *
                          *       ENTRY   JSR     DCT
                          *               X = offset to timer value
                          *
                          *       EXIT
                          *               A = 0, if timer expired
                          *                 = $FF, if timer did not expire
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

      = C255        DCT           =       *               ;entry
 C255  BC1802                     LDY     CDTMV1,X        ;low timer value
```

```
C258  D008 ^C262           BNE    DCT1             ;if low timer value not zer:

C25A  BC1902               LDY    CDTMV1+1,X       ;high timer value
C25D  F010 ^C26F           BEQ    DCT2             ;if timer value zero, exit

C25F  DE1902               DEC    CDTMV1+1,X       ;decrement high timer value

C262  DE1802        DCT1   DEC    CDTMV1,X         ;decrement low timer value
C265  D008 ^C26F           BNE    DCT2             ;if low timer value not zer:

C267  BC1902               LDY    CDTMV1+1,X       ;high timer value
C26A  D003 ^C26F           BNE    DCT2             ;if high timer value not ze:

C26C  A900                 LDA    #0               ;indicate timer expired
C26E  60                   RTS                     ;return

C26F  A9FF          DCT2   LDA    #$FF             ;indicate timer did not exp:
C271  60                   RTS                     ;return


                    **     SVP - Set Vertical Blank Parameters
                    *
                    *      SVP sets countdown timers and VBLANK vectors.
                    *
                    *      ENTRY   JSR     SVP
                    *              X = high initial timer value or high vector:
                    *              Y = low initial timer value or low vector a:
                    *              A = 1, if timer 1 value
                    *                  2, if timer 2 value
                    *                  3, if timer 3 value
                    *                  4, if timer 4 value
                    *                  5, if timer 5 value
                    *                  6, if immediate VBLANK vector
                    *                  7, if deferred VBLANK vector
                    *
                    *      MODS
                    *              Original Author Unknown
                    *              1. Bring closer to Coding Standard (object :
                    *                 R. K. Nordin 11/01/83

            = C272  SVP    =      *                ;entry

                    ;      Initialize.

C272  0A                   ASL    A                ;compute offset+2 to value :
C273  8D2D02               STA    INTEMP           ;offset+2 to value or vecto:
C276  8A                   TXA                     ;high timer value or high v:

                    ;      Ensure no VBLANK in progress by delaying after HBLA:

C277  A205                 LDX    #5               ;20 CPU cycles
C279  8D0AD4               STA    WSYNC            ;wait for HBLANK synchroniz:

C27C  CA            SVP1   DEX
```

```
C27D  D0FD ^C27C              BNE     SVP1            ;if not done delaying

                        ;     Set timer value or vector address.

C27F  AE2D02                  LDX     INTEMP          ;offset+2 to value or vecto:
C282  9D1702                  STA     CDTMV1-2+1,X    ;high timer value or high v:
C285  98                      TYA
C286  9D1602                  STA     CDTMV1-2,X      ;low timer value or low vec:
C289  60                      RTS                     ;return



                        **    DVNM - Process Deferred VBLANK NMI
                        *
                        *     ENTRY   JMP     DVNM
                        *
                        *     EXIT
                        *             Exits via RTI
                        *
                        *     MODS
                        *             Original Author Unknown.
                        *             1. Bring closer to Coding Standard (object :
                        *                R. K. Nordin 11/01/83

      = C28A            DVNM  =       *       ;entry
C28A  68                      PLA
C28B  A8                      TAY             ;restore Y
C28C  68                      PLA
C28D  AA                      TAX             ;restore X
C28E  68                      PLA             ;restore A
C28F  40                      RTI             ;return
```

```
C290                **      PWS - Perform Warmstart
                    *
                    *       ENTRY   JMP     PWS
                    *
                    *       EXIT
                    *               Exits to PCS or PRS
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

        = C290      PWS     =       *       ;entry

                    ;       Initialize.

 C290  78                   SEI

                    ;       Check for cartridge change.

 C291  AD13D0               LDA     TRIG3   ;cartridge interlock
 C294  CDFA03               CMP     GINTLK  ;previous cartridge interlock statu:
 C297  D02F ^C2C8           BNE     PCS     ;if cartridge changed, perform cold:

                    ;       Check for cartridge.

 C299  6A                   ROR     A
 C29A  9005 ^C2A1           BCC     PWS1    ;if no cartridge

                    ;       Verify no change in cartridge.

 C29C  20C9C4               JSR     CCE     ;check cartridge equivalence
 C29F  D027 ^C2C8           BNE     PCS     ;if different cartridge, coldstart

                    ;       Check coldstart status.

 C2A1  AD4402       PWS1    LDA     COLDST  ;coldstart status
 C2A4  D022 ^C2C8           BNE     PCS     ;if coldstart was in progress, perf:

                    ;       Perform warmstart.

 C2A6  A9FF                 LDA     #$FF    ;indicate warmstart
 C2A8  D020 ^C2CA           BNE     PRS     ;preset memory, return
```

```
                        **      RES - Process RESET
                        *
                        *       ENTRY   JMP     RES
                        *
                        *       EXIT
                        *               Exits to PCS, if coldstart, or PWS, if warm:
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C2AA          RES     =       *       ;entry

                        ;       Initialize.

  C2AA  78                      SEI

                        ;       Delay 0.1 second for RESET bounce.

  C2AB  A28C                    LDX     #140    ;0.1 second delay

  C2AD  88              RES1    DEY
  C2AE  D0FD ^C2AD              BNE     RES1    ;if inner loop not done

  C2B0  CA                      DEX
  C2B1  D0FA ^C2AD              BNE     RES1    ;if outer loop not done

                        ;       Check power-up validation bytes.

  C2B3  AD3D03                  LDA     PUPBT1
  C2B6  C95C                    CMP     #PUPVL1
  C2B8  D00E ^C2C8              BNE     PCS     ;if validation byte 1 differs, cold:

  C2BA  AD3E03                  LDA     PUPBT2
  C2BD  C993                    CMP     #PUPVL2
  C2BF  D007 ^C2C8              BNE     PCS     ;if validation byte 2 differs, cold:

  C2C1  AD3F03                  LDA     PUPBT3
  C2C4  C925                    CMP     #PUPVL3
  C2C6  F0C8 ^C290              BEQ     PWS     ;if all bytes validated, perform wa:

                        ;       JMP     PCS     ;perform coldstart, return
```

```
                         **     PCS - Perform Coldstart
                         *
                         *      ENTRY   JMP     PCS
                         *
                         *      EXIT
                         *              Exits to PRS
                         *
                         *      MODS
                         *              Original Author Unknown
                         *              1. Bring closer to Coding Standard (object :
                         *                 R. K. Nordin 11/01/83

         = C2C8          PCS    =       *       ;entry
C2C8  A900                      LDA     #0      ;indicate coldstart
                         ;      JMP     PRS     ;preset memory, return


                         **     PRS - Preset Memory
                         *
                         *      ENTRY   JMP     PRS
                         *
                         *      EXIT
                         *              Exits via CARTCS vector or DOSVEC vector
                         *
                         *      NOTES
                         *              Problem: in the CRASS65 version, APPMHI was:
                         *              zero-page.
                         *
                         *      MODS
                         *              Original Author Unknown
                         *              1. Bring closer to Coding Standard (object :
                         *                 R. K. Nordin 11/01/83

         = C2CA          PRS    =       *               ;entry

                         ;      Update warmstart flag.

C2CA  8508                      STA     WARMST          ;update warmstart flag

                         ;      Set initial conditions.

C2CC  78                        SEI
C2CD  D8                        CLD
C2CE  A2FF                      LDX     #$FF
C2D0  9A                        TXS                     ;set stack pointer

                         ;      Initialize LNBUG flag, if necessary.

         = 0000          LNBUG  IF      LNBUG
                         LNBUG  ENDIF

                         ;      Perform miscellaneous initialization.
```

```
C2D1  2071C4                JSR    PMI             ;perform miscellaneous init:

                      ;     Initialize memory status.

C2D4  A901                  LDA    #1              ;no failure indicator
C2D6  8501                  STA    NGFLAG          ;memory status flag

                      ;     Check type.

C2D8  A508                  LDA    WARMST          ;warmstart flag
C2DA  D052 ^C32E            BNE    PRS8            ;if warmstart

                      ;     Zero all RAM (except beginning of page zero).

C2DC  A900                  LDA    #0
C2DE  A008                  LDY    #WARMST         ;initial offset into page z:
C2E0  8504                  STA    RAMLO
C2E2  8505                  STA    RAMLO+1         ;initialize RAM pointer

C2E4  A9FF            PRS3  LDA    #$FF
C2E6  9104                  STA    (RAMLO),Y       ;attempt to store $FF
C2E8  D104                  CMP    (RAMLO),Y
C2EA  F002 ^C2EE            BEQ    PRS4            ;if $FF stored successfully

C2EC  4601                  LSR    NGFLAG          ;indicate memory failure

C2EE  A900            PRS4  LDA    #$00
C2F0  9104                  STA    (RAMLO),Y       ;attempt to store $00
C2F2  D104                  CMP    (RAMLO),Y
C2F4  F002 ^C2F8            BEQ    PRS5            ;if $00 stored successfully

C2F6  4601                  LSR    NGFLAG          ;indicate memory failure

C2F8  C8              PRS5  INY
C2F9  D0E9 ^C2E4            BNE    PRS3            ;if not end of page

                      ;     Advance to next page and check for completion.

C2FB  E605                  INC    RAMLO+1         ;advance RAM pointer to next
C2FD  A605                  LDX    RAMLO+1
C2FF  E406                  CPX    TRAMSZ          ;RAM size
C301  D0E1 ^C2E4            BNE    PRS3            ;if not at end of RAM

                      ;     Initialize DOSVEC.

C303  A923                  LDA    #low PPD        ;power-up display routine a:
C305  850A                  STA    DOSVEC          ;initialize DOS vector
C307  A9F2                  LDA    #high PPD
C309  850B                  STA    DOSVEC+1

                      ;     Verify ROM checksums.

C30B  AD01D3                LDA    PORTB
C30E  297F                  AND    #$7F            ;select self-test ROM
C310  8D01D3                STA    PORTB           ;port B memory control

C313  2073FF                JSR    VFR             ;verify first 8K ROM
```

```
C316  B005 ^C31D           BCS     PRS6             ;if first 8K ROM bad

C318  2092FF               JSR     VSR              ;verify second 8K ROM
C31B  9002 ^C31F           BCC     PRS7             ;if seond 8K ROM good

C31D  4601         PRS6    LSR     NGFLAG           ;indicate memory bad

C31F  AD01D3       PRS7    LDA     PORTB
C322  0980                 ORA     #$80             ;disable self-test ROM
C324  8D01D3               STA     PORTB            ;update port B memory contr:

                   ;       Indicate coldstart in progress.

C327  A9FF                 LDA     #$FF
C329  8D4402               STA     COLDST           ;indicate coldstart in prog:
C32C  D022 ^C350           BNE     PRS12            ;continue with coldstart pr:

                   ;       Perform warmstart procedures.

C32E  A200         PRS8    LDX     #0

C330  ADEC03               LDA     DERRF            ;screen OPEN error flag
C333  F007 ^C33C           BEQ     PRS9             ;if in screen OPEN

                   ;       Clean up APPMHI.

                   ;       STX     APPMHI
C335  8E0E00               VFD     8\$8E,8\low APPMHI,8\high APPMHI
                   ;       STX     APPMHI+1
C338  8E0F00               VFD     8\$8E,8\low [APPMHI+1],8\high [APPMHI+1]

C33B  8A                   TXA

                   ;       Clear page 2 and part of page 3.

C33C  9D0002       PRS9    STA     $0200,X          ;clear byte of page 2

C33F  E0ED                 CPX     #low ACMVAR      ;start of page 3 locations :
C341  B003 ^C346           BCS     PRS10            ;if not to clear this page :

C343  9D0003               STA     $0300,X          ;clear byte of page 3

C346  CA           PRS10   DEX
C347  D0F3 ^C33C           BNE     PRS9             ;if not done

                   ;       Clear part of page 0.

C349  A210                 LDX     #INTZBS          ;offset to first page 0 byt:

C34B  9500         PRS11   STA     $0000,X          ;clear byte of page 0
C34D  E8                   INX
C34E  10FB ^C34B           BPL     PRS11            ;if not done

                   ;       Record BASIC status.

C350  A200         PRS12   LDX     #0               ;initially assume BASIC ena:
C352  AD01D3               LDA     PORTB            ;port B memory control
```

```
C355  2902                 AND    #$02             ;BASIC enabled indicator
C357  F001 ^C35A           BEQ    PRS13            ;if BASIC enabled

C359  E8                   INX                     ;indicate BASIC disabled

C35A  8EF803       PRS13   STX    BASICF           ;BASIC flag

                   ;       Establish power-up validation bytes.

C35D  A95C                 LDA    #PUPVL1
C35F  8D3D03               STA    PUPBT1           ;validation byte 1
C362  A993                 LDA    #PUPVL2
C364  8D3E03               STA    PUPBT2           ;validation byte 2
C367  A925                 LDA    #PUPVL3
C369  8D3F03               STA    PUPBT3           ;validation byte 3

                   ;       Establish screen margins.

C36C  A902                 LDA    #LEDGE
C36E  8552                 STA    LMARGN           ;left margin
C370  A927                 LDA    #REDGE
C372  8553                 STA    RMARGN           ;right margin

                   ;       Establish parameters for NTSC or PAL.

C374  AD14D0               LDA    PAL              ;GTIA flag bits
C377  290E                 AND    #$0E             ;PAL/NTSC indicator
C379  D008 ^C383           BNE    PRS14            ;if NTSC

C37B  A905                 LDA    #5               ;PAL key repeat delay
C37D  A201                 LDX    #1               ;PAL indicator
C37F  A028                 LDY    #40              ;PAL key repeat initial del;
C381  D006 ^C389           BNE    PRS15            ;set parameters

C383  A906         PRS14   LDA    #6               ;NTSC key repeat delay
C385  A200                 LDX    #0               ;NTSC indicator
C387  A030                 LDY    #48              ;NTSC key repeat initial de;

C389  8DDA02       PRS15   STA    KEYREP           ;set key repeat rate
C38C  8662                 STX    PALNTS           ;set PAL/NTSC status
C38E  8CD902               STY    KRPDEL           ;set key repeat initial del;

                   ;       Initialize missing controller ports, if not simulat;

      = 0000       VGC     IF     not VGC
                   VGC     ENDIF

                   ;       Copy interrupt vector table from ROM to RAM.

C391  A225                 LDX    #TIHVL-1         ;offset to last byte of tab;

C393  BD48C4       PRS17   LDA    TIHV,X           ;byte of table of interrupt;
C396  9D0002               STA    INTABS,X         ;byte of RAM table
C399  CA                   DEX
C39A  10F7 ^C393           BPL    PRS17            ;if not done

                   ;       Copy handler vector table from ROM to RAM.
```

```
C39C  A20E                LDX     #THAVL-1          ;offset to last byte of tab:

C39E  BD2EC4      PRS18   LDA     THAV,X            ;byte of handler vector tab:
C3A1  9D1A03              STA     HATABS,X          ;byte of RAM table
C3A4  CA                  DEX
C3A5  10F7 ^C39E          BPL     PRS18             ;if not done

                  ;       Initialize software.

C3A7  2035C5              JSR     ISW               ;initialize software

                  ;       Initialize ACMI module, if present.

      = 0000      ACMI    IF      ACMI
                  ACMI    ENDIF

                  ;       Enable IRQ interrupts.

C3AA  58                  CLI

                  ;       Check for memory problems.

C3AB  A501                LDA     NGFLAG            ;memory status
C3AD  D015 ^C3C4          BNE     PRS21             ;if memory good

                  ;       Perform memory self-test on bad memory.

C3AF  AD01D3              LDA     PORTB
C3B2  297F                AND     #$7F              ;enable self-test ROM
C3B4  8D01D3              STA     PORTB             ;update port B memory contr:
C3B7  A902                LDA     #2
C3B9  8DF302              STA     CHACT             ;CHACTL (character control):
C3BC  A9E0                LDA     #high DCSORG      ;high domestic character se:
C3BE  8DF402              STA     CHBAS             ;CHBASE (character base) sh:
C3C1  4C0350              JMP     EMS               ;execute memory self-test

                  ;       Check for cartridge.

C3C4  A200        PRS21   LDX     #0
C3C6  8606                STX     TRAMSZ            ;clear cartridge flag

C3C8  AEE402              LDX     RAMSIZ            ;RAM size
C3CB  E0B0                CPX     #high $B000       ;start of cartridge area
C3CD  B00D ^C3DC          BCS     PRS22             ;if RAM in cartridge area

C3CF  AEFCBF              LDX     CART
C3D2  D008 ^C3DC          BNE     PRS22             ;if no cartridge

C3D4  E606                INC     TRAMSZ            ;set cartridge flag
C3D6  20C9C4              JSR     CCE               ;check cartridge equivalenc:
C3D9  2029C4              JSR     ICS               ;initialize cartridge softw:

                  ;       Open screen editor.

C3DC  A903        PRS22   LDA     #OPEN
C3DE  A200                LDX     #SEIOCB           ;screen editor IOCB index
```

```
C3E0  9D4203               STA     ICCOM,X          ;command
C3E3  A948                 LDA     #low SEDS        ;screen editor device speci:
C3E5  9D4403               STA     ICBAL,X          ;buffer address
C3E8  A9C4                 LDA     #high SEDS
C3EA  9D4503               STA     ICBAH,X
C3ED  A90C                 LDA     #OPNIN+OPNOT     ;open for input/output
C3EF  9D4A03               STA     ICAX1,X          ;auxiliary informatin 1
C3F2  2056E4               JSR     CIOV             ;vector to CIO
C3F5  1003 ^C3FA           BPL     PRS23            ;if no error

                   ;       Process error (which should never happen).

C3F7  4CAAC2               JMP     RES              ;retry power-up

                   ;       Delay, ensuring VBLANK.

C3FA  E8           PRS23   INX
C3FB  D0FD ^C3FA           BNE     PRS23            ;if inner loop not done

C3FD  C8                   INY
C3FE  10FA ^C3FA           BPL     PRS23            ;if outer loop not done

                   ;       Attempt cassette boot.

C400  206EC6               JSR     ACB              ;attempt cassette boot

                   ;       Check cartridge for disk boot.

C403  A506                 LDA     TRAMSZ
C405  F006 ^C40D           BEQ     PRS24            ;if no cartridge

C407  ADFDBF               LDA     CARTFG           ;cartridge mode flags
C40A  6A                   ROR     A
C40B  9006 ^C413           BCC     PRS25            ;if disk boot not desired

                   ;       Attempt disk boot.

C40D  208BC5       PRS24   JSR     ADB              ;attempt disk boot

                   ;       Initialize periperhal handler loading facility.

C410  2039E7               JSR     PHR              ;poll, load, relocate, init:

                   ;       Indicate coldstart complete.

C413  A900         PRS25   LDA     #0
C415  8D4402               STA     COLDST           ;indicate coldstart complet:

                   ;       Check cartridge for execution.

C418  A506                 LDA     TRAMSZ
C41A  F00A ^C426           BEQ     PRS26            ;if no cartridge

C41C  ADFDBF               LDA     CARTFG           ;cartridge mode flags
C41F  2904                 AND     #$04
C421  F003 ^C426           BEQ     PRS26            ;if execution not desired
```

```
                        ;       Execute cartridge.

 C423   6CFABF                  JMP     (CARTCS)        ;execute cartridge

                        ;       Exit to power-up display or booted program.

 C426   6C0A00          PRS26   JMP     (DOSVEC)        ;vector to booted program



                        **      ICS - Initialize Cartridge Software
                        *
                        *       ENTRY   JSR     ICS
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = C429         ICS     =       *               ;entry
 C429   6CFEBF                  JMP     (CARTAD)        ;initialize cartridge softw:



                        **      PAI - Process ACMI Interrupt
                        *
                        *       PAI does nothing.
                        *
                        *       ENTRY   JSR     PAI
                        *
                        *       NOTES
                        *               Problem: this code is unneeded unless ACMI :
                        *               option is selected.
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = C42C         PAI     =       *               ;entry
 C42C   18                      CLC
 C42D   60                      RTS                     ;return
```

```
                        **      THAV - Table of Handler Vectors
                        *
                        *       NOTES
                        *               THAV is moved to RAM table HATABS.

C42E  50                THAV    DB      PRINTR  ;printer device code
C42F  30E4                      DW      PRINTV  ;printer handler vector table

C431  43                        DB      CASSET  ;cassette device code
C432  40E4                      DW      CASETV  ;cassette handler vector table

C434  45                        DB      SCREDT  ;editor device code
C435  00E4                      DW      EDITRV  ;editor handler vector table

C437  53                        DB      DISPLY  ;screen device code
C438  10E4                      DW      SCRENV  ;screen handler vector table

C43A  4B                        DB      KBD     ;keyboard device code
C43B  20E4                      DW      KEYBDV  ;keyboard handler vector table

      = 000F            THAVL   =       *-THAV  ;length


                        **      BMSG - Boot Error Message

C43D  424F4F5420        BMSG    DB      'BOOT ERROR',EOL


                        **      Screen Editor Device Specification

C448  453A9B            SEDS    DB      'E:',EOL


                        **      TIHV - Table of Interrupt Handler Vectors
                        *
                        *       NOTES
                        *               TIHV is moved to RAM table INTABS.

C44B  CEC0              TIHV    DW      RIR     ;VDSLST - display list NMI vector
C44D  CDC0                      DW      XIR     ;VPRCED - proceed line IRQ vector
C44F  CDC0                      DW      XIR     ;VINTER - interrupt line IRQ vector
C451  CDC0                      DW      XIR     ;VBREAK - BRK instruction IRQ vecto
C453  19FC                      DW      KIR     ;VKEYBD - keyboard IRQ vector
C455  2CEB                      DW      IRIR    ;VSERIN - serial input ready IRQ ve
C457  ADEA                      DW      ORIR    ;VSEROR - serial output ready IRQ v
C459  ECEA                      DW      OCIR    ;VSEROC - serial output complete IR
C45B  CDC0                      DW      XIR     ;VTIMR1 - POKEY timer 1 IRQ vector
```

```
C45D  C0C0              DW      XIR     ;VTIMR2 - POKEY timer 2 IRQ vector
C45F  C0C0              DW      XIR     ;VTIMR4 - POKEY timer 4 IRQ vector
C461  30C0              DW      IIR     ;VIMIRQ - immediate IRQ vector
C463  0000              DW      0       ;CDTMV1 - countdown timer 1 vector
C465  0000              DW      0       ;CDTMV2 - countdown timer 2 vector
C467  0000              DW      0       ;CDTMV3 - countdown timer 3 vector
C469  0000              DW      0       ;CDTMV4 - countdown timer 4 vector
C46B  0000              DW      0       ;CDTMV5 - countdown timer 5 vector
C46D  E2C0              DW      IVNM    ;VVBLKI - immediate VBLANK NMI vect:
C46F  8AC2              DW      DVNM    ;VVBLKD - deferred VBLANK NMI vecto:

      = 0026    TIHVL   =       *-TIHV  ;length



                **      PMI - Perform Miscellaneous Initialization
                *
                *       ENTRY   JSR     PMI
                *
                *       NOTES
                *               Problem: initial address for sizing RAM sho:
                *               $4000 (16K) instead of $2800.
                *
                *       MODS
                *               Original Author Unknown
                *               1. Bring closer to Coding Standard (object :
                *                  R. K. Nordin 11/01/83


      = C471    PMI     =       *       ;entry

                ;       Check for cartridge special execution case.

C471  AD13D0            LDA     TRIG3
C474  6A                ROR     A
C475  900D ^C484        BCC     PMI1    ;if cartridge not inserted

C477  ADFCBF            LDA     CART
C47A  D008 ^C484        BNE     PMI1    ;if not cartridge

C47C  ADFDBF            LDA     CARTFG  ;cartridge flags
C47F  1003 ^C484        BPL     PMI1    ;if special execution not desired

                ;       Execute cartridge.

C481  6CFEBF            JMP     (CARTAD)        ;execute cartridge

                ;       Initialize hardware.

C484  20DAC4    PMI1    JSR     IHW     ;initialize hardware

                ;       Disable BASIC.

C487  AD01D3            LDA     PORTB
C48A  0902              ORA     #$02    ;disable BASIC
C48C  8D01D3            STA     PORTB   ;update port B memory control
```

```
                        ;       If warmstart, check previous BASIC status.

C48F  A508                      LDA     WARMST
C491  F007 ^C49A                BEQ     PMI2    ;if coldstart

C493  ADF803                    LDA     BASICF  ;BASIC flag
C496  D011 ^C4A9                BNE     PMI4    ;if BASIC not previously enabled

C498  F007 ^C4A1                BEQ     PMI3    ;enable BASIC

                        ;       Check OPTION key.

C49A  AD1FD0            PMI2    LDA     CONSOL  ;console switches
C49D  2904                      AND     #$04    ;OPTION key indicator
C49F  F008 ^C4A9                BEQ     PMI4    ;if OPTION key pressed, do not enab:

                        ;       Enable BASIC.

C4A1  AD01D3            PMI3    LDA     PORTB
C4A4  29FD                      AND     #$FD    ;disable BASIC
C4A6  8D01D3                    STA     PORTB   ;update port B memory control

                        ;       Determine size of RAM.

      = 0000            RAMSYS  IF      RAMSYS
                        RAMSYS  ELSE

C4A9  A900              PMI4    LDA     #low $2800      ;initial low address
C4AB  A8                        TAY                     ;offset to first byte of pa:
C4AC  8505                      STA     TRAMSZ-1        ;set initial low address

C4AE  A928                      LDA     #high $2800     ;initial RAM size
C4B0  8506                      STA     TRAMSZ          ;set initial RAM size (high:

C4B2  B105              PMI5    LDA     (TRAMSZ-1),Y    ;first byte of page
C4B4  49FF                      EOR     #$FF            ;complement
C4B6  9105                      STA     (TRAMSZ-1),Y    ;attempt to store complemen:
C4B8  D105                      CMP     (TRAMSZ-1),Y
C4BA  D00C ^C4C8                BNE     PMI6            ;if complement not stored

C4BC  49FF                      EOR     #$FF            ;original value
C4BE  9105                      STA     (TRAMSZ-1),Y    ;attempt to store original :
C4C0  D105                      CMP     (TRAMSZ-1),Y
C4C2  D004 ^C4C8                BNE     PMI6            ;if original value not stor:

C4C4  E606                      INC     TRAMSZ          ;increment high address
C4C6  D0EA ^C4B2                BNE     PMI5            ;continue

                        ;       Exit.

C4C8  60                PMI6    RTS                     ;return
                        RAMSYS  ENDIF
                        **      CCE - Check Cartridge Equivalence
                        *
                        *       ENTRY   JSR     CCE
                        *
                        *       NOTES
                        *               Problem: this code checksums $BFF0 - $C0EF;:
                        *               checksum $BF00 - $BFFF.
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C4C9          CCE     =       *       ;entry

                        ;       Initialize.

 C4C9  A900                     LDA     #0      ;initial sum
 C4CB  AA                       TAX             ;offset to first byte
 C4CC  18                       CLC

                        ;       Checksum 256 bytes of cartridge area.

 C4CD  7DF0BF           CCE1    ADC     $BFF0,X ;add in byte
 C4D0  E8                       INX
 C4D1  D0FA ^C4CD               BNE     CCE1    ;if not done

                        ;       Exit.

 C4D3  CDEB03                   CMP     CARTCK  ;previous checksum
 C4D6  8DEB03                   STA     CARTCK  ;new checksum
 C4D9  60                       RTS             ;return


                        **      IHW - Initialize Hardware
                        *
                        *       ENTRY   JSR     IHW
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C4DA          IHW     =       *       ;entry

                        ;       Initialize CTIA, ANTIC and POKEY areas.

 C4DA  A900                     LDA     #0              ;initialization value
 C4DC  AA                       TAX                     ;initial offset
 C4DD  8D03D3                   STA     PBCTL           ;set for direction register:

 C4E0  9D00D0           IHW1    STA     CTIA,X          ;initialize CTIA/GTIA area :
 C4E3  9D00D4                   STA     ANTIC,X         ;initialize ANTIC area regi:
```

```
C4E6  9D00D2                STA    POKEY,X       ;initialize POKEY area regi:
C4E9  E001                  CPX    #low PORTB
C4EB  F003 ^C4F0            BEQ    IHW2          ;if port B, don't initializ:

C4ED  9D00D3                STA    PIA,X         ;initialize PIA area regist:

C4F0  E8           IHW2     INX
C4F1  D0ED ^C4E0            BNE    IHW1          ;if not done

                   ;        Initialize PIA.

C4F3  A93C                  LDA    #$3C
C4F5  8D03D3                STA    PBCTL   ;precondition port B outputs
C4F8  A9FF                  LDA    #$FF
C4FA  8D01D3                STA    PORTB   ;all high
C4FD  A938                  LDA    #$38
C4FF  8D02D3                STA    PACTL   ;select data direction register
C502  8D03D3                STA    PBCTL   ;select data direction register
C505  A900                  LDA    #$00
C507  8D00D3                STA    PORTA   ;all inputs
C50A  A9FF                  LDA    #$FF
C50C  8D01D3                STA    PORTB   ;all outputs
C50F  A93C                  LDA    #$3C
C511  8D02D3                STA    PACTL   ;back to port
C514  8D03D3                STA    PBCTL   ;back to port
C517  AD01D3                LDA    PORTB   ;clear interrupts
C51A  AD00D3                LDA    PORTA   ;clear interrupts

                   ;        Initialize POKEY.

C51D  A922                  LDA    #$22    ;get POKEY out of initialize mode a:
C51F  8D0FD2                STA    SKCTL   ;set serial port control

C522  A9A0                  LDA    #$A0    ;pure tone, no volume
C524  8D05D2                STA    AUDC3   ;turn off channel 3
C527  8D07D2                STA    AUDC4   ;turn off channel 4

C52A  A928                  LDA    #$28    ;clock ch. 3 with 1.79 MHz, ch. 4 w:
C52C  8D08D2                STA    AUDCTL  ;set audio control

C52F  A9FF                  LDA    #$FF
C531  8D0DD2                STA    SEROUT  ;start bit only

C534  60                    RTS            ;return
```

```
                        **      ISW - Initialize Software
                        *
                        *       ENTRY   JSR     ISW
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = C535          ISW     =       *               ;entry

                        ;       Initialize BREAK key handling.

C535  C611                      DEC     BRKKEY          ;turn off BREAK key flag

C537  A992                      LDA     #low BIR
C539  8D3602                    STA     BRKKY           ;set BREAK key IRQ routine :
C53C  A9C0                      LDA     #high BIR
C53E  8D3702                    STA     BRKKY+1

                        ;       Initialize RAMSIZ and MEMTOP.

C541  A506                      LDA     TRAMSZ          ;determined size of RAM
C543  8DE402                    STA     RAMSIZ          ;size of RAM
C546  8DE602                    STA     MEMTOP+1        ;high top of memory
C549  A900                      LDA     #$00
C54B  8DE502                    STA     MEMTOP          ;low top of memory

                        ;       Initialize MEMLO.

C54E  A900                      LDA     #low INIML      ;initial MEMLO address
C550  8DE702                    STA     MEMLO
C553  A907                      LDA     #high INIML
C555  8DE802                    STA     MEMLO+1

                        ;       Initialize device handlers.

C558  200CE4                    JSR     EDITRV+12       ;initialize editor handler
C55B  201CE4                    JSR     SCRENV+12       ;initialize screen handler
C55E  202CE4                    JSR     KEYBDV+12       ;initialize keyboard handle:
C561  203CE4                    JSR     PRINTV+12       ;initialize printer handler
C564  204CE4                    JSR     CASETV+12       ;initialize cassette handle:

                        ;       Initialize various routines.

C567  206EE4                    JSR     CIOINV          ;initialize CIO
C56A  2065E4                    JSR     SIOINV          ;initialize SIO
C56D  206BE4                    JSR     INTINV          ;initialize interrupt handl:
C570  2050E4                    JSR     DINITV          ;initialize DIO

                        ;       Initialize generic parallel device handler.

C573  A96E                      LDA     #low PIR
C575  8D3802                    STA     VPIRQ           ;parallel device IRQ routin:
C578  A9C9                      LDA     #high PIR
C57A  8D3902                    STA     VPIRQ+1
```

```
C57D  209BE4               JSR     GPDVV+12         ;initialize parallel device:

                    ;      Set status of START key.

C580  AD1FD0               LDA     CONSOL           ;console switches
C583  2901                 AND     #$01             ;START key indicator
C585  4901                 EOR     #$01             ;START key status
C587  8DE903               STA     CKEY             ;cassette boot request flag

C58A  60                   RTS                      ;return



                    **     ADB - Attempt Disk Boot
                    *
                    *      ENTRY   JSR     ADB
                    *
                    *      MODS
                    *              Original Author Unknown
                    *              1. Bring closer to Coding Standard (object :
                    *                 R. K. Nordin 11/01/83


         = C58B     ADB    =       *       ;entry

                    ;      Check type of reset.

C58B  A508                 LDA     WARMST
C58D  F009 ^C598           BEQ     ADB1    ;if not warmstart

                    ;      Process warmstart.

C58F  A509                 LDA     BOOT?   ;successful boot flags
C591  2901                 AND     #$01    ;successful disk boot indicator
C593  F033 ^C5C8           BEQ     BAI2    ;if disk boot not successful, retur:

                    ;      Initialize disk booted software.

C595  4C3BC6               JMP     IBS     ;initialize booted software

                    ;      Process coldstart.

C598  A901          ADB1   LDA     #1
C59A  8D0103               STA     DUNIT   ;disk unit number
C59D  A953                 LDA     #STATC  ;status
C59F  8D0203               STA     DCOMND  ;command
C5A2  2053E4               JSR     DSKINV  ;issue command
C5A5  3021 ^C5C8           BMI     BAI2    ;if error, return

                    ;      Boot.

                    ;      JMP     ABI     ;attempt boot and initialize
```

```
                          **      ABI - Attempt Boot and Initialize
                          *
                          *       ENTRY   JSR     ABI
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

       = C5A7             ABI     =       *                ;entry

 C5A7  A900                       LDA     #high 1
 C5A9  8D0B03                     STA     DAUX2
 C5AC  A901                       LDA     #low 1           ;sector number
 C5AE  8D0A03                     STA     DAUX1

 C5B1  A900                       LDA     #low [CASBUF+3]  ;buffer address
 C5B3  8D0403                     STA     DBUFLO
 C5B6  A904                       LDA     #high [CASBUF+3]
 C5B8  8D0503                     STA     DBUFHI

                          ;       JMP     BAI              ;boot and initialize


                          **      BAI - Boot and Initialize
                          *
                          *       ENTRY   JSR     BAI
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

       = C5BB             BAI     =       *          ;entry

                          ;       Read first sector.

 C5BB  2059C6                     JSR     GNS        ;get next sector
 C5BE  1009 ^C5C9                 BPL     CBI        ;if no error, complete boot and ini:

                          ;       Process error.

 C5C0  203EC6             BAI1    JSR     DBE        ;display boot error message

 C5C3  ADEA03                     LDA     CASSBT
 C5C6  F0DF ^C5A7                 BEQ     ABI        ;if not cassette boot, try again

                          ;       Exit.

 C5C8  60                 BAI2    RTS                ;return
```

```
                        **      CBI - Complete Boot and Initialize
                        *
                        *       ENTRY   JSR     CBI
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

       = C5C9           CBI     =       *               ;entry

                        ;       Transfer flags.

C5C9   A203                     LDX     #3

C5CB   BD0004           CBI1    LDA     CASBUF+3,X      ;byte from buffer
C5CE   9D4002                   STA     DFLAGS,X        ;flag byte
C5D1   CA                       DEX
C5D2   10F7 ^C5CB               BPL     CBI1            ;if not done

                        ;       Transfer sector.

C5D4   AD4202                   LDA     BOOTAD
C5D7   8504                     STA     RAMLO           ;set boot address
C5D9   AD4302                   LDA     BOOTAD+1
C5DC   8505                     STA     RAMLO+1

C5DE   AD0404                   LDA     CASBUF+7
C5E1   850C                     STA     DOSINI          ;establish initializtion ad:
C5E3   AD0504                   LDA     CASBUF+8
C5E6   850D                     STA     DOSINI+1

C5E8   A07F             CBI2    LDY     #127            ;offset to last byte of sec:

C5EA   B90004           CBI3    LDA     CASBUF+3,Y      ;byte of sector buffer
C5ED   9104                     STA     (RAMLO),Y       ;byte of boot program
C5EF   88                       DEY
C5F0   10F8 ^C5EA               BPL     CBI3            ;if not done

                        ;       Increment loader buffer pointer.

C5F2   18                       CLC
C5F3   A504                     LDA     RAMLO
C5F5   6980                     ADC     #$80
C5F7   8504                     STA     RAMLO
C5F9   A505                     LDA     RAMLO+1
C5FB   6900                     ADC     #0
C5FD   8505                     STA     RAMLO+1         ;increment boot loader buff:

                        ;       Decrement and check number of sectors.

C5FF   CE4102                   DEC     DBSECT          ;decrement number of sector:
C602   F012 ^C616               BEQ     CBI5            ;if no more sectors

                        ;       Get next sector.
```

```
C604  EE0A03                  INC     DAUX1             ;increment sector number

C607  2059C6          CBI4    JSR     GNS               ;get next sector
C60A  10DC ^C5E8              BPL     CBI2              ;if status OK

                      ;       Process error.

C60C  203EC6                  JSR     DBE               ;display boot error message
C60F  ADEA03                  LDA     CASSBT
C612  D0AC ^C5C0              BNE     BAI1              ;if cassette, start over

C614  F0F1 ^C607              BEQ     CBI4              ;try sector again

                      ;       Clean up.

C616  ADEA03          CBI5    LDA     CASSBT
C619  F003 ^C61E              BEQ     CBI6              ;if not cassette boot

C61B  2059C6                  JSR     GNS               ;get EOF record (but do not:

                      ;       Execute boot loader.

C61E  2029C6          CBI6    JSR     EBL               ;execute boot loader
C621  B09D ^C5C0              BCS     BAI1              ;if bad boot, try again

                      ;       Initialize booted software.

C623  203BC6                  JSR     IBS               ;initialize booted software
C626  E609                    INC     BOOT?             ;indicate boot success
C628  60                      RTS                       ;return



                      **      EBL - Execute Boot Loader
                      *
                      *       ENTRY   JSR     EBL
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = C629          EBL     =       *                 ;entry

                      ;       Move boot loader start address to RAMLO.

C629  18                      CLC
C62A  AD4202                  LDA     BOOTAD
C62D  6906                    ADC     #6
C62F  8504                    STA     RAMLO             ;boot loader start address
C631  AD4302                  LDA     BOOTAD+1
C634  6900                    ADC     #0
C636  8505                    STA     RAMLO+1

                      ;       Execute boot loader.
```

```
C638  6C0400                 JMP     (RAMLO)          ;execute boot loader


                   **        IBS - Initialize Booted Software
                   *
                   *         ENTRY   JSR      IBS
                   *
                   *         MODS
                   *                 Original Author Unknown
                   *                 1. Bring closer to Coding Standard (object :
                   *                    R. K. Nordin 11/01/83


       = C638      IBS       =       *                ;entry
C638  6C0C00                 JMP     (DOSINI)         ;initialize booted software


                   **        DBE - Display Boot Error Message
                   *
                   *         ENTRY   JSR      DBE
                   *
                   *         NOTES
                   *                 Problem: bytes wasted by LDX/TXA and LDY/TY:
                   *                 combinations.
                   *
                   *         MODS
                   *                 Original Author Unknown
                   *                 1. Bring closer to Coding Standard (object :
                   *                    R. K. Nordin 11/01/83


       = C63E      DBE       =       *                ;entry

                   ;         Set up IOCB.

C63E  A23D                   LDX     #low BMSG        ;boot error message
C640  A0C4                   LDY     #high BMSG
C642  8A                     TXA
C643  A200                   LDX     #SEIOCB          ;screen editor IOCB index
C645  9D4403                 STA     ICBAL,X          ;low buffer address
C648  98                     TYA
C649  9D4503                 STA     ICBAH,X          ;high buffer address
C64C  A909                   LDA     #PUTREC
C64E  9D4203                 STA     ICCOM,X          ;command
C651  A9FF                   LDA     #$FF
C653  9D4803                 STA     ICBLL,X          ;buffer length

                   ;         Perform CIO.

C656  4C56E4                 JMP     CIOV             ;vector to CIO, return
```

```
                          **      GNS - Get Next Sector
                          *
                          *       ENTRY   JSR     GNS
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


      = C659              GNS     =       *       ;entry

                          ;       Check type of boot.

C659  ADEA03                      LDA     CASSBT
C65C  F003 ^C661                  BEQ     GNS1    ;if not cassette boot

                          ;       Read block from cassette.

C65E  4C7AE4                      JMP     RBLOKV  ;vector to read cassette block rout:

                          ;       Read sector from disk.

C661  A952                GNS1    LDA     #READ
C663  8D0203                      STA     DCOMND  ;command
C666  A901                        LDA     #1      ;drive number 1
C668  8D0103                      STA     DUNIT   ;set drive number
C66B  4C53E4                      JMP     DSKINV  ;vector to DIO, return



                          **      ACB - Attempt Cassette Boot
                          *
                          *       ENTRY   JSR     ACB
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


      = C66E              ACB     =       *               ;entry

                          ;       Check type.

C66E  A508                        LDA     WARMST          ;warmstart flag
C670  F009 ^C67B                  BEQ     ACB1            ;if coldstart

                          ;       Perform warmstart procedures.

C672  A509                        LDA     BOOT?           ;successful boot flags
C674  2902                        AND     #$02            ;successful cassette boot f:
C676  F027 ^C69F                  BEQ     ACB2            ;if cassette boot not succe:

C678  4CA0C6                      JMP     ACB3            ;initialize cassette
```

```
                        ;       Perform coldstart procedures.

C67B  ADE903      ACB1  LDA     CKEY            ;cassette boot request flag
C67E  F01F ^C69F        BEQ     ACB2            ;if cassette boot not reque:

                        ;       Boot cassette.

C680  A980              LDA     #$80
C682  853E              STA     FTYPE           ;set long IRG type
C684  EEEA03            INC     CASSBT          ;set cassette boot flag
C687  207DE4            JSR     CSOPIV          ;open cassette for input
C68A  208BC5            JSR     BAI             ;boot and initialize
C68D  A900              LDA     #0
C68F  8DEA03            STA     CASSBT          ;clear cassette boot flag
C692  8DE903            STA     CKEY            ;clear cassette boot reques:
C695  0609              ASL     BOOT?           ;indicate successful casset:

C697  A50C              LDA     DOSINI
C699  8502              STA     CASINI          ;cassette initialization ad:
C69B  A50D              LDA     DOSINI+1
C69D  8503              STA     CASINI+1

                        ;       Exit.

C69F  60          ACB2  RTS                     ;return

                        ;       Initialize cassette booted program.

C6A0  6C0200      ACB3  JMP     (CASINI)        ;initialize cassette booted:
```

```
C6A3                  **      IDIO - Initialize DIO
                      *
                      *       ENTRY   JSR     IDIO
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83

       = C6A3         IDIO    =       *               ;entry
C6A3   A9A0                   LDA     #160            ;160 second timeout
C6A5   8D4602                 STA     DSKTIM          ;set initial disk timeout
C6A8   A980                   LDA     #low DSCTSZ     ;disk sector size
C6AA   8DD502                 STA     DSCTLN
C6AD   A900                   LDA     #high DSCTSZ
C6AF   8DD602                 STA     DSCTLN+1
C6B2   60                     RTS                     ;return


                      **      DIO - Disk I/O
                      *
                      *       ENTRY   JSR     DIO
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83

       = C6B3         DIO     =       *               ;entry

                      ;       Initialize.

C6B3   A931                   LDA     #DISKID         ;disk bus ID
C6B5   8D0003                 STA     DDEVIC          ;device bus ID
C6B8   AD4602                 LDA     DSKTIM          ;timeout
C6BB   AE0203                 LDX     DCOMND          ;command
C6BE   E021                   CPX     #FOMAT
C6C0   F002 ^C6C4             BEQ     DIO1            ;if FORMAT command

C6C2   A907                   LDA     #7              ;set timeout to 7 seconds

C6C4   8D0603         DIO1    STA     DTIMLO          ;timeout

                      ;       Set SIO command.

C6C7   A240                   LDX     #GETDAT         ;assume GET DATA

C6C9   AD0203                 LDA     DCOMND          ;command
C6CC   C950                   CMP     #PUTSEC
C6CE   F004 ^C6D4             BEQ     DIO2            ;if PUT SECTOR command

C6D0   C957                   CMP     #WRITE
C6D2   D002 ^C6D6             BNE     DIO3            ;if not WRITE command
```

```
C6D4  A280          DIO2    LDX     #PUTDAT         ;select PUT DATA

                    ;       Check command.

C6D6  C953          DIO3    CMP     #STATC
C6D8  D010 ^C6EA            BNE     DIO4            ;if not STATUS command

                    ;       Set up STATUS command.

C6DA  A9EA                  LDA     #low DVSTAT
C6DC  8D0403                STA     DBUFLO          ;buffer address
C6DF  A902                  LDA     #high DVSTAT
C6E1  8D0503                STA     DBUFHI
C6E4  A004                  LDY     #low 4          ;low byte count
C6E6  A900                  LDA     #high 4         ;high byte count
C6E8  F006 ^C6F0            BEQ     DIO5            ;perform SIO

                    ;       Set up other commands.

C6EA  ACD502        DIO4    LDY     DSCTLN          ;low byte count
C6ED  ADD602                LDA     DSCTLN+1        ;high byte count

                    ;       Perform SIO.

C6F0  8E0303        DIO5    STX     DSTATS          ;SIO command
C6F3  8C0803                STY     DBYTLO          ;low byte count
C6F6  8D0903                STA     DBYTHI          ;high byte count
C6F9  2059E4                JSR     SIOV            ;vector to SIO
C6FC  1001 ^C6FF            BPL     DIO6            ;if no error

                    ;       Process error.

C6FE  60                    RTS                     ;return

                    ;       Process successful operation.

C6FF  AD0203        DIO6    LDA     DCOMND          ;command
C702  C953                  CMP     #STATC
C704  D00A ^C710            BNE     DIO7            ;if not STATUS command

C706  203AC7                JSR     SBA             ;set buffer address
C709  A002                  LDY     #2
C70B  B115                  LDA     (BUFADR),Y      ;timeout status
C70D  8D4602                STA     DSKTIM          ;disk timeout

                    ;       Set byte count.

C710  AD0203        DIO7    LDA     DCOMND
C713  C921                  CMP     #FOMAT
C715  D01F ^C736            BNE     DIO10           ;if not FORMAT command

C717  203AC7                JSR     SBA             ;set buffer address
C71A  A0FE                  LDY     #$FE            ;initial buffer pointer

C71C  C8            DIO8    INY                     ;increment buffer pointer
C71D  C8                    INY                     ;increment buffer pointer
```

```
C71E  B115           DI09    LDA    (BUFADR),Y       ;low bad sector data
C720  C9FF                   CMP    #$FF
C722  D0F8 ^C71C             BNE    DI08             ;if low not $FF

C724  C8                     INY
C725  B115                   LDA    (BUFADR),Y       ;high bad sector data
C727  C8                     INY
C728  C9FF                   CMP    #$FF
C72A  D0F2 ^C71E             BNE    DI09             ;if high not $FF

C72C  88                     DEY
C72D  88                     DEY
C72E  8C0803                 STY    DBYTLO           ;low bad sector byte count
C731  A900                   LDA    #0
C733  8D0903                 STA    DBYTHI           ;high bad sector byte count

                     ;       Exit.

C736  AC0303         DI010   LDY    DSTATS           ;status
C739  60                     RTS                     ;return


                     **      SBA - Set Buffer Address
                     *
                     *       ENTRY   JSR     SBA
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83

      = C73A         SBA     =      *                ;entry
C73A  AD0403                 LDA    DBUFLO
C73D  8515                   STA    BUFADR           ;buffer address
C73F  AD0503                 LDA    DBUFHI
C742  8516                   STA    BUFADR+1
C744  60                     RTS                     ;return
```

```
C745                 **      RLR - Relocate Routine
                     *
                     *       RLR reloactes a relocatable routine which is assemb:
                     *       origin 0.
                     *
                     *       ENTRY   JSR     RLR
                     *               GBYTEA - GBYTEA+1 = address of get-byte rou:
                     *
                     *       MODS
                     *               Y. M. Chen      04/01/82
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83

      = C745         RLR     =       *                       ;entry

                     ;       Clear parameter block.

C745  A205                   LDX     #5                      ;offset to last parameter

C747  A900           RLR1    LDA     #0
C749  9DC902                 STA     PARMBL,X                ;clear byte of parameter bl:
C74C  CA                     DEX
C74D  10F8 ^C747             BPL     RLR1                    ;if not done

                     ;       Get a new record type and set the subroutine vector:

C74F  A900           RLR2    LDA     #0
C751  8D3302                 STA     LCOUNT                  ;process 0th byte of a reco:
C754  20CFC7                 JSR     GBY                     ;get type ID
C757  A09C                   LDY     #DATAER
C759  B039 ^C794             BCS     RLR4                    ;if EOF before END record

C75B  8D8802                 STA     HIBYTE                  ;save type ID
C75E  20CFC7                 JSR     GBY                     ;get record length
C761  A09C                   LDY     #DATAER
C763  B02F ^C794             BCS     RLR4                    ;if EOF before END record

C765  8D4502                 STA     RECLEN
C768  AD8802                 LDA     HIBYTE                  ;get type ID
C76B  C90B                   CMP     #$0B                    ;END record
C76D  F026 ^C795             BEQ     END                     ;if END record

C76F  2A                     ROL     A                       ;set subroutine vectors
C770  AA                     TAX
C771  BDE4C8                 LDA     TRPR,X
C774  8DC902                 STA     RUNADR
C777  BDE5C8                 LDA     TRPR+1,X
C77A  8DCA02                 STA     RUNADR+1

C77D  AD4502         RLR3    LDA     RECLEN
C780  CD3302                 CMP     LCOUNT
C783  F0CA ^C74F             BEQ     RLR2                    ;if LCOUNT=RECLEN, get new :

C785  20CFC7                 JSR     GBY                     ;get next byte
C788  A09C                   LDY     #DATAER
C78A  B008 ^C794             BCS     RLR4                    ;if EOF before END record
```

```
C78C  20D2C7               JSR     CAL              ;call record subroutine
C78F  EE3302               INC     LCOUNT
C792  D0E9 ^C77D           BNE     RLR3             ;continue

C794  60           RLR4    RTS                      ;return



                   **      END - Handle END Record
                   *
                   *       END handles record type of
                   *       1.End Record
                   *
                   *       Record format:
                   *       Byte 0          Type ID
                   *       Byte 1          Self-start flag
                   *       Bytes 2 - 3     Run address
                   *
                   *       Process formula
                   *
                   *       RUNADR+LOADAD ==> Start Execution Address n Loader-:
                   *       parameter block.
                   *
                   *       End record calculates the start execution address b:
                   *       RUNADR with LOADAD, and returns to the Caller with :
                   *       block and a status byte in the Y register. Y=1 mean:
                   *       successful, else is a data structure error.
                   *
                   *       ENTRY   JSR     END
                   *
                   *       MODS
                   *               Y. M. Chen      04/01/82
                   *               1. Bring closer to Coding Standard (object :
                   *                  R. K. Nordin 11/01/83


      = C795       END     =       *       ;entry
C795  20CFC7               JSR     GBY     ;get low byte of the RUNADR
C798  A09C                 LDY     #DATAER
C79A  B02C ^C7C8           BCS     END3    ;if EOF before END record

C79C  8DC902               STA     RUNADR
C79F  20CFC7               JSR     GBY     ;get high byte of the RUNADR
C7A2  A09C                 LDY     #DATAER
C7A4  B022 ^C7C8           BCS     END3    ;if EOF before END record

C7A6  8DCA02               STA     RUNADR+1
C7A9  AD4502               LDA     RECLEN  ;RECLEN here is self-start flag
C7AC  C901                 CMP     #1
C7AE  F016 ^C7C6           BEQ     END2    ;if 1, an absolute RUNADR, no fixup

C7B0  9017 ^C7C9           BCC     END4    ;if 0, this is not a self-start pro:

                   ;       Process relative start.
```

```
C7B2  18                      CLC
C7B3  ADC902                  LDA     RUNADR          ;execution address, needs f:
C7B6  6DD102                  ADC     LOADAD
C7B9  A8                      TAY
C7BA  ADCA02                  LDA     RUNADR+1
C7BD  6DD202                  ADC     LOADAD+1        ;A= high byte, Y=low byte

C7C0  8CC902         END1     STY     RUNADR          ;set up Loader-Caller param:
C7C3  8DCA02                  STA     RUNADR+1

C7C6  A001           END2     LDY     #SUCCES         ;Y=1 successful operation

C7C8  60             END3     RTS                     ;return

C7C9  A000           END4     LDY     #0              ;fill self-start parameter :
C7CB  A900                    LDA     #0              ;for non-self start program
C7CD  F0F1 ^C7C0              BEQ     END1            ;continue


                     **       GBY - Get Byte
                     *
                     *        ENTRY   JSR     GBY
                     *
                     *        MODS
                     *                Y. M. Chen      04/01/82
                     *                1. Bring closer to Coding Standard (object :
                     *                   R. K. Nordin 11/01/83

      = C7CF         GBY      =       *               ;entry
C7CF  6CCF02                  JMP     (GBYTEA)        ;get byte, return


                     **       CAL - Execute at Run Address
                     *
                     *        ENTRY   JSR     CAL
                     *
                     *        MODS
                     *                Y. M. Chen      04/01/82
                     *                1. Bring closer to Coding Standard (object :
                     *                   R. K. Nordin 11/01/83

      = C7D2         CAL      =       *               ;entry
C7D2  6CC902                  JMP     (RUNADR)        ;process record, return
```

```
                        **      TEX - Handle Text Record
                        *
                        *       TEX handles record types of
                        *
                        *       1.Non-zero page relocatable text
                        *       2.Zero page relocatable text
                        *       3.Absolute text
                        *
                        *       Record format
                        *
                        *       |Type    |Length          |Relative addr. |text    |
                        *       |ID      |(RECLEN)        |(RELADR)       |        |
                        *
                        *       Process formula
                        *       A register ===> (NEWADR+LCOUNT)
                        *
                        *       Relocate object text into fixed address of NEWADR+L:
                        *
                        *       ENTRY   JSR     TEX
                        *
                        *       NOTES
                        *
                        *       1.The relocating address (NEWADR) for absolute text:
                        *       relative address (RELADR), relocating address fixup:
                        *       needed.
                        *       2.There is no need to compare MEMTOP for processing:
                        *       text.
                        *       3.X register is used as an indexing to zero page va:
                        *       or non-zero page variables.     X=0 means pointing :
                        *       page variable, whereas X=2 means pointing to zero p:
                        *       variables.
                        *       4.Each byte of the object text comes in A register.
                        *
                        *       MODS
                        *               Y. M. Chen      04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = C7D5      TEX           =       *       ;entry
C7D5  AC3302                    LDY     LCOUNT  ;A register=data coming in
C7D8  C001                      CPY     #$01
C7DA  F00A ^C7E6                BEQ     TEX1    ;if 1, process highest used address

C7DC  B073 ^C851                BCS     FTX     ;if 2 or greater, relocate object t:

C7DE  8D4A02                    STA     RELADR
C7E1  8D8E02                    STA     NEWADR  ;for absolute text NEWADR=RELADR
C7E4  906A ^C850                BCC     TEX8

                        ;       Set highest used address.

C7E6  8D4B02      TEX1          STA     RELADR+1        ;save high byte of RELADR
C7E9  8D8F02                    STA     NEWADR+1        ;for absolute text NEWADR=R:
C7EC  A200                      LDX     #0              ;X=an index to non-zero or :
C7EE  AD8802                    LDA     HIBYTE          ;HIBYTE=Type ID
C7F1  F006 ^C7F9                BEQ     TEX2            ;if 0, process non-zero pag:
```

```
C7F3  C90A                CMP     #$0A
C7F5  F015 ^C80C          BEQ     TEX3                ;if $0A, needs no relative :

C7F7  A202                LDX     #2                  ;X=2 for zero page text rec:

C7F9  18          TEX2    CLC                         ;fix relocating addr. for n:
C7FA  AD4A02              LDA     RELADR              ;text & zero page text
C7FD  7DD102              ADC     LOADAD,X            ;NEWADR=RELADR+LOADAD
C800  8D8E02              STA     NEWADR
C803  AD4B02              LDA     RELADR+1
C806  7DD202              ADC     LOADAD+1,X
C809  8D8F02              STA     NEWADR+1            ;Loader start relocating

C80C  18          TEX3    CLC
C80D  AD8E02              LDA     NEWADR  ;NEWADR+RECLEN is the last used mem:
C810  6D4502              ADC     RECLEN  ;for this particular record
C813  48                  PHA
C814  A900                LDA     #0      ;A=high byte, S=low byte
C816  6D8F02              ADC     NEWADR+1
C819  A8                  TAY             ;high byte
C81A  68                  PLA             ;low byte
C81B  38                  SEC
C81C  E902                SBC     #2      ;skip unwanted 2 bytes of relative :
C81E  B001 ^C821          BCS     TEX4

C820  88                  DEY

C821  48          TEX4    PHA
C822  98                  TYA
C823  DDCC02              CMP     HIUSED+1,X          ;HIUSED stores the highest :
C826  68                  PLA
C827  9010 ^C839          BCC     TEX6                ;if HIUSED>(NEWADR+RECLEN),:

C829  D005 ^C830          BNE     TEX5                ;if HIUSED<=(NEWADR+RECLEN)

C82B  DDCB02              CMP     HIUSED,X
C82E  9009 ^C839          BCC     TEX6

                  ;       Update HIUSED.

C830  9DCB02      TEX5    STA     HIUSED,X            ;update HIUSED
C833  48                  PHA
C834  98                  TYA
C835  9DCC02              STA     HIUSED+1,X
C838  68                  PLA

C839  AE8802      TEX6    LDX     HIBYTE
C83C  E001                CPX     #$01
C83E  F010 ^C850          BEQ     TEX8    ;if zero page text

                  ;       Check MEMTOP.

C840  CCE602              CPY     MEMTOP+1            ;MEMTOP>HIUSED, OK
C843  9008 ^C850          BCC     TEX8

C845  D005 ^C84C          BNE     TEX7
```

```
C847  CDE502                CMP     MEMTOP
C84A  9004 ^C850            BCC     TEX8

C84C  68           TEX7     PLA                        ;MEMTOP<=HIUSED then error
C84D  68                    PLA                        ;do a force return to calle:
C84E  A09D                  LDY     #MEMERR            ;set memory insufficient fl:

C850  60           TEX8     RTS                        ;return


                   **       FTX - Relocate Text into Memory
                   *
                   *        ENTRY   JSR     FTX
                   *
                   *        NOTES
                   *                Problem: bytes wasted by JMP to RTS.
                   *
                   *        MODS
                   *                Y. M. Chen      04/01/82
                   *                1. Bring closer to Coding Standard (object :
                   *                   R. K. Nordin 11/01/83


      = C851       FTX      =       *       ;entry
C851  38                    SEC
C852  48                    PHA             ;A register has object text
C853  AD3302                LDA     LCOUNT  ;LCOUNT counts 2 bytes of relative :
C856  E902                  SBC     #2      ;-2 is the total bytes of object te:
C858  18                    CLC
C859  6D8E02                ADC     NEWADR
C85C  8536                  STA     LTEMP   ;A ===>(NEWADR+LCOUNT-2)
C85E  A900                  LDA     #0
C860  6D8F02                ADC     NEWADR+1
C863  8537                  STA     LTEMP+1
C865  68                    PLA
C866  A000                  LDY     #0
C868  9136                  STA     (LTEMP),Y
C86A  4C50C8                JMP     TEX8    ;return


                   **       WOR - Handle Word Reference Record Type
                   *
                   *        WOR handles record types of
                   *
                   *        1.Non-zero page word references to non-zero page.
                   *        2.Zero page word references to non-zero page.
                   *
                   *        Record format
                   *
                   *        |Type   |Length          |Offset1|Offset2|Offsetn|
                   *        | ID    |(RECLEN)        |A Reg. |       |       |
                   *
                   *        Process formula
```

```
                       *
                       *        (A register +NEWADR)W +LOADAD ===> (NEWADR+ A regis:
                       *
                       *        Count, the offset from the start relocating address:
                       *        low byte
                       *        of a word needing to be fixed.  The fixup process i:
                       *        content of the word and add loading address, then r:
                       *        fixed word.
                       *
                       *        Offset information comes in A register.
                       *
                       *        ENTRY   JSR     WOR
                       *
                       *        MODS
                       *                Y. M. Chen      04/01/82
                       *                1. Bring closer to Coding Standard (object :
                       *                 R. K. Nordin 11/01/83


       = C86D          WOR      =       *               ;entry
C86D   18                       CLC
C86E   6D8E02                   ADC     NEWADR          ;offset in A register
C871   8536                     STA     LTEMP
C873   A900                     LDA     #0
C875   6D8F02                   ADC     NEWADR+1
C878   8537                     STA     LTEMP+1         ;offset +NEWADR= LTEMP
C87A   A000                     LDY     #0
C87C   B136                     LDA     (LTEMP),Y       ;get low byte content of wh:
C87E   18                       CLC
C87F   6D0102                   ADC     LOADAD          ;fix low byte of a word
C882   9136                     STA     (LTEMP),Y
C884   E636                     INC     LTEMP           ;increment LTEMP pointer by:
C886   D002 ^C88A               BNE     WOR1            ;if low not zero

C888   E637                     INC     LTEMP+1         ;increment high

C88A   B136            WOR1     LDA     (LTEMP),Y       ;fix high byte of a word
C88C   6D0202                   ADC     LOADAD+1
C88F   9136                     STA     (LTEMP),Y       ;restore processed content
C891   60                       RTS                     ;return



                       **       LOO - Handle Low Byte and One Byte Record Types
                       *
                       *        LOO handles record types of
                       *
                       *        1.Non-zero page low byte references to non-zero ppa:
                       *        2.Zero page low-byte references to non-zero page.
                       *        3.Non-zero page one byte references to zero page.
                       *        4.Zero page one byte references to zero page.
                       *
                       *        Record format
                       *
                       *        |Type   |Length         |Offset1|Offset2|Offsetn|
                       *        |ID     |(RECLEN)       |A Reg. |A Reg. |       |
                         *
                         *      The process formula for non-zero page low byte refe:
                         *      non-zero page record and zero page low byte referen:
                         *      non-zero page record is
                         *
                         *      (offset + NEWADR)+LOADAD ===> (offset +NEWADR)
                         *
                         *      The process formula for non-zero page one byte refe:
                         *      zero
                         *      page record and zero page one byte references to ze:
                         *      record
                         *      is
                         *
                         *      (offset + NEWADR)+LOADADZ ===> (offset + NEWADR)
                         *
                         *      Count from the offset from the start relocating add:
                         *      low byte or one byte need to be fixed. Get the cont:
                         *      low byte or one byteand add either LOADAD or LOADAD:
                         *      page loading address), then restore the value.
                         *
                         *      The offset comes in A register.
                         *
                         *      The X register for this routine points to either no:
                         *      variables or zero page variables. Record type 2 & 3:
                         *      non-zero page variable, type 4 & 5 needs zero page :
                         *
                         *      X=2 points to zero page variable.
                         *
                         *      ENTRY   JSR     LOO
                         *
                         *      MODS
                         *              Y. M. Chen      04/01/82
                         *              1. Bring closer to Coding Standard (object :
                         *                 R. K. Nordin 11/01/83

      = C892       LOO     =       *        ;entry
C892  A200                 LDX     #0       ;X=0 points to non-zero page variab:
C894  AC8802               LDY     HIBYTE   ;HIBYTE has Type ID
C897  C004                 CPY     #4       ;type 4 & 5 needs zero page variabl:
C899  9002 ^C89D           BCC     LOO1     ;if type 2 or 3, need non-zero page:

C89B  A202                 LDX     #2       ;point to zero page variable

C89D  18           LOO1    CLC              ;offset is in A register
C89E  6D8E02               ADC     NEWADR   ;offset+NEWADR=the byte needs fixup
C8A1  8536                 STA     LTEMP
C8A3  A900                 LDA     #0
C8A5  6D8F02               ADC     NEWADR+1
C8A8  8537                 STA     LTEMP+1
C8AA  A000                 LDY     #0
C8AC  B136                 LDA     (LTEMP),Y        ;get the content of offset+:
C8AE  18                   CLC
C8AF  7D0102               ADC     LOADAD,X         ;do relocating fixup
C8B2  9136                 STA     (LTEMP),Y        ;restore the being fixed va:
C8B4  60                   RTS                      ;return
```

```
                        **      HIG - Handle High Byte Record Types
                        *
                        *       HIG handles record types of
                        *
                        *       1.Non-zero page high bytes references to non-zero p:
                        *       2.Zero page high bytes references to non-zero page.
                        *
                        *       Record format
                        *
                        *       |Type   |Length         |Offset1|Low Byte|Offset2|L:
                        *       |ID     |(RECLEN)       |HIBYTE |A Reg.  | (HIBYTE):
                        *
                        *       Process formula
                        *
                        *       (HIBYTE+NEWADR)+[[LOADAD+A]/256] ==> (HIBYTE+NEWADR:
                        *
                        *       Count the offset from the start relocating address :
                        *       byte needs to be fixed. Get the low byte informatio:
                        *       A register, then add the low byte with LOADAD and s:
                        *       flag depending on the calculation. Next do an addit:
                        *       high byte, NEWADR and the C flag. Restore the addit:
                        *       back to the high byte location.in memory.
                        *
                        *       HIBYTE is not Type ID here. HIBYTE is used to store:
                        *       byte value.
                        *
                        *       ENTRY   JSR     HIG
                        *
                        *       NOTES
                        *               Problem: many instances of jumping to RTS i:
                        *               wastes bytes.
                        *
                        *       MODS
                        *               Y. M. Chen      04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = C8B5            HIG     =       *               ;entry

                        ;       Initialize.

C8B5  48                        PHA                     ;save offset pointing to hi:

                        ;       Check LCOUNT odd or even.

C8B6  AD3302                    LDA     LCOUNT
C8B9  6A                        ROR     A
C8BA  68                        PLA
C8BB  B015 ^C8D2                BCS     HIG2            ;if even number, process lo:

                        ;       Process high byte.

C8BD  18                        CLC
C8BE  6D3E02                    ADC     NEWADR
C8C1  8536                      STA     LTEMP           ;get high byte value
C8C3  A900                      LDA     #0
```

```
C8C5  6D8F02               ADC     NEWADR+1
C8C8  8537                 STA     LTEMP+1
C8CA  A000                 LDY     #0
C8CC  B136                 LDA     (LTEMP),Y
C8CE  8D8802               STA     HIBYTE          ;save high byte content

C8D1  60          HIG1     RTS                     ;return

                  ;        Process low byte.

C8D2  18          HIG2     CLC
C8D3  6DD102               ADC     LOADAD          ;add low byte with LOADAD
C8D6  A900                 LDA     #0
C8D8  6DD202               ADC     LOADAD+1
C8DB  6D8802               ADC     HIBYTE          ;C flag+LOADAD(high byte)+H:
C8DE  A000                 LDY     #0
C8E0  9136                 STA     (LTEMP),Y       ;store being fixed high byt:
C8E2  F0ED ^C8D1           BEQ     HIG1


                  **       TRPR - Table of Record Processing Routines


C8E4  D5C7        TRPR     DW      TEX      ;0 - non-zero page relocatable text
C8E6  D5C7                 DW      TEX      ;1 - zero page relocatable text
C8E8  92C8                 DW      LOO      ;2 - non-zero page low byte to non-:
C8EA  92C8                 DW      LOO      ;3 - zero page low byte to non-zero:
C8EC  92C8                 DW      LOO      ;4 - non-zero page one byte to zero:
C8EE  92C8                 DW      LOO      ;5 - zero page one byte to zero pag:
C8F0  6DC8                 DW      WOR      ;6 - non-zero page word to non-zero:
C8F2  6DC8                 DW      WOR      ;7 - zero page word to non-zero pag:
C8F4  85C8                 DW      HIG      ;8 - non-zero page high byte to non:
C8F6  85C8                 DW      HIG      ;9 - zero page high byte to non-zer:
C8F8  D5C7                 DW      TEX      ;10 - absolute text
C8FA  95C7                 DW      END      ;11 - end record
```

```
C8FC                    **      SES - Select and Execute Self-test
                        *
                        *       SES selects the self-test ROM and executes the self:
                        *
                        *       ENTRY   JSR     SES
                        *
                        *       NOTES
                        *               Problem: this could be contiguous with othe:
                        *               self-test code (near TSTO).
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = C8FC            SES     =       *       ;entry

C8FC  A9FF                      LDA     #$FF
C8FE  8D4402                    STA     COLDST  ;force coldstart on RESET

C901  AD01D3                    LDA     PORTB
C904  297F                      AND     #$7F    ;enable self-test ROM
C906  8D01D3                    STA     PORTB   ;update port B memory control

C909  4C83E4                    JMP     SLFTSV  ;vector to self-test
```

```
C90C                   **      GIN - Initialize Generic Parallel Device
                       *
                       *       ENTRY   JSR     GIN
                       *
                       *       MODS
                       *               Y. M. Chen      02/18/83
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83


      = C90C     GIN           =       *       ;entry

                       ;       Initialize.

C90C  A901                     LDA     #$01    ;initially select device 0
C90E  8D4802                   STA     SHPDVS  ;device select shadow

                       ;       For each potential device, initialize if device pre:

C911  AD4802     GIN1          LDA     SHPDVS  ;device select shadow
C914  8DFFD1                   STA     PDVS    ;device select

C917  AD03D8                   LDA     PDID1   ;first ID
C91A  C980                     CMP     #$80    ;required value
C91C  D00A ^C928               BNE     GIN2    ;if first ID not verified

C91E  AD0BD8                   LDA     PDID2   ;second ID
C921  C991                     CMP     #$91    ;required value
C923  D003 ^C928               BNE     GIN2    ;if second ID not verified

C925  2019D8                   JSR     PDVV+12 ;initialize parallel device handler

C928  0E4802     GIN2          ASL     SHPDVS  ;advance to next device
C92B  D0E4 ^C911               BNE     GIN1    ;if devices remain

                       ;       Exit.

C92D  A900                     LDA     #$00    ;select FPP (deselect device)
                       ;       STA     SHPDVS  ;device select shadow
C92F  8DFFD1                   STA     PDVS    ;device select
C932  60                       RTS             ;return



                       **      PIO - Parallel Input/Output
                       *
                       *       ENTRY   JSR     PIO
                       *
                       *       NOTES
                       *               Problem: in the CRASS65 version, CRITIC was:
                       *               zero-page.
                       *
                       *       MODS
                       *               Y. M. Chen      02/18/83
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83
      = C933        PIO      =        *       ;entry

                    ;        Initialize.

C933  A901                   LDA      #1
                    ;        STA      CRITIC  ;indicate critical section
C935  8D4200                 VFD      8\$8D,8\low CRITIC,8\high CRITIC
C938  AD0103                 LDA      DUNIT   ;device unit number
C93B  48                     PHA              ;save device unit number
C93C  AD4702                 LDA      PDVMSK  ;device selection mask
C93F  F01A ^C95B             BEQ      PIO2    ;if no device to select

                    ;        For each device, pass request to device I/O routine:

C941  A208                   LDX      #TPDSL  ;offset to first byte beyond table

C943  20AFC9        PIO1     JSR      SNP     ;select next parallel device
C946  F013 ^C95B             BEQ      PIO2    ;if no device selected

C948  8A                     TXA
C949  48                     PHA              ;save offset
C94A  2005D8                 JSR      PDIOV   ;perform parallel device I/O
C94D  68                     PLA              ;saved offset
C94E  AA                     TAX              ;restore offset
C94F  90F2 ^C943             BCC      PIO1    ;if device did not field request

                    ;        Restore Floating Point Package.

C951  A900                   LDA      #$00    ;select FPP (deselect device)
C953  8D4802                 STA      SHPDVS  ;device select shadow
C956  8DFFD1                 STA      PDVS    ;device select
C959  F003 ^C95E             BEQ      PIO3    ;exit

                    ;        Perform SIO.

C95B  2071E9        PIO2     JSR      SIO     ;perform SIO

                    ;        Exit.

C95E  68            PIO3     PLA              ;saved device unit number
C95F  8D0103                 STA      DUNIT   ;restore device unit number
C962  A900                   LDA      #0
                    ;        STA      CRITIC  ;indicate non-critical section
C964  8D4200                 VFD      8\$8D,8\low CRITIC,8\high CRITIC
C967  8C0303                 STY      DSTATS
C96A  AC0303                 LDY      DSTATS  ;status (re-establish N)
C96D  60                     RTS              ;return
```

```
                        **      PIR - Handle Parallel Device IRQ
                        *
                        *       ENTRY   JMP     PIR
                        *
                        *       EXIT
                        *               Exits via RTI
                        *
                        *       MODS
                        *               Y. M. Chen      02/18/83
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = C96E            PIR     =       *       ;entry

                        ;       Determine which device made IRQ, in order of priori:

C96E  A208                      LDX     #TPDSL  ;offset to first byte beyond table

C970  6A                PIR1    ROR     A
C971  B003 ^C976                BCS     PIR2    ;if IRQ of that device

C973  CA                        DEX
C974  D0FA ^C970                BNE     PIR1    ;if devices remain

                        ;       Select device and process IRQ.

C976  AD4802            PIR2    LDA     SHPDVS          ;current device selection
C979  48                        PHA                     ;save current device select:
C97A  BD20CA                    LDA     TPDS-1,X        ;device selection desired
C97D  8D4802                    STA     SHPDVS          ;device select shadow
C980  8DFFD1                    STA     PDVS            ;device select
C983  2008D8                    JSR     PDIRQV          ;process IRQ

                        ;       Exit.

C986  68                        PLA                     ;saved device selection
C987  8D4802                    STA     SHPDVS          ;restore device select shad:
C98A  8DFFD1                    STA     PDVS            ;device select
C98D  68                        PLA                     ;saved X
C98E  AA                        TAX                     ;restore X
C98F  68                        PLA                     ;restore A
C990  40                        RTI                     ;return
```

```
                                **      GOP - Perform Generic Parallel Device OPEN
                                *
                                *       ENTRY   JSR     GOP
                                *
                                *       MODS
                                *               Y. M. Chen      02/18/83
                                *               1. Bring closer to Coding Standard (object :
                                *                  R. K. Nordin 11/01/83

          = C991        GOP     =       *       ;entry
C991  A001                      LDY     #1      ;offset for OPEN
C993  4CDCC9                    JMP     EPC     ;execute parallel device handler co:


                                **      GCL - Perform Generic Parallel Device CLOSE
                                *
                                *       ENTRY   JSR     GCL
                                *
                                *       MODS
                                *               Y. M. Chen      02/18/83
                                *               1. Bring closer to Coding Standard (object :
                                *                  R. K. Nordin 11/01/83

          = C996        GCL     =       *       ;entry
C996  A003                      LDY     #3      ;offset for CLOSE
C998  4CDCC9                    JMP     EPC     ;execute parallel device handler co:


                                **      GGB - Perform Generic Parallel Device GET-BYTE
                                *
                                *       ENTRY   JSR     GGB
                                *
                                *       MODS
                                *               Y. M. Chen      02/18/83
                                *               1. Bring closer to Coding Standard (object :
                                *                  R. K. Nordin 11/01/83

          = C99B        GGB     =       *       ;entry
C99B  A005                      LDY     #5      ;offset for GET-BYTE
C99D  4CDCC9                    JMP     EPC     ;execute parallel device handler co:
```

```
                        **      GPB - Perform Generic Parallel Device PUT-BYTE
                        *
                        *       ENTRY   JSR     GPB
                        *
                        *       MODS
                        *               Y. M. Chen      02/18/83
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = C9A0          GPB     =       *       ;entry
C9A0  A007                      LDY     #7      ;offset for PUT-BYTE
C9A2  4CDCC9                    JMP     EPC     ;execute parallel device handler co:



                        **      GST - Perform Generic Parallel Device STATUS
                        *
                        *       ENTRY   JSR     GST
                        *
                        *       MODS
                        *               Y. M. Chen      02/18/83
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = C9A5          GST     =       *       ;entry
C9A5  A009                      LDY     #9      ;offset for STATUS
C9A7  4CDCC9                    JMP     EPC     ;execute parallel device handler co:



                        **      GSP - Perform Generic Parallel Device SPECIAL
                        *
                        *       ENTRY   JSR     GSP
                        *
                        *       MODS
                        *               Y. M. Chen      02/18/83
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = C9AA          GSP     =       *       ;entry
C9AA  A00B                      LDY     #11     ;offset for SPECIAL
C9AC  4CDCC9                    JMP     EPC     ;execute parallel device handler co:
```

```
                    **      SNP - Select Next Parallel Device
                    *
                    *       ENTRY   JSR     SNP
                    *
                    *       MODS
                    *               Y. M. Chen      02/18/83
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = C9AF        SNP     =       *       ;entry

                    ;       Decrement and check offset.

C9AF  CA            SNP1    DEX             ;decrement offset
C9B0  1009 ^C9BB            BPL     SNP2    ;if devices remain

                    ;       Exit.

C9B2  A900                  LDA     #$00    ;select FPP (deselect device)
C9B4  8D4802                STA     SHPDVS  ;device select shadow
C9B7  8DFFD1                STA     PDVS    ;device select.
C9BA  60                    RTS             ;return

                    ;       Ensure device is indicated by selection mask.

C9BB  AD4702        SNP2    LDA     PDVMSK  ;device selection mask
C9BE  3D21CA                AND     TPDS,X  ;device select
C9C1  F0EC ^C9AF            BEQ     SNP1    ;if device not indicated for select:

                    ;       Select device.

C9C3  8D4802                STA     SHPDVS  ;device select shadow
C9C6  8DFFD1                STA     PDVS    ;device select
C9C9  60                    RTS             ;return



                    **      IPH - Invoke Parallel Device Handler
                    *
                    *       ENTRY   JSR     IPH
                    *               Y = offset into parallel device vector tabl:
                    *               PPTMPA = original A value
                    *               PPTMPX = original X value
                    *
                    *       NOTES
                    *               Problem: wasted byte for DEY.
                    *
                    *       MODS
                    *               Y. M. Chen      02/18/83
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = C9CA        IPH     =       *       ;entry
C9CA  B90DD8                LDA     PDVV,Y  ;high routine address-1
```

```
C9CD  48                    PHA                 ;place on stack
C9CE  88                    DEY
C9CF  B900D8                LDA     PDVV,Y      ;low routine address-1
C9D2  48                    PHA                 ;place on stack
C9D3  AD4C02                LDA     PPTMPA      ;restore A for handler
C9D6  AE4D02                LDX     PPTMPX      ;restore X for handler
C9D9  A092                  LDY     #FNCNOT     ;preset status
C9DB  60                    RTS                 ;invoke handler routine (address on


                   **       EPC - Execute Parallel Device Handler Command
                   *
                   *        ENTRY   JSR     EPC
                   *
                   *        NOTES
                   *                Problem: in the CRASS65 version, CRITIC was
                   *                zero-page.
                   *
                   *        MODS
                   *                Y. M. Chen      02/18/83
                   *                1. Bring closer to Coding Standard (object
                   *                   R. K. Nordin 11/01/83

      = C9DC       EPC      =       *           ;entry

                   ;        Initialize.

C9DC  8D4C02                STA     PPTMPA      ;save data byte
C9DF  8E4D02                STX     PPTMPX      ;save X
                   ;        LDA     CRITIC
C9E2  AD4200                VFD     8\$AD,8\low CRITIC,8\high CRITIC
C9E5  48                    PHA                 ;save critical section status
C9E6  A901                  LDA     #1
                   ;        STA     CRITIC      ;indicate critical section
C9E8  8D4200                VFD     8\$8D,8\low CRITIC,8\high CRITIC

                   ;        For each device, pass request to device handler.

C9EB  A208                  LDX     #TPDSL      ;offset to first byte beyond table

C9ED  20AFC9       EPC1     JSR     SNP         ;select next device
C9F0  F011 ^CA03            BEQ     EPC2        ;if no device selected, return erro

C9F2  8A                    TXA
C9F3  48                    PHA                 ;save offset
C9F4  98                    TYA
C9F5  48                    PHA                 ;save Y
C9F6  20CAC9                JSR     IPH         ;invoke parallel device handler
C9F9  9020 ^CA1B            BCC     EPC4        ;if device did not field, try next

                   ;        Clean up.

C9FB  8D4C02                STA     PPTMPA      ;save possible data byte
C9FE  68                    PLA                 ;clean stack
```

```
C9FF  68                    PLA
CA00  4C05CA                JMP     EPC3     ;exit

                  ;         Return Nonexistent Device error.

CA03  A082        EPC2      LDY     #NONDEV

                  ;         Restore Floating Point Package.

CA05  A900        EPC3      LDA     #$00     ;select FPP (deselect device)
CA07  8D4802                STA     SHPDVS   ;device select shadow
CA0A  8DFFD1                STA     PDVS     ;device select
CA0D  68                    PLA              ;saved critical section status
                  ;         STA     CRITIC   ;restore critical section status
CA0E  8D4200                VFD     8\$8D,8\low CRITIC,8\high CRITIC
CA11  AD4C02                LDA     PPTMPA   ;restore possible data byte
CA14  8C4D02                STY     PPTMPX
CA17  AC4D02                LDY     PPTMPX   ;status (re-establish N)
CA1A  60                    RTS              ;return

                  ;         Prepare to try next device.

CA1B  68          EPC4      PLA
CA1C  A8                    TAY              ;restore Y
CA1D  68                    PLA
CA1E  AA                    TAX              ;restore X
CA1F  90CC ^C9ED            BCC     EPC1     ;try next device



                  **        TPDS - Table of Parallel Device Selects
                  *
                  *         NOTES
                  *                 Problem: bytes wasted by replication of thi:
                  *                 elsewhere.

CA21  80          TPDS      DB      $80      ;0 - device 7 (lowest priority)
CA22  40                    DB      $40      ;1 - device 6
CA23  20                    DB      $20      ;2 - device 5
CA24  10                    DB      $10      ;3 - device 4
CA25  08                    DB      $08      ;4 - device 3
CA26  04                    DB      $04      ;5 - device 2
CA27  02                    DB      $02      ;6 - device 1
CA28  01                    DB      $01      ;7 - device 0 (highest priority)

      = 0008      TPDSL     =       *-TPDS   ;length
```

```
CA29                  **      PHL - Load and Initialize Peripheral Handler
                      *
                      *       Subroutine to load, relocate, initialize and open a
                      *       "provisionally" opened IOCB. This routine is called
                      *       upon first I/O attempt following provisional open.
                      *       It does the final opening by simulating the first
                      *       part of a normal CIO OPEN and then finishing with
                      *       code which is in CIO.
                      *
                      *       Input parameters:
                      *       ICIDNO (specifies which IOCB);
                      *       various values in the provisionally-opened IOCB:
                      *               ICSPR (handler name)
                      *               ICSPR+1 (serial address for loading);
                      *       whatever the called subroutines require.
                      *
                      *       Output parameters:
                      *       None. (Error returns are all handled by called subr:
                      *               in fact, all returns are handled by called :
                      *
                      *       Modified:
                      *       ICHID in both calling IOCB and ZIOCB (part of compl:
                      *       ICCOMT (a CIO variable);
                      *       Registers not saved.
                      *
                      *       Subroutines called:
                      *       LPH (does the loading);
                      *       PHC (initializes the loaded handler);
                      *       FDH (a CIO entry--finds handler table entry of
                      *               newly loaded/initialized handler);
                      *       IIO (a CIO entry--finishes full proper opening of I:
                      *               including calling handler OPEN entry--IIO r:
                      *               to PHL's caller);
                      *       IND (a CIO entry--returns with error to PHL's calle:
                      *
                      *       ENTRY   JSR     PHL
                      *
                      *       NOTES
                      *               Problem: in the CRASS65 version, ICIDNO was:
                      *               zero-page.
                      *
                      *       MODS
                      *               R. S. Scheiman  04/01/82
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83

         = CA29       PHL     =       *                 ;entry

                      ;       Load peripheral handler.

                      ;       LDX     ICIDNO            ;IOCB index
CA29   AE2E00                 VFD     8\$AE,8\low ICIDNO,8\high ICIDNO
CA2C   BD4D03                 LDA     ICSPR+1,X
CA2F   20DEE7                 JSR     LPH               ;load peripheral handler
CA32   B020 ^CA54             BCS     PHL1              ;if error
```

```
                    ;       Initialize peripheral handler.

CA34  18                    CLC                         ;indicate zero handler size
CA35  209EE8                JSR     PHC                 ;initialize peripheral hand:
CA38  B01A ^CA54            BCS     PHL1                ;if error

                    ;       Find device handler.

                    ;       LDX     ICIDNO              ;IOCB index
CA3A  AE2E00                VFD     8\$AE,8\low ICIDNO,8\high ICIDNO
CA3D  BD4C03                LDA     ICSPR,X
CA40  2016E7                JSR     FDH                 ;find device handler
CA43  B00F ^CA54            BCS     PHL1                ;if not found

                    ;       Set handler ID.

                    ;       LDX     ICIDNO              ;IOCB index
CA45  AE2E00                VFD     8\$AE,8\low ICIDNO,8\high ICIDNO
CA48  9D4003                STA     ICHID,X             ;handler ID
CA4B  8520                  STA     ICHIDZ

                    ;       Simulate initial CIO OPEN processing.

CA4D  A903                  LDA     #OPEN   ;OPEN command
CA4F  8517                  STA     ICCOMT  ;command
CA51  4C5CE5                JMP     IIO     ;initialize IOCB for OPEN, return

                    ;       Indicate nonexistent device error.

CA54  4C10E5        PHL1    JMP     IND     ;indicate nonexistent device error,:
```

```
CA57                 **      TSTO - Table of Self-test Text Offsets


CA57  00             TSTO    DB       TXT0-TTXT        ;0 - offset to "MEMORY TEST:
CA58  13                     DB       TXT1-TTXT        ;1 - offset to "RAM" text
CA59  16                     DB       TXT2-TTXT        ;2 - offset to "KEYBOARD TE:
CA5A  D1                     DB       TXT3-TTXT        ;3 - offset to "S P A C E  :
CA5B  E4                     DB       TXT4-TTXT        ;4 - offset to "SH" text
CA5C  E4                     DB       TXT5-TTXT        ;5 - offset to "SH" text
CA5D  E8                     DB       TXT6-TTXT        ;6 - offset to "B S" text
CA5E  29                     DB       TXT7-TTXT        ;7 - offset to keyboard tex:
CA5F  EB                     DB       TXT8-TTXT        ;8 - offset to control key :
CA60  EE                     DB       TXT9-TTXT        ;9 - offset to "VOICE #" te:



                     **      TTXT - Table of Text Sequences


      = CA61         TTXT    =        *



                     **      TXT0 - "MEMORY TEST   ROM" Text


CA61  0000           TXT0    DB       $00,$00
CA63  2D252D2F32             DB       $2D,$25,$2D,$2F,$32,$39 ;"MEMORY"
CA69  00                     DB       $00
CA6A  34253334               DB       $34,$25,$33,$34         ;"TEST"
CA6E  000000                 DB       $00,$00,$00
CA71  322F2D                 DB       $32,$2F,$2D             ;"ROM"

      = 0013         TXT0L   =        *-TXT0  ;length



                     **      TXT1 - "RAM" Text


CA74  32212D         TXT1    DB       $32,$21,$2D             ;"RAM"

      = 0003         TXT1L   =        *-TXT1  ;length
```

```
                         **      TXT2 - "KEYBOARD TEST" Text

CA77   0000              TXT2    DB      $00,$00
CA79   2B2539222F                DB      $2B,$25,$39,$22,$2F,$21,$32,$24 ;"KEYBOARD"
CA81   00                        DB      $00
CA82   34253334                  DB      $34,$25,$33,$34                 ;"TEST"
CA86   000000                    DB      $00,$00,$00
CA89   B2                        DB      $B2

       = 0013            TXT2L   =       *-TXT2  ;length


                         **      TXT7 - Keyboard

       = CA8A            TXT7    =       *

                         ;       First Row (Function Keys)

CA8A   91                        DB      $91             ;"1"
CA8B   00                        DB      $00
CA8C   92                        DB      $92             ;"2"
CA8D   00                        DB      $00
CA8E   93                        DB      $93             ;"3"
CA8F   00                        DB      $00
CA90   94                        DB      $94             ;"4"
CA91   00                        DB      $00
CA92   A8                        DB      $A8             ;"H"
CA93   00                        DB      $00
CA94   A1                        DB      $A1             ;"A"
CA95   00                        DB      $00
CA96   A2                        DB      $A2             ;"B"
CA97   000000                    DB      $00,$00,$00

                         ;       Second Row ("1 2 3 4 5 6 7 8 9 0 < >")

CA9A   5B                        DB      $5B
CA9B   00                        DB      $00
CA9C   11                        DB      $11             ;"1"
CA9D   00                        DB      $00
CA9E   12                        DB      $12             ;"2"
CA9F   00                        DB      $00
CAA0   13                        DB      $13             ;"3"
CAA1   00                        DB      $00
CAA2   14                        DB      $14             ;"4"
CAA3   00                        DB      $00
CAA4   15                        DB      $15             ;"5"
CAA5   00                        DB      $00
CAA6   16                        DB      $16             ;"6"
CAA7   00                        DB      $00
CAA8   17                        DB      $17             ;"7"
CAA9   00                        DB      $00
CAAA   18                        DB      $18             ;"8"
CAAB   00                        DB      $00
```

```
CAAC  19                 DB      $19             ;"9"
CAAD  00                 DB      $00
CAAE  10                 DB      $10             ;"0"
CAAF  00                 DB      $00
CAB0  1C                 DB      $1C             ;"<"
CAB1  00                 DB      $00
CAB2  1E                 DB      $1E             ;">"
CAB3  00                 DB      $00
CAB4  A2                 DB      $A2             ;"B"
CAB5  80                 DB      $80
CAB6  B3                 DB      $B3             ;"S"
CAB7  000000             DB      $00,$00,$00

                  ;      Third Row ("Q W E R T Y U I O P - =")

CABA  FF                 DB      $FF
CABB  FF                 DB      $FF
CABC  00                 DB      $00
CABD  31                 DB      $31             ;"Q"
CABE  00                 DB      $00
CABF  37                 DB      $37             ;"W"
CAC0  00                 DB      $00
CAC1  25                 DB      $25             ;"E"
CAC2  00                 DB      $00
CAC3  32                 DB      $32             ;"R"
CAC4  00                 DB      $00
CAC5  34                 DB      $34             ;"T"
CAC6  00                 DB      $00
CAC7  39                 DB      $39             ;"Y"
CAC8  00                 DB      $00
CAC9  35                 DB      $35             ;"U"
CACA  00                 DB      $00
CACB  29                 DB      $29             ;"I"
CACC  00                 DB      $00
CACD  2F                 DB      $2F             ;"O"
CACE  00                 DB      $00
CACF  30                 DB      $30             ;"P"
CAD0  00                 DB      $00
CAD1  0D                 DB      $0D             ;"-"
CAD2  00                 DB      $00
CAD3  1D                 DB      $1D             ;"="
CAD4  00                 DB      $00
CAD5  B2                 DB      $B2             ;"R"
CAD6  B4                 DB      $B4             ;"T"
CAD7  000000             DB      $00,$00,$00

                  ;      Fourth Row ("A S D F G H J K L ; + *")

CADA  80                 DB      $80
CADB  DC                 DB      $DC
CADC  80                 DB      $80
CADD  00                 DB      $00
CADE  21                 DB      $21             ;"A"
CADF  00                 DB      $00
CAE0  33                 DB      $33             ;"S"
CAE1  00                 DB      $00
CAE2  24                 DB      $24             ;"D"
```

```
CAE3  00                   DB      $00
CAE4  26                   DB      $26              ;"F"
CAE5  00                   DB      $00
CAE6  27                   DB      $27              ;"G"
CAE7  00                   DB      $00
CAE8  28                   DB      $28              ;"H"
CAE9  00                   DB      $00
CAEA  2A                   DB      $2A              ;"J"
CAEB  00                   DB      $00
CAEC  2B                   DB      $2B              ;"K"
CAED  00                   DB      $00
CAEE  2C                   DB      $2C              ;"L"
CAEF  00                   DB      $00
CAF0  18                   DB      $18              ;";"
CAF1  00                   DB      $00
CAF2  0B                   DB      $0B              ;"+"
CAF3  00                   DB      $00
CAF4  0A                   DB      $0A              ;"*"
CAF5  00                   DB      $00
CAF6  A3                   DB      $A3              ;"C"
CAF7  000000               DB      $00,$00,$00

                   ;       Fifth Row ("Z X C V B N M , . /")

CAFA  80                   DB      $80
CAFB  B3                   DB      $B3              ;"S"
CAFC  A8                   DB      $A8              ;"H"
CAFD  80                   DB      $80
CAFE  00                   DB      $00
CAFF  3A                   DB      $3A              ;"Z"
CB00  00                   DB      $00
CB01  38                   DB      $38              ;"X"
CB02  00                   DB      $00
CB03  23                   DB      $23              ;"C"
CB04  00                   DB      $00
CB05  36                   DB      $36              ;"V"
CB06  00                   DB      $00
CB07  22                   DB      $22              ;"B"
CB08  00                   DB      $00
CB09  2E                   DB      $2E              ;"N"
CB0A  00                   DB      $00
CB0B  2D                   DB      $2D              ;"M"
CB0C  00                   DB      $00
CB0D  0C                   DB      $0C              ;","
CB0E  00                   DB      $00
CB0F  0E                   DB      $0E              ;"."
CB10  00                   DB      $00
CB11  0F                   DB      $0F              ;"/"
CB12  00                   DB      $00
CB13  80                   DB      $80
CB14  B3                   DB      $B3              ;"S"
CB15  A8                   DB      $A8              ;"H"
CB16  80                   DB      $80
CB17  000000               DB      $00,$00,$00

                   ;       Sixth Row (Space Bar)
```

```
CB1A  0000000000           DB      $00,$00,$00,$00,$00
CB1F  80                   DB      $80
CB20  B3                   DB      $B3          ;"S"
CB21  80                   DB      $80
CB22  B0                   DB      $B0          ;"P"
CB23  80                   DB      $80
CB24  A1                   DB      $A1          ;"A"
CB25  80                   DB      $80
CB26  A3                   DB      $A3          ;"C"
CB27  80                   DB      $80
CB28  A5                   DB      $A5          ;"E"
CB29  80                   DB      $80
CB2A  80                   DB      $80
CB2B  80                   DB      $80
CB2C  A2                   DB      $A2          ;"B"
CB2D  80                   DB      $80
CB2E  A1                   DB      $A1          ;"A"
CB2F  80                   DB      $80
CB30  B2                   DB      $B2          ;"R"
CB31  80                   DB      $80

      = 00A8      TXT7L    =       *-TXT7   ;length



                  **       TXT3 - "S P A C E  B A R" Text


CB32  00          TXT3     DB      $00
CB33  33                   DB      $33          ;"S"
CB34  00                   DB      $00
CB35  30                   DB      $30          ;"P"
CB36  00                   DB      $00
CB37  21                   DB      $21          ;"A"
CB38  00                   DB      $00
CB39  23                   DB      $23          ;"C"
CB3A  00                   DB      $00
CB3B  25                   DB      $25          ;"E"
CB3C  00                   DB      $00
CB3D  00                   DB      $00
CB3E  00                   DB      $00
CB3F  22                   DB      $22          ;"B"
CB40  00                   DB      $00
CB41  21                   DB      $21          ;"A"
CB42  00                   DB      $00
CB43  32                   DB      $32          ;"R"
CB44  00                   DB      $00

      = 0013      TXT3L    =       *-TXT3   ;length
```

```
                        **      TXT4 - "SH" Text


CB45  00                TXT4    DB      $00
CB46  3328                      DB      $33,$28         ;"SH"
CB48  00                        DB      $00

      = 0004            TXT4L   =       *-TXT4  ;length



                        **      TXT5 - "SH" Text


      = CB45            TXT5    =       TXT4

      = 0004            TXT5L   =       TXT4L   ;length



                        **      TXT6 - "B S" Text


CB49  22                TXT6    DB      $22     ;"B"
CB4A  00                        DB      $00
CB4B  33                        DB      $33     ;"S"

      = 0003            TXT6L   =       *-TXT6  ;length



                        **      TXT8 - Control Key


CB4C  00                TXT8    DB      $00
CB4D  5C                        DB      $5C
CB4E  00                        DB      $00

      = 0003            TXT8L   =       *-TXT8  ;length
```

```
                        **        TXT9 - "VOICE #" Text

CB4F   362F292325  TXT9     DB        $36,$2F,$29,$23,$25      ;"VOICE"
CB54   00                   DB        $00
CB55   03                   DB        $03                      ;"#"

       = 0007      TXT9L    =         *-TXT9  ;length
```

```
CB56                    **      CLT - Checksum Linkage Table
                        *
                        *       ENTRY   JSR     CLT
                        *               ZCHAIN - ZCHAIN+1 = address of linkage tabl:
                        *
                        *       EXIT
                        *               A = checksum of linkage table
                        *
                        *       CHANGES
                        *               Y
                        *
                        *       CALLS
                        *               -none-
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

     = CB56             CLT     =       *               ;entry

CB56  A011                      LDY     #17             ;offset to last byte to sum
CB58  A900                      LDA     #0              ;initial sum
CB5A  18                        CLC

CB5B  714A              CLT1    ADC     (ZCHAIN),Y      ;add byte
CB5D  88                        DEY
CB5E  10FB ^CB5B                BPL     CLT1            ;if not done

CB60  6900                      ADC     #0              ;add final carry
CB62  49FF                      EOR     #$FF            ;complement
CB64  60                        RTS                     ;return
```

```
CB65                        FIX     ICSORG



                    **      International Character Set

CC00  0000000000            DB      $00,$00,$00,$00,$00,$00,$00,$00 ;$00 - spac:
CC08  0018181818            DB      $00,$18,$18,$18,$18,$00,$18,$00 ;$01 - !
CC10  0066666600            DB      $00,$66,$66,$66,$00,$00,$00,$00 ;$02 - "
CC18  0066FF6666            DB      $00,$66,$FF,$66,$66,$FF,$66,$00 ;$03 - #
CC20  183E603C06            DB      $18,$3E,$60,$3C,$06,$7C,$18,$00 ;$04 - $
CC28  00666C1830            DB      $00,$66,$6C,$18,$30,$66,$46,$00 ;$05 - %
CC30  1C361C386F            DB      $1C,$36,$1C,$38,$6F,$66,$3B,$00 ;$06 - &
CC38  0018181800            DB      $00,$18,$18,$18,$00,$00,$00,$00 ;$07 - '
CC40  000E1C1818            DB      $00,$0E,$1C,$18,$18,$1C,$0E,$00 ;$08 - (
CC48  0070381818            DB      $00,$70,$38,$18,$18,$38,$70,$00 ;$09 - )
CC50  00663CFF3C            DB      $00,$66,$3C,$FF,$3C,$66,$00,$00 ;$0A - aste:
CC58  0018187E18            DB      $00,$18,$18,$7E,$18,$18,$00,$00 ;$0B - plus
CC60  0000000000            DB      $00,$00,$00,$00,$00,$18,$18,$30 ;$0C - comm:
CC68  0000007E00            DB      $00,$00,$00,$7E,$00,$00,$00,$00 ;$0D - minu:
CC70  0000000000            DB      $00,$00,$00,$00,$00,$18,$18,$00 ;$0E - peri:
CC78  00060C1830            DB      $00,$06,$0C,$18,$30,$60,$40,$00 ;$0F - /

CC80  003C666E76            DB      $00,$3C,$66,$6E,$76,$66,$3C,$00 ;$10 - 0
CC88  0018381818            DB      $00,$18,$38,$18,$18,$18,$7E,$00 ;$11 - 1
CC90  003C660C18            DB      $00,$3C,$66,$0C,$18,$30,$7E,$00 ;$12 - 2
CC98  007E0C180C            DB      $00,$7E,$0C,$18,$0C,$66,$3C,$00 ;$13 - 3
CCA0  000C1C3C6C            DB      $00,$0C,$1C,$3C,$6C,$7E,$0C,$00 ;$14 - 4
CCA8  007E607C06            DB      $00,$7E,$60,$7C,$06,$66,$3C,$00 ;$15 - 5
CCB0  003C607C66            DB      $00,$3C,$60,$7C,$66,$66,$3C,$00 ;$16 - 6
CCB8  007E060C18            DB      $00,$7E,$06,$0C,$18,$30,$30,$00 ;$17 - 7
CCC0  003C663C66            DB      $00,$3C,$66,$3C,$66,$66,$3C,$00 ;$18 - 8
CCC8  003C663E06            DB      $00,$3C,$66,$3E,$06,$0C,$38,$00 ;$19 - 9
CCD0  0000181800            DB      $00,$00,$18,$18,$00,$18,$18,$00 ;$1A - colo:
CCD8  0000181800            DB      $00,$00,$18,$18,$00,$18,$18,$30 ;$1B - semi:
CCE0  060C183018            DB      $06,$0C,$18,$30,$18,$0C,$06,$00 ;$1C - <
CCE8  00007E0000            DB      $00,$00,$7E,$00,$00,$7E,$00,$00 ;$1D - =
CCF0  6030180C18            DB      $60,$30,$18,$0C,$18,$30,$60,$00 ;$1E - >
CCF8  003C660C18            DB      $00,$3C,$66,$0C,$18,$00,$18,$00 ;$1F - ?

CD00  003C666E6E            DB      $00,$3C,$66,$6E,$6E,$60,$3E,$00 ;$20 - @
CD08  00183C6666            DB      $00,$18,$3C,$66,$66,$7E,$66,$00 ;$21 - A
CD10  007C667C66            DB      $00,$7C,$66,$7C,$66,$66,$7C,$00 ;$22 - B
CD18  003C666060            DB      $00,$3C,$66,$60,$60,$66,$3C,$00 ;$23 - C
CD20  00786C6666            DB      $00,$78,$6C,$66,$66,$6C,$78,$00 ;$24 - D
CD28  007E607C60            DB      $00,$7E,$60,$7C,$60,$60,$7E,$00 ;$25 - E
CD30  007E607C60            DB      $00,$7E,$60,$7C,$60,$60,$60,$00 ;$26 - F
CD38  003E60606E            DB      $00,$3E,$60,$60,$6E,$66,$3E,$00 ;$27 - G
CD40  0066667E66            DB      $00,$66,$66,$7E,$66,$66,$66,$00 ;$28 - H
CD48  007E181818            DB      $00,$7E,$18,$18,$18,$18,$7E,$00 ;$29 - I
CD50  0006060606            DB      $00,$06,$06,$06,$06,$66,$3C,$00 ;$2A - J
CD58  00666C7878            DB      $00,$66,$6C,$78,$78,$6C,$66,$00 ;$2B - K
CD60  0060606060            DB      $00,$60,$60,$60,$60,$60,$7E,$00 ;$2C - L
CD68  0063777F6B            DB      $00,$63,$77,$7F,$6B,$63,$63,$00 ;$2D - M
CD70  0066767E7E            DB      $00,$66,$76,$7E,$7E,$6E,$66,$00 ;$2E - N
```

```
CD78  003C666666       DB      $00,$3C,$66,$66,$66,$66,$3C,$00 ;$2F - O

CD80  007C66667C       DB      $00,$7C,$66,$66,$7C,$60,$60,$00 ;$30 - P
CD88  003C666666       DB      $00,$3C,$66,$66,$66,$6C,$36,$00 ;$31 - Q
CD90  007C66667C       DB      $00,$7C,$66,$66,$7C,$6C,$66,$00 ;$32 - R
CD98  003C603C06       DB      $00,$3C,$60,$3C,$06,$06,$3C,$00 ;$33 - S
CDA0  007E181818       DB      $00,$7E,$18,$18,$18,$18,$18,$00 ;$34 - T
CDA8  0066666666       DB      $00,$66,$66,$66,$66,$66,$7E,$00 ;$35 - U
CDB0  0066666666       DB      $00,$66,$66,$66,$66,$3C,$18,$00 ;$36 - V
CDB8  006363687F       DB      $00,$63,$63,$6B,$7F,$77,$63,$00 ;$37 - W
CDC0  0066663C3C       DB      $00,$66,$66,$3C,$3C,$66,$66,$00 ;$38 - X
CDC8  0066663C18       DB      $00,$66,$66,$3C,$18,$18,$18,$00 ;$39 - Y
CDD0  007E0C1830       DB      $00,$7E,$0C,$18,$30,$60,$7E,$00 ;$3A - Z
CDD8  001E181818       DB      $00,$1E,$18,$18,$18,$18,$1E,$00 ;$3B - [
CDE0  0040603018       DB      $00,$40,$60,$30,$18,$0C,$06,$00 ;$3C - \
CDE8  0078181818       DB      $00,$78,$18,$18,$18,$18,$78,$00 ;$3D - ]
CDF0  00081C3663       DB      $00,$08,$1C,$36,$63,$00,$00,$00 ;$3E - ^
CDF8  0000000000       DB      $00,$00,$00,$00,$00,$00,$FF,$00 ;$3F - unde:

CE00  0C183C063E       DB      $0C,$18,$3C,$06,$3E,$66,$3E,$00 ;$40 - acut:
CE08  3018006666       DB      $30,$18,$00,$66,$66,$66,$3E,$00 ;$41 - acut:
CE10  366C007676       DB      $36,$6C,$00,$76,$76,$7E,$6E,$00 ;$42 - tild:
CE18  0C187E607C       DB      $0C,$18,$7E,$60,$7C,$60,$7E,$00 ;$43 - acut:
CE20  00003C6060       DB      $00,$00,$3C,$60,$60,$3C,$18,$30 ;$44 - cedi:
CE28  3C66003C66       DB      $3C,$66,$00,$3C,$66,$66,$3C,$00 ;$45 - circ:
CE30  3018003C66       DB      $30,$18,$00,$3C,$66,$66,$3C,$00 ;$46 - grav:
CE38  3018003818       DB      $30,$18,$00,$38,$18,$18,$3C,$00 ;$47 - grav:
CE40  1C30307830       DB      $1C,$30,$30,$78,$30,$30,$7E,$00 ;$48 - U.K.:
CE48  0066003818       DB      $00,$66,$00,$38,$18,$18,$3C,$00 ;$49 - diae:
CE50  0066006666       DB      $00,$66,$00,$66,$66,$66,$3E,$00 ;$4A - umla:
CE58  36003C063E       DB      $36,$00,$3C,$06,$3E,$66,$3E,$00 ;$4B - umla:
CE60  66003C6666       DB      $66,$00,$3C,$66,$66,$66,$3C,$00 ;$4C - umla:
CE68  0C18006666       DB      $0C,$18,$00,$66,$66,$66,$3E,$00 ;$4D - grav:
CE70  0C18003C66       DB      $0C,$18,$00,$3C,$66,$66,$3C,$00 ;$4E - acut:
CE78  0066003C66       DB      $00,$66,$00,$3C,$66,$66,$3C,$00 ;$4F - umla:

CE80  6600666666       DB      $66,$00,$66,$66,$66,$66,$7E,$00 ;$50 - umla:
CE88  3C661C063E       DB      $3C,$66,$1C,$06,$3E,$66,$3E,$00 ;$51 - circ:
CE90  3C66006666       DB      $3C,$66,$00,$66,$66,$66,$3E,$00 ;$52 - circ:
CE98  3C66003818       DB      $3C,$66,$00,$38,$18,$18,$3C,$00 ;$53 - circ:
CEA0  0C183C667E       DB      $0C,$18,$3C,$66,$7E,$60,$3C,$00 ;$54 - acut:
CEA8  30183C667E       DB      $30,$18,$3C,$66,$7E,$60,$3C,$00 ;$55 - grav:
CEB0  366C007C66       DB      $36,$6C,$00,$7C,$66,$66,$66,$00 ;$56 - tild:
CEB8  3CC33C667E       DB      $3C,$C3,$3C,$66,$7E,$60,$3C,$00 ;$57 - circ:
CEC0  18003C063E       DB      $18,$00,$3C,$06,$3E,$66,$3E,$00 ;$58 - ring:
CEC8  30183C063E       DB      $30,$18,$3C,$06,$3E,$66,$3E,$00 ;$59 - grav:
CED0  1800183C66       DB      $18,$00,$18,$3C,$66,$7E,$66,$00 ;$5A - ring:
CED8  786078607E       DB      $78,$60,$78,$60,$7E,$18,$1E,$00 ;$5B - disp:
CEE0  00183C7E18       DB      $00,$18,$3C,$7E,$18,$18,$18,$00 ;$5C - up a:
CEE8  001818187E       DB      $00,$18,$18,$18,$7E,$3C,$18,$00 ;$5D - down:
CEF0  0018307E30       DB      $00,$18,$30,$7E,$30,$18,$00,$00 ;$5E - left:
CEF8  00180C7E0C       DB      $00,$18,$0C,$7E,$0C,$18,$00,$00 ;$5F - righ:

CF00  1800181818       DB      $18,$00,$18,$18,$18,$18,$18,$00 ;$60 - Span:
CF08  00003C063E       DB      $00,$00,$3C,$06,$3E,$66,$3E,$00 ;$61 - a
CF10  0060607C66       DB      $00,$60,$60,$7C,$66,$66,$7C,$00 ;$62 - b
CF18  00003C6060       DB      $00,$00,$3C,$60,$60,$60,$3C,$00 ;$63 - c
```

```
CF20   0006063E66         DB        $00,$06,$06,$3E,$66,$66,$3E,$00  ;$64 - d
CF28   00003C667E         DB        $00,$00,$3C,$66,$7E,$60,$3C,$00  ;$65 - e
CF30   000E183E18         DB        $00,$0E,$18,$3E,$18,$18,$18,$00  ;$66 - f
CF38   00003E6666         DB        $00,$00,$3E,$66,$66,$3E,$06,$7C  ;$67 - g
CF40   0060607C66         DB        $00,$60,$60,$7C,$66,$66,$66,$00  ;$68 - h
CF48   0018003818         DB        $00,$18,$00,$38,$18,$18,$3C,$00  ;$69 - i
CF50   0006000606         DB        $00,$06,$00,$06,$06,$06,$06,$3C  ;$6A - j
CF58   0060606C78         DB        $00,$60,$60,$6C,$78,$6C,$66,$00  ;$6B - k
CF60   0038181818         DB        $00,$38,$18,$18,$18,$18,$3C,$00  ;$6C - l
CF68   0000667F7F         DB        $00,$00,$66,$7F,$7F,$6B,$63,$00  ;$6D - m
CF70   00007C6666         DB        $00,$00,$7C,$66,$66,$66,$66,$00  ;$6E - n
CF78   00003C6666         DB        $00,$00,$3C,$66,$66,$66,$3C,$00  ;$6F - o

CF80   00007C6666         DB        $00,$00,$7C,$66,$66,$7C,$60,$60  ;$70 - p
CF88   00003E6666         DB        $00,$00,$3E,$66,$66,$3E,$06,$06  ;$71 - q
CF90   00007C6660         DB        $00,$00,$7C,$66,$60,$60,$60,$00  ;$72 - r
CF98   00003E603C         DB        $00,$00,$3E,$60,$3C,$06,$7C,$00  ;$73 - s
CFA0   00187E1818         DB        $00,$18,$7E,$18,$18,$18,$0E,$00  ;$74 - t
CFA8   0000666666         DB        $00,$00,$66,$66,$66,$66,$3E,$00  ;$75 - u
CFB0   0000666666         DB        $00,$00,$66,$66,$66,$3C,$18,$00  ;$76 - v
CFB8   000063687F         DB        $00,$00,$63,$6B,$7F,$3E,$36,$00  ;$77 - w
CFC0   0000663C18         DB        $00,$00,$66,$3C,$18,$3C,$66,$00  ;$78 - x
CFC8   0000666666         DB        $00,$00,$66,$66,$66,$3E,$0C,$78  ;$79 - y
CFD0   00007E0C18         DB        $00,$00,$7E,$0C,$18,$30,$7E,$00  ;$7A - z
CFD8   6666183C66         DB        $66,$66,$18,$3C,$66,$7E,$66,$00  ;$7B - umla:
CFE0   1818181818         DB        $18,$18,$18,$18,$18,$18,$18,$18  ;$7C - |
CFE8   007E787C6E         DB        $00,$7E,$78,$7C,$6E,$66,$06,$00  ;$7D - disp:
CFF0   0818387838         DB        $08,$18,$38,$78,$38,$18,$08,$00  ;$7E - disp:
CFF8   10181C1E1C         DB        $10,$18,$1C,$1E,$1C,$18,$10,$00  ;$7F - disp:
```

```
D000                         FIX     $D000
D000   = 5000#               LOC     $5000    ;$D000 - $D7FF mapped to $5000 - $5:


                    **       STH - Self-test Hardware
                    *
                    *        ENTRY   JSR     STH
                    *
                    *        NOTES
                    *                Problem: this is superfluous; SLFTSV could :
                    *                EST.
                    *
                    *        MODS
                    *                M. W. Colburn   10/26/82
                    *                1. Bring closer to Coding Standard (object :
                    *                   R. K. Nordin 11/01/83

       = 5000       STH      =       *       ;entry
5000# 4C0950                 JMP     EST     ;execute self-test


                    **       EMS - Execute Memory Self-test
                    *
                    *        ENTRY   JSR     EMS
                    *
                    *        MODS
                    *                M. W. Colburn   10/26/82
                    *                1. Bring closer to Coding Standard (object :
                    *                   R. K. Nordin 11/01/83

       = 5003       EMS      =       *       ;entry
5003# 208650                 JSR     IST     ;initialize self-test
5006# 4C9152                 JMP     STM     ;self-test memory


                    **       EST - Execute Self-test
                    *
                    *        ENTRY   JSR     EST
                    *
                    *        MODS
                    *                M. W. Colburn   10/26/82
                    *                1. Bring closer to Coding Standard (object :
                    *                   R. K. Nordin 11/01/83

       = 5009       EST      =       *       ;entry
5009# 208650                 JSR     IST     ;initialize self-test
                    ;        JMP     SEL     ;self-test
```

```
                        **      SEL - Self-test
                        *
                        *       ENTRY   JSR     SEL
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                 R. K. Nordin 11/01/83


        = 500C          SEL     =       *               ;entry

                        ;       Initialize.

500C# A900                      LDA     #0
500E# 8580                      STA     STTIME          ;clear main screen timeout :
5010# 8581                      STA     STTIME+1
5012# 8582                      STA     STAUT           ;clear auto-mode flag
5014# 8D08D2                    STA     AUDCTL          ;initialize audio control r:
5017# A903                      LDA     #$03            ;initialize POKEY
5019# 8D0FD2                    STA     SKCTL           ;serial port control
501C# 201055                    JSR     SAS             ;silence all sounds
501F# A940                      LDA     #$40            ;disable DLI
5021# 8D0ED4                    STA     NMIEN           ;NMI enable
5024# A200                      LDX     #0              ;main screen colors
5026# 207357                    JSR     SUC             ;set up colors
5029# A23A                      LDX     #low DISL1      ;display list for main scre:
502B# A051                      LDY     #high DISL1
502D# 209E50                    JSR     SDL             ;set up display list
5030# A9D0                      LDA     #low PMD        ;process main screen DLI ro:
5032# 8D0002                    STA     VDSLST          ;display list NMI address
5035# A950                      LDA     #high PMD
5037# 8D0102                    STA     VDSLST+1
503A# A20C                      LDX     #3*4            ;main screen bold lines
503C# A9AA                      LDA     #$AA            ;color 1
503E# 202A57                    JSR     SVR             ;set value in range

                        ;       Wait for all screen DLI's to clear and for VBLANK.

5041# A200                      LDX     #0

5043# 8E0AD4            SEL1    STX     WSYNC           ;wait for HBLANK synchroniz:
5046# E8                        INX
5047# D0FA ^5043                BNE     SEL1            ;if not done waiting

                        ;       Wait until beam close to top (main screen DLI near):

5049# AD0BD4            SEL2    LDA     VCOUNT
504C# C918                      CMP     #24
504E# B0F9 ^5049                BCS     SEL2            ;if not done waiting

                        ;       Preset for self-test type determination.

5050# A910                      LDA     #$10            ;initially select memory te:
5052# 8587                      STA     STPASS          ;pass indicator
5054# A9C0                      LDA     #$C0            ;enable DLI
5056# 8D0ED4                    STA     NMIEN
```

```
                        ;       Determine type of self-test.

5059# AD1FD0    SEL3    LDA     CONSOL          ;console switches
505C# 2901              AND     #$01            ;START key indicator
505E# D0F9 ^5059        BNE     SEL3            ;if START key not pressed

5060# A9FF              LDA     #$FF            ;clear character
5062# 8DFC02            STA     CH

5065# A586              LDA     STSEL           ;selection
5067# 290F              AND     #$0F            ;selection
5069# C901              CMP     #$01            ;memory test indicator
506B# F010 ^507D        BEQ     SEL5            ;if memory test

506D# C902              CMP     #$02
506F# F00F ^5080        BEQ     SEL6            ;if audio-visual test

5071# C904              CMP     #$04
5073# F00E ^5083        BEQ     SEL7            ;if keyboard test

                        ;       Self-test all.

5075# A988      SEL4    LDA     #$88            ;indicate all tests
5077# 8586              STA     STSEL           ;selection
5079# A9FF              LDA     #$FF            ;auto-mode indicator
507B# 8582              STA     STAUT           ;auto-mode flag

                        ;       Self-test memory.

507D# 4C9152    SEL5    JMP     STM             ;self-test memory

                        ;       Self-test audio-visual.

5080# 4C5755    SEL6    JMP     STV             ;self-test audio-visual

                        ;       Self-test keyboard.

5083# 4C5054    SEL7    JMP     STK             ;self-test keyboard



                        **      IST - Initialize Self-test
                        *
                        *       ENTRY   JSR     IST
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = 5086    IST     =       *       ;entry
5086# A911              LDA     #$11    ;indicate memory test
5088# 8586              STA     STSEL   ;selection
508A# A921              LDA     #$21
```

```
508C# 8D2F02              STA     SDMCTL  ;select small size playfield
508F# A9C0                LDA     #$C0
5091# 8D0ED4              STA     NMIEN   ;enable DLI
5094# A941                LDA     #$41
5096# 8583                STA     STJMP   ;ANTIC jump instruction
5098# A9FF                LDA     #$FF    ;clear code indicator
509A# 8DFC02              STA     CH      ;key code
509D# 60                  RTS             ;return



                  **      SDL - Set Up Display List
                  *
                  *       ENTRY   JSR     SDL
                  *
                  *       MODS
                  *               M. W. Colburn   10/26/82
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


      = 509E      SDL     =       *       ;entry

509E# 858A                STA     STKST           ;keyboard self-test flag
50A0# 98                  TYA
50A1# 48                  PHA                     ;save high address
50A2# 8A                  TXA
50A3# 48                  PHA                     ;save low address
50A4# A900                LDA     #0
50A6# 8D2F02              STA     SDMCTL          ;DMACTL (DMA control) shado:
50A9# 8DDC02              STA     HELPFG          ;HELP key flag
50AC# A9DA                LDA     #low POD        ;process DLI routine
50AE# 8D0002              STA     VDSLST
50B1# A953                LDA     #high POD
50B3# 8D0102              STA     VDSLST+1
50B6# A200                LDX     #0*4            ;screen memory
50B8# 8A                  TXA                     ;value is 0
50B9# 202A57              JSR     SVR             ;set value in range
50BC# 68                  PLA                     ;saved low address
50BD# AA                  TAX
50BE# 68                  PLA                     ;saved high address
50BF# A8                  TAY
50C0# 8E3002              STX     SDLSTL          ;low display list address
50C3# 8684                STX     STJMP+1         ;low display list address
50C5# 8C3102              STY     SDLSTH          ;high display list address
50C8# 8485                STY     STJMP+2         ;high display list address
50CA# A921                LDA     #$21
50CC# 8D2F02              STA     SDMCTL
50CF# 60                  RTS             ;return
```

```
                        **      PMD - Process Main Screen DLI
                        *
                        *       1) IF MAIN SCREEN IS ON FOR MORE than FIVE MINUTES
                        *       THEN 'ALL TESTS' SELECTION IS SELECTED AND EXECUTED
                        *       2) COLORS FOR CURRENTLY SELECTED CHOICE AND THE
                        *       NON-SELECTED CHOICES ARE DISPLAYED ON FLY
                        *       3) SELECTION PROCESS IS HANDLED
                        *
                        *       ENTRY   JMP     PMD
                        *
                        *       EXIT
                        *               Exits via RTI
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                 R. K. Nordin 11/01/83


       = 50D0           PMD     =       *       ;entry

                        ;       Initialize.

50D0# 48                        PHA             ;save A
50D1# 8A                        TXA
50D2# 48                        PHA             ;save X

                        ;       Check for 4th time.

50D3# A27A              PMD1    LDX     #$7A    ;assume non-selected color
50D5# A587                      LDA     STPASS  ;pass indicator
50D7# C901                      CMP     #$01    ;4th time indicator
50D9# F01F ^50FA                BEQ     PMD3    ;if 4th time

                        ;       Check for selection.

50DB# 2901                      AND     #$01    ;selection indicatorn
50DD# F00A ^50E9                BEQ     PMD2    ;if selected

                        ;       Increment and check blink counter.

50DF# E6A2                      INC     STBL    ;increment blink counter
50E1# A5A2                      LDA     STBL    ;blink counter
50E3# 2920                      AND     #$20    ;blink indicator
50E5# F002 ^50E9                BEQ     PMD2    ;if not to blink

50E7# A22C                      LDX     #$2C    ;use selected color

                        ;       Set color.

50E9# 8E0AD4            PMD2    STX     WSYNC   ;wait for HBLANK synchronization
50EC# 8E16D0                    STX     COLPF0  ;playfield 0 color
50EF# 18                        CLC
50F0# 6687                      ROR     STPASS  ;advance pass indicator
50F2# A900                      LDA     #0
50F4# 854D                      STA     ATRACT
```

```
                          ;       Exit.

50F6# 68                          PLA
50F7# AA                          TAX                 ;restore X
50F8# 68                          PLA                 ;restore A
50F9# 40                          RTI                 ;return

                          ;       Check for SELECT previously pressed.

50FA# A588                PMD3    LDA      STSPP      ;SELECT previously pressed  flag
50FC# D016 ^5114                  BNE      PMD4       ;if SELECT previously pressed

                          ;       Check for SELECT pressed.

50FE# AD1FD0                      LDA      CONSOL     ;console switches
5101# 2902                        AND      #$02       ;SELECT key indicator
5103# D01A ^511F                  BNE      PMD5       ;if SELECT not pressed, exit

                          ;       Process SELECT pressed.

5105# A586                        LDA      STSEL      ;current selection
5107# 2A                          ROL      A
5108# 2686                        ROL      STSEL      ;next selection
510A# A920                        LDA      #$20       ;blink indicator
510C# 85A2                        STA      STBL       ;blink counter
510E# A9FF                        LDA      #$FF       ;SELECT previously pressed indicato
5110# 8588                        STA      STSPP      ;SELECT previously pressed flag
5112# D00B ^511F                  BNE      PMD5

                          ;       Process SELECT previously pressed.

5114# AD1FD0              PMD4    LDA      CONSOL     ;console switches
5117# 2902                        AND      #$02       ;SELECT key indicator
5119# F004 ^511F                  BEQ      PMD5       ;if SELECT still pressed

511B# A900                        LDA      #0         ;SELECT not previously pressed indi
511D# 8588                        STA      STSPP      ;SELECT previously pressed flag

511F# A586                PMD5    LDA      STSEL      ;selection
5121# 290F                        AND      #$0F
5123# 0910                        ORA      #$10       ;reset indicate memory test
5125# 8587                        STA      STPASS     ;pass indicator

                          ;       Advance main screen timer.

5127# E680                        INC      STTIME
5129# D002 ^512D                  BNE      PMD6       ;if low not zero

512B# E681                        INC      STTIME+1

                          ;       Check main screen timer.

512D# A581                PMD6    LDA      STTIME+1
512F# C9FA                        CMP      #250       ;main screen timeout
5131# D004 ^5137                  BNE      PMD7       ;if main screen timed out

                          ;       Process main screen timeout.
```

```
5133# 58                        CLI
5134# 4C7550                    JMP      SEL4      ;self-test all

                      ;         Continue.

5137# 4CD350          PMD7      JMP      PMD1      ;continue



                      **        DISL1 - Display List for Main Screen

513A# 7070707070      DISL1     DB       $70,$70,$70,$70,$70
513F# 47                        DB       $47
5140# 6151                      DW       SMEM1
5142# 707070                    DB       $70,$70,$70
5145# 4E                        DB       $4E
5146# 0030                      DW       ST3000
5148# 70                        DB       $70
5149# F0                        DB       $F0
514A# C6                        DB       $C6
514B# 7151                      DW       SMEM2
514D# 7086                      DB       $70,$86
514F# 7086                      DB       $70,$86
5151# 7006                      DB       $70,$06
5153# 7070                      DB       $70,$70
5155# 4E                        DB       $4E
5156# 0030                      DW       ST3000
5158# 707070                    DB       $70,$70,$70
515B# 42                        DB       $42
515C# 8151                      DW       SMEM3
515E# 41                        DB       $41
515F# 3A51                      DW       DISL1



                      **        SMEM1 - "SELF TEST" Text

5161# 00000000        SMEM1     DB       $00,$00,$00,$00
5165# 33252C26                  DB       $33,$25,$2C,$26           ;"SELF"
5169# 00                        DB       $00
516A# 34253334                  DB       $34,$25,$33,$34           ;"TEST"
516E# 000000                    DB       $00,$00,$00
```

```
                    **      SMEM2 - "MEMORY AUDIO-VISUAL KEYBOARD ALL TESTS" Te:


5171# 0000          SMEM2   DB      $00,$00
5173# 2D252D2F32            DB      $2D,$25,$2D,$2F,$32,$39                  ;"M:
5179# 0000000000            DB      $00,$00,$00,$00,$00
517E# 0000000000            DB      $00,$00,$00,$00,$00
5183# 213524292F            DB      $21,$35,$24,$29,$2F                      ;"A:
5188# 0D                    DB      $0D                                      ;"-:
5189# 3629333521            DB      $36,$29,$33,$35,$21,$2C                  ;"V:
518F# 00000000              DB      $00,$00,$00,$00
5193# 2B2539222F            DB      $2B,$25,$39,$22,$2F,$21,$32,$24          ;"K:
519B# 0000000000            DB      $00,$00,$00,$00,$00,$00,$00,$00
51A3# 212C2C                DB      $21,$2C,$2C                              ;"A:
51A6# 00                    DB      $00
51A7# 3425333433            DB      $34,$25,$33,$34,$33                      ;"T:
51AC# 0000000000            DB      $00,$00,$00,$00,$00


                    **      SMEM3 - "SELECT,START OR RESET" Text


51B1# 00000000      SMEM3   DB      $00,$00,$00,$00
51B5# 42                    DB      $42
51B6# B3A5ACA5A3            DB      $B3,$A5,$AC,$A5,$A3,$B4          ;"SELECT"
51BC# 56                    DB      $56
51BD# 0C                    DB      $0C                              ;","
51BE# 42                    DB      $42
51BF# B3B4A1B2B4            DB      $B3,$B4,$A1,$B2,$B4              ;"START"
51C4# 56                    DB      $56
51C5# 2F32                  DB      $2F,$32                          ;"OR"
51C7# 42                    DB      $42
51C8# B2A5B3A5B4            DB      $B2,$A5,$B3,$A5,$B4              ;"RESET"
51CD# 56                    DB      $56
51CE# 000000                DB      $00,$00,$00


                    **      DISL2 - Display List for Memory Test


51D1# 707070        DISL2   DB      $70,$70,$70
51D4# 46                    DB      $46
51D5# 0030                  DW      ST3000
51D7# 70                    DB      $70
51D8# 7006                  DB      $70,$06
51DA# 7008                  DB      $70,$08
51DC# 70                    DB      $70
51DD# 7006                  DB      $70,$06
51DF# 7008                  DB      $70,$08
51E1# 7008                  DB      $70,$08
51E3# 7008                  DB      $70,$08
51E5# 7008                  DB      $70,$08
51E7# 707070                DB      $70,$70,$70
```

```
51EA# 01                          DB      $01
51EB# ED51                        DW      DISL3


                      **          DISL3 - Display List for Exit Text


51ED# A040            DISL3       DB      $A0,$40
51EF# 42                          DB      $42
51F0# F551                        DW      SMEM4
51F2# 01                          DB      $01
51F3# 8300                        DW      STJMP


                      **          SMEM4 - "RESET OR HELP TO EXIT" Text


51F5# 0000000000      SMEM4       DB      $00,$00,$00,$00,$00
51FA# 42                          DB      $42
51FB# B2A5B3A5B4                  DB      $B2,$A5,$B3,$A5,$B4      ;"RESET"
5200# 56                          DB      $56
5201# 2F32                        DB      $2F,$32                  ;"OR"
5203# 42                          DB      $42
5204# A8A5ACB0                    DB      $A8,$A5,$AC,$B0          ;"HELP"
5208# 56                          DB      $56
5209# 342F                        DB      $34,$2F                  ;"TO"
520B# 00                          DB      $00
520C# 25382934                    DB      $25,$38,$29,$34          ;"EXIT"
5210# 0000000000                  DB      $00,$00,$00,$00,$00


                      **          DISL4 - Display List for Keyboard Test


5215# 70707070        DISL4       DB      $70,$70,$70,$70
5219# 46                          DB      $46
521A# 0030                        DW      ST3000
521C# 707070                      DB      $70,$70,$70
521F# 7002                        DB      $70,$02
5221# 70                          DB      $70
5222# 7002                        DB      $70,$02
5224# 7002                        DB      $70,$02
5226# 7002                        DB      $70,$02
5228# 7002                        DB      $70,$02
522A# 7002                        DB      $70,$02
522C# 7070                        DB      $70,$70
522E# 01                          DB      $01
522F# ED51                        DW      DISL3
```

```
                        **      DISL5 - Display List for Audio-visual Test


5231# 70707070      DISL5   DB      $70,$70,$70,$70
5235# 46                    DB      $46
5236# 7152                  DW      SMEM5
5238# 7006                  DB      $70,$06
523A# 7070                  DB      $70,$70
523C# 48                    DB      $48
523D# 0031                  DW      ST3100
523F# 0B0B0B0B0B0B          DB      $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
5247# 0B0B0B0B0B0B          DB      $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
524F# 0B0B0B0B0B0B          DB      $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
5257# 0B0B0B0B0B0B          DB      $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
525F# 0B0B0B0B0B0B          DB      $0B,$0B,$0B,$0B,$0B,$0B,$0B,$0B
5267# 0B0B                  DB      $0B,$0B
5269# 70                    DB      $70
526A# 46                    DB      $46
526B# 0030                  DW      ST3000
526D# 70                    DB      $70
526E# 01                    DB      $01
526F# ED51                  DW      DISL3



                        **      SMEM5 - "AUDIO-VISUAL TEST" Text


5271# 0000          SMEM5   DB      $00,$00
5273# 213524292F            DB      $21,$35,$24,$29,$2F             ;"AUDIO"
5278# 0D                    DB      $0D                             ;"-"
5279# 3629333521            DB      $36,$29,$33,$35,$21,$2C         ;"VISUAL"
527F# 00000000              DB      $00,$00,$00,$00
5283# 00000000              DB      $00,$00,$00,$00
5287# 34253334              DB      $34,$25,$33,$34                 ;"TEST"
528B# 0000000000            DB      $00,$00,$00,$00,$00,$00



                        **      STM - Self-test Memory
                        *
                        *       STM verifies ROM and RAM by verifying the ROM check:
                        *       writing and reading all possible values to each byt:
                        *
                        *       ENTRY   JSR     STM
                        *
                        *       NOTES
                        *               Problem: searches beyond end of TMNT.
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83
```

```
      = 5291        STM       =        *                 ;entry

                    ;         Initialize.

5291# A201                    LDX      #low DISL2        ;memory test display list
5293# A051                    LDY      #high DISL2
5295# A900                    LDA      #0                ;indicate not keyboard self:
5297# 209E50                  JSR      SDL               ;set up display list
529A# A201                    LDX      #1                ;memory test colors
529C# 207357                  JSR      SUC               ;set up colors
529F# A200                    LDX      #0                ;offset to "MEMORY TEST   R:
52A1# 205957                  JSR      SSM               ;set screen memory
52A4# A201                    LDX      #1                ;offset to "RAM" text
52A6# 205957                  JSR      SSM               ;set screen memory

                    ;         Test first 8K ROM.

52A9# AD2030        STM1      LDA      ST3020
52AC# C9AA                    CMP      #$AA              ;color 1 for failure
52AE# F017 ^52C7              BEQ      STM4              ;if first 8K ROM already fa:

52B0# A955                    LDA      #$55              ;color 0 for test
52B2# 208E53                  JSR      DFS               ;display first ROM status
52B5# 208153                  JSR      DMW               ;delay a middling while
52B8# 2073FF                  JSR      VFR               ;verify first 8K ROM
52BB# B005 ^52C2              BCS      STM2              ;if ROM failed

52BD# A9FF                    LDA      #$FF              ;color 2 for success
52BF# 4CC452                  JMP      STM3

52C2# A9AA          STM2      LDA      #$AA              ;color 1 for failure

52C4# 208E53        STM3      JSR      DFS               ;display first ROM status

                    ;         Test second 8K ROM.

52C7# AD2430        STM4      LDA      ST3024
52CA# C9AA                    CMP      #$AA              ;color 1 for failure
52CC# F017 ^52E5              BEQ      STM7              ;if second 8K ROM already f:

52CE# A955                    LDA      #$55              ;color 0 for test
52D0# 209953                  JSR      DSS               ;display second ROM status
52D3# 208153                  JSR      DMW               ;delay a middling while
52D6# 2092FF                  JSR      VSR               ;verify second 8K ROM
52D9# B005 ^52E0              BCS      STM5              ;if ROM failed

52DB# A9FF                    LDA      #$FF              ;color 2 for success
52DD# 4CE252                  JMP      STM6

52E0# A9AA          STM5      LDA      #$AA              ;color 1 for failure

52E2# 209953        STM6      JSR      DSS               ;display second ROM status

                    ;         Test RAM.

52E5# A9C0          STM7      LDA      #$C0              ;mask for left side of a sc:
```

```
52E7# 858D                      STA      STSMM
52E9# A904                      LDA      #$04            ;initially select LED 1 off
52EB# 85A4                      STA      STLM            ;LED mask
52ED# A900                      LDA      #0
52EF# 858E                      STA      STSMP
52F1# 8590                      STA      STPAG           ;initialize current page
52F3# 8591                      STA      STPAG+1
52F5# 858F                      STA      ST1K            ;initialize current 1K to t:

                       ;        Test 1K of RAM.

52F7# A68E             STM8     LDX      STSMP           ;screen memory pointer
52F9# BD3830                    LDA      ST3038,X
52FC# 258D                      AND      STSMM
52FE# C980                      CMP      #$80
5300# F05C ^535E                BEQ      STM17           ;if already failed

5302# C908                      CMP      #$08
5304# F058 ^535E                BEQ      STM17           ;if already failed

5306# A944                      LDA      #$44            ;color 0 for test
5308# 20C353                    JSR      DRS             ;display RAM block status
530B# A5A4                      LDA      STLM            ;LED mask
530D# 20A453                    JSR      SLD             ;set LED's
5310# A5A4                      LDA      STLM            ;current LED mask
5312# 490C                      EOR      #$0C            ;complement LED's selected
5314# 85A4                      STA      STLM            ;update LED mask

                       ;        Check for memory not to test.

5316# A207                      LDX      #TMNTL-1+2      ;2 bytes beyond last byte o:

5318# BD4A54           STM9     LDA      TMNT,X          ;range to test
531B# C591                      CMP      STPAG+1         ;high current page
531D# F037 ^5356                BEQ      STM15           ;if not to test, indicate s:

531F# CA                        DEX
5320# 10F6 ^5318                BPL      STM9            ;if not done

                       ;        Test 1K of RAM.

5322# A904                      LDA      #4              ;number of pages to test
5324# 8592                      STA      STPC            ;page count

                       ;        Write initial list to page.

5326# A200             STM10    LDX      #0              ;initial value to write

                       ;        Write list to page.

5328# A000             STM11    LDY      #0              ;offset to first byte of pa:

532A# 8A               STM12    TXA
532B# 9190                      STA      (STPAG),Y       ;byte of page
532D# E8                        INX
532E# C8                        INY
532F# D0F9 ^532A                BNE      STM12           ;if not done writing page
```

```
                       ;       Verify list written to page.

5331# 8693                     STX     STMVAL           ;first correct value to tes
5333# A000                     LDY     #0               ;offset to first byte of pa

5335# B190             STM13   LDA     (STPAG),Y        ;byte of page
5337# C593                     CMP     STMVAL           ;correct value
5339# D010 ^534B               BNE     STM14            ;if not correct value

533B# E693                     INC     STMVAL           ;increment value to test
533D# C8                       INY
533E# D0F5 ^5335               BNE     STM13            ;if not done verifying page

                       ;       Increment and test initial value to write.

5340# E8                       INX                      ;increment initial value to
5341# D0E5 ^5328               BNE     STM11            ;if not done, write another

                       ;       Decrement and test page counter.

5343# E691                     INC     STPAG+1          ;increment high current pag
5345# C692                     DEC     STPC             ;decrement page count
5347# D0DD ^5326               BNE     STM10            ;if not done testing pages

5349# F00E ^5359               BEQ     STM16            ;indicate success

                       ;       Display failure.

534B# 208153           STM14   JSR     DMW              ;delay a middling while
534E# A988                     LDA     #$88             ;color 1 for failure
5350# 20C353                   JSR     DRS              ;display RAM block status
5353# 4C5E53                   JMP     STM17

                       ;       Delay for simulating test of memory not to test.

5356# 208553           STM15   JSR     DLW              ;delay a long while

                       ;       Display success.

5359# A9CC             STM16   LDA     #$CC             ;color 2 for success
535B# 20C353                   JSR     DRS              ;display RAM block status

535E# A58D             STM17   LDA     STSMM
5360# 3026 ^5388               BMI     STM20

5362# A9C0                     LDA     #$C0
5364# 858D                     STA     STSMM
5366# E68E                     INC     STSMP            ;increment screen memory po

5368# 18               STM18   CLC
5369# A58F                     LDA     ST1K             ;current 1K to test
536B# 6904                     ADC     #high $0400      ;add 1K
536D# 8591                     STA     STPAG+1          ;high current page
536F# 858F                     STA     ST1K             ;update current 1K to test
5371# CDE402                   CMP     RAMSIZ           ;RAM size
5374# D081 ^52F7               BNE     STM8             ;if not done testing RAM
```

```
                        ;       Check for auto-mode.

5376# A582                      LDA     STAUT           ;auto-mode flag
5378# D003 ^537D                BNE     STM19           ;if auto-mode, perform audi:

                        ;       Test memory again.

537A# 4CA952                    JMP     STM1            ;test memory again

                        ;       Process auto-mode.

537D# A90C              STM19   LDA     #$0C    ;indicate LED 1 and 2 off
537F# 20A453                    JSR     SLD     ;set LED's
5382# 20B553                    JSR     DLW     ;delay a long while
5385# 4C5755                    JMP     STV     ;self-test audio-visual

5388# A90C              STM20   LDA     #$0C
538A# 858D                      STA     STSMM
538C# D0DA ^5368                BNE     STM18


                        **      DFS - Display First ROM Status
                        *
                        *       ENTRY   JSR     DFS
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = 538E            DFS     =       *               ;entry
538E# A204                      LDX     #1*4            ;first 8K ROM display
5390# 202A57                    JSR     SVR             ;set value in range
5393# 29FC                      AND     #$FC
5395# 8D2330                    STA     ST3020+3
5398# 60                        RTS                     ;return


                        **      DSS - Display Second ROM Status
                        *
                        *       ENTRY   JSR     DSS
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = 5399            DSS     =       *               ;entry
5399# A208                      LDX     #2*4            ;second 8K ROM display
539B# 202A57                    JSR     SVR             ;set value in range
```

```
539E# 29FC                  AND     #$FC
53A0# 8D2730                STA     ST3024+3
53A3# 60                    RTS                     ;return



                    **      SLD - Set LED's
                    *
                    *       ENTRY   JSR     SLD
                    *               A = LED mask (bit 3 - LED 2, bit 2 - LED 1)
                    *
                    *       MODS
                    *               M. W. Colburn   10/26/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = 53A4        SLD     =       *               ;entry
53A4# 85A5                  STA     STTMP5          ;save LED mask
53A6# AD01D3                LDA     PORTB
53A9# 29F3                  AND     #$F3            ;clear LED control
53AB# 05A5                  ORA     STTMP5          ;set LED control according :
53AD# 8D01D3                STA     PORTB           ;update port B memory contr:
53B0# 60                    RTS                     ;return



                    **      DMW - Delay a Middling While
                    *
                    *       ENTRY   JSR     DMW
                    *
                    *       MODS
                    *               M. W. Colburn   10/26/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = 53B1        DMW     =       *               ;entry
53B1# A23C                  LDX     #60             ;60-VBLANK delay
53B3# D002 ^53B7            BNE     DAW             ;delay a while
```

```
                        **      DLW - Delay a Long While
                        *
                        *       ENTRY   JSR     DLW
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = 53B5         DLW     =       *               ;entry
53B5# A296                      LDX     #150            ;150-VBLANK delay
                        ;       JMP     DAW             ;delay a while, return


                        **      DAW - Delay a While
                        *
                        *       ENTRY   JSR     DAW
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = 53B7         DAW     =       *               ;entry

53B7# A0FF              DAW1    LDY     #$FF            ;initialize inner loop coun:

53B9# 8C0AD4            DAW2    STY     WSYNC           ;wait for HBLANK synchroniz:
53BC# 88                        DEY
53BD# D0FA ^53B9                BNE     DAW2            ;if inner loop not done

53BF# CA                        DEX
53C0# D0F5 ^53B7                BNE     DAW1            ;if outer loop not done

53C2# 60                        RTS                     ;return


                        **      DRS - Display RAM Block Status
                        *
                        *       ENTRY   JSR     DRS
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = 53C3         DRS     =       *               ;entry
53C3# 48                        PHA                     ;save color
53C4# A68E                      LDX     STSMP
53C6# A58D                      LDA     STSMM
```

```
53C8# 49FF                      EOR     #$FF            ;complement
53CA# 3D3830                    AND     ST3038,X
53CD# 9D3830                    STA     ST3038,X
53D0# 68                        PLA                     ;saved color
53D1# 258D                      AND     STSMM
53D3# 1D3830                    ORA     ST3038,X
53D6# 9D3830                    STA     ST3038,X
53D9# 60                        RTS                     ;return


                    **          POD - Process Other DLI's
                    *
                    *           POD turns the last line on the screen into white on:
                    *           handles keyboard self-test display of console switc:
                    *           HELP key for exit, and ensures no attract-mode.
                    *
                    *           ENTRY   JMP     POD
                    *
                    *           EXIT
                    *                   Exits via RTI
                    *
                    *           MODS
                    *                   M. W. Colburn   10/26/82
                    *                   1. Bring closer to Coding Standard (object :
                    *                      R. K. Nordin 11/01/83

      = 53DA        POD         =       *       ;entry

                    ;           Initialize.

53DA# 48                        PHA             ;save A

                    ;           Select colors.

53DB# A90C                      LDA     #$0C    ;white color
53DD# 8D17D0                    STA     COLPF1  ;playfield 1 color
53E0# ADC802                    LDA     COLOR4  ;background color
53E3# 8D18D0                    STA     COLPF2  ;playfield 2 color

                    ;           Ensure no attract-mode.

53E6# A900                      LDA     #0      ;no attract-mode
53E8# 854D                      STA     ATRACT  ;attract-mode timer/flag

                    ;           Check HELP key.

53EA# ADDC02                    LDA     HELPFG  ;HELP key flag
53ED# F00E ^53FD                BEQ     POD1    ;if HELP not pressed

                    ;           Process HELP key.

53EF# A900                      LDA     #0      ;HELP key not pressed indicator
53F1# 8DDC02                    STA     HELPFG  ;HELP key flag
53F4# A90C                      LDA     #$0C    ;LED's off
```

```
53F6# 20A453                JSR     SLD      ;set LED's
53F9# 58                    CLI
53FA# 4C0C50                JMP     SEL      ;start over with main screen

                    ;       Check for keyboard self-test.

53FD# A58A          POD1    LDA     STKST    ;keyboard self-test flag
53FF# F047 ^5448            BEQ     POD10    ;if not keyboard self-test, exit

                    ;       Set display of console switches pressed.

5401# AD1FD0                LDA     CONSOL   ;console switches
5404# 2901                  AND     #$01     ;START key indicator
5406# F004 ^540C            BEQ     POD2     ;if START key pressed

5408# A9B3                  LDA     #$B3
540A# D002 ^540E            BNE     POD3     ;set display

540C# A933          POD2    LDA     #$33

540E# 8D1C30        POD3    STA     ST301C   ;set START key display

5411# AD1FD0                LDA     CONSOL   ;console switches
5414# 2902                  AND     #$02     ;SELECT key indicator
5416# F004 ^541C            BEQ     POD4     ;if SELECT key pressed

5418# A9F3                  LDA     #$F3
541A# D002 ^541E            BNE     POD5     ;set display

541C# A973          POD4    LDA     #$73

541E# 8D1E30        POD5    STA     ST301E   ;set SELECT key display

5421# AD1FD0                LDA     CONSOL   ;console switches
5424# 2904                  AND     #$04     ;OPTION key indicator
5426# F004 ^542C            BEQ     POD6     ;if OPTION key pressed

5428# A9AF                  LDA     #$AF
542A# D002 ^542E            BNE     POD7     ;set display

542C# A92F          POD6    LDA     #$2F

542E# 8D2030        POD7    STA     ST3020   ;set OPTION key display

                    ;       Sound tone if console switches pressed.

5431# AD1FD0                LDA     CONSOL   ;console switches
5434# 2907                  AND     #$07     ;key indicators
5436# C907                  CMP     #$07     ;no keys pressed
5438# F009 ^5443            BEQ     POD8     ;if no keys pressed

543A# A964                  LDA     #100     ;frequency
543C# 8D02D2                STA     AUDF2    ;set frequency of voice 2
543F# A9A8                  LDA     #$A8     ;pure tone, half volume
5441# D002 ^5445            BNE     POD9     ;set control of voice 2

5443# A900          POD8    LDA     #0       ;zero volume
```

```
5445# 8D03D2        POD9    STA     AUDC2   ;set control of voice 2

                    ;       Exit.

5448# 68            POD10   PLA             ;restore A
5449# 40                    RTI             ;return



                    **      TMNT - Table of Memory Not to Test
                    *
                    *       NOTES
                    *               Problem: bytes wasted by redundant entries.


544A# 00            TMNT    DB      high $0000      ;$0000 - $03FF, zero page a:
544B# 50                    DB      high $5000      ;$5000 - $53FF, self-test R:
544C# 54                    DB      high $5400      ;$5400 - $57FF, self-test R:
544D# 30                    DB      high ST3000     ;ST3000 - ST3000+$03FF, scr:
544E# 30                    DB      high ST3000     ;ST3000 - ST3000+$03FF, scr:
544F# 30                    DB      high ST3000     ;ST3000 - ST3000+$03FF, scr:

      = 0006        TMNTL   =       *-TMNT  ;length



                    **      STK - Self-test Keyboard
                    *
                    *       STK verifies the operation of the keyboard by displ:
                    *       keys as they are pressed.  In auto-mode, the verifi:
                    *       is simulated.
                    *
                    *       ENTRY   JSR     STK
                    *
                    *       NOTES
                    *               Problem: one too many bytes taken from TSKP:
                    *               Problem: wasted bytes for extra LDA CH.
                    *               Problem: logic is convoluted (due to SBT an:
                    *               subroutines appearing in the middle of STK):
                    *
                    *       MODS
                    *               M. W. Colburn   10/26/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = 5450        STK     =       *               ;entry

                    ;       Initialize.

5450# A200                  LDX     #0
5452# 8694                  STX     STSKP           ;initialize simulated keypr:
5454# A203                  LDX     #3              ;keyboard test colors
```

```
5456# 207357                JSR     SUC             ;set up colors
5459# A215                  LDX     #low DISL4      ;keyboard display list
545B# A052                  LDY     #high DISL4
545D# A9FF                  LDA     #$FF            ;indicate keyboard self-tes:
545F# 209E50                JSR     SDL             ;set up display list

                  ;         Test keyboard.

5462# A202        STK1      LDX     #2              ;offset to "KEYBOARD TEST" :
5464# 205957                JSR     SSM             ;set screen memory
5467# A207                  LDX     #7              ;offset to keyboard text
5469# 205957                JSR     SSM             ;set screen memory

                  ;         Check auto-mode.

546C# A582                  LDA     STAUT           ;auto-mode flag
546E# F013 ^5483            BEQ     STK3            ;if not auto-mode

                  ;         Simulate keypress.

5470# A694        STK2      LDX     STSKP           ;offset to next simulated k:
5472# BD4555                LDA     TSKP,X          ;simulated keypress
5475# E694                  INC     STSKP           ;advance offset to simulate:
5477# A694                  LDX     STSKP           ;offset to simulated keypre:
5479# E013                  CPX     #TSKPL+1        ;last offset+1+1
547B# D014 ^5491            BNE     STK4            ;if last keypress not proce:

                  ;         Self-test memory.

547D# 208553                JSR     DLW             ;delay a long while
5480# 4C9152                JMP     STM             ;self-test memory

                  ;         Get a keypress.

5483# ADFC02      STK3      LDA     CH              ;key code
5486# C9FF                  CMP     #$FF            ;clear code indicator
5488# F0F9 ^5483            BEQ     STK3            ;if no key pressed

548A# C9C0                  CMP     #$C0
548C# B0F5 ^5483            BCS     STK3

548E# ADFC02                LDA     CH              ;key code

                  ;         Process keypress.

5491# A2FF        STK4      LDX     #$FF            ;clear code indicator
5493# 8EFC02                STX     CH              ;key code
5496# 48                    PHA                     ;save key code
5497# 2980                  AND     #$80
5499# F005 ^54A0            BEQ     STK5            ;if not CTRL

549B# A208                  LDX     #8              ;offset to control key text
549D# 205957                JSR     SSM             ;set screen memory

                  ;         Check for shift key.

54A0# 68          STK5      PLA                     ;saved key code
```

```
54A1# 48                      PHA                      ;save key code
54A2# 2940                    AND     #$40
54A4# F00A ^54B0              BEQ     STK6             ;if not shift key

                  ;           Process keyboard shift key display.

54A6# A205                    LDX     #5               ;offset to "SH"
54A8# 205957                  JSR     SSM              ;set screen memory
54AB# A204                    LDX     #4               ;offset to "SH"
54AD# 205957                  JSR     SSM              ;set screen memory

                  ;           Check for special keys.

54B0# 68          STK6        PLA                      ;saved key code
54B1# 293F                    AND     #$3F
54B3# C921                    CMP     #$21
54B5# F068 ^551F              BEQ     KSB              ;if space bar, process disp:

54B7# C92C                    CMP     #$2C
54B9# F074 ^552F              BEQ     KTK              ;if tab key, process displa:

54BB# C934                    CMP     #$34
54BD# F068 ^5527              BEQ     KBK              ;if backspace key, process :

54BF# C90C                    CMP     #$0C
54C1# F076 ^5539              BEQ     KRK              ;if return key, process dis:

                  ;           Process other key displays.

54C3# AA                      TAX                      ;key code
54C4# BD9C57                  LDA     TSMC,X           ;display character
54C7# 48                      PHA                      ;save display character

54C8# A921                    LDA     #low ST3021
54CA# 8595                    STA     STTMP1           ;screen pointer
54CC# A930                    LDA     #high ST3021
54CE# 8596                    STA     STTMP1+1

                  ;           Find display character in screen memory.

54D0# 68                      PLA                      ;saved display character
54D1# A0FF                    LDY     #$FF             ;preset offset

54D3# C8          STK7        INY
54D4# D195                    CMP     (STTMP1),Y
54D6# D0FB ^54D3              BNE     STK7             ;if not found

                  ;           Display inverse video.

54D8# B195                    LDA     (STTMP1),Y
54DA# 4980                    EOR     #$80             ;invert video
54DC# 9195                    STA     (STTMP1),Y

                  ;           Check auto-mode.

54DE# A582        STK8        LDA     STAUT            ;auto-mode flag
54E0# F013 ^54F5              BEQ     STK9             ;if not auto-mode
```

```
                         ;       Process auto-mode.

54E2# 200555                    JSR     SBT                 ;sound beep tone
54E5# A214                      LDX     #20                 ;20-VBLANK delay
54E7# 208753                    JSR     DAW                 ;delay a while
54EA# 201055                    JSR     SAS                 ;silence all sounds
54ED# A20A                      LDX     #10                 ;10-VBLANK delay
54EF# 208753                    JSR     DAW                 ;delay a while
54F2# 4C6254                    JMP     STK1                ;get next simulated keypres:

                         ;       Process manual mode.

54F5# 200555             STK9   JSR     SBT                 ;sound beep tone

54F8# AD0FD2             STK10  LDA     SKSTAT
54FB# 2904                      AND     #$04
54FD# F0F9 ^54F8                BEQ     STK10

54FF# 201055                    JSR     SAS                 ;silence all sounds
5502# 4C6254                    JMP     STK1                ;get next keypress


                         **      SBT - Sound Beep Tone
                         *
                         *       ENTRY   JSR     SBT
                         *
                         *       MODS
                         *               M. W. Colburn   10/26/82
                         *               1. Bring closer to Coding Standard (object :
                         *                  R. K. Nordin 11/01/83


     = 5505              SBT     =       *       ;entry
5505# A964                       LDA     #$64    ;frequency
5507# 8D00D2                     STA     AUDF1   ;set frequency
550A# A9A8                       LDA     #$A8    ;pure tone, half volume
550C# 8D01D2                     STA     AUDC1   ;set control
550F# 60                         RTS             ;return


                         **      SAS - Silence All Sounds
                         *
                         *       ENTRY   JSR     SAS
                         *
                         *       MODS
                         *               M. W. Colburn   10/26/82
                         *               1. Bring closer to Coding Standard (object :
                         *                  R. K. Nordin 11/01/83


     = 5510              SAS     =       *       ;entry
5510# A900                       LDA     #0      ;volume 0
```

```
5512# 8D01D2                  STA     AUDC1    ;silence voice 1
5515# 8D03D2                  STA     AUDC2    ;silence voice 2
5518# 8D05D2                  STA     AUDC3    ;silence voice 3
551B# 8D07D2                  STA     AUDC4    ;silence voice 4
551E# 60                      RTS              ;return



                      **      KSB - Process Keyboard Space Bar Display
                      *
                      *       ENTRY   JSR     KSB
                      *
                      *       MODS
                      *               M. W. Colburn   10/26/82
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = 551F          KSB     =       *        ;entry
551F# A203                    LDX     #3       ;offset to "S P A C E   B A R" text
5521# 205957                  JSR     SSM      ;set screen memory
5524# 4CDE54                  JMP     STK8     ;continue



                      **      KBK - Process Keyboard Backspace Key Display
                      *
                      *       ENTRY   JSR     KBK
                      *
                      *       MODS
                      *               M. W. Colburn   10/26/82
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = 5527          KBK     =       *        ;entry
5527# A206                    LDX     #6       ;offset to "B S" text
5529# 205957                  JSR     SSM      ;set screen memory
552C# 4CDE54                  JMP     STK8     ;continue



                      **      KTK - Process Keyboard Tab Key Display
                      *
                      *       ENTRY   JSR     KTK
                      *
                      *       MODS
                      *               M. W. Colburn   10/26/82
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = 552F          KTK     =       *        ;entry
552F# A97F                    LDA     #$7F
```

```
5531# 8D5230                STA     ST3052
5534# 8D5330                STA     ST3052+1
5537# D0A5 ^54DE            BNE     STK8     ;continue



                    **      KRK - Process Keyboard Return Key Display
                    *
                    *       ENTRY   JSR     KRK
                    *
                    *       MODS
                    *               M. W. Colburn   10/26/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

    = 5539          KRK     =       *        ;entry
5539# A932                  LDA     #$32
553B# 8D6D30                STA     ST306D
553E# A934                  LDA     #$34
5540# 8D6E30                STA     ST306D+1
5543# D099 ^54DE            BNE     STK8     ;continue



                    **      TSKP - Table of Simulated Keypresses


5545# 52080A2B28    TSKP    DB      $52,$08,$0A,$2B,$28,$0D,$3D,$39,$2D     ;"C:
554E# 1F30351A              DB      $1F,$30,$35,$1A                         ;"1:
5552# 7F2D3F280D            DB      $7F,$2D,$3F,$28,$0D                     ;"A:

    = 0012          TSKPL   =       *-TSKP   ;length



                    **      STV - Self-test Audio-visual
                    *
                    *       STV verifies the operation of the display and voice:
                    *       displaying and playing a tune.
                    *
                    *       ENTRY   JSR     STV
                    *
                    *       MODS    M. W. Colburn   10/26/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

    = 5557          STV     =       *               ;entry

                    ;       Initialize.

5557# A202                  LDX     #2              ;audio-visual test colors
```

```
5559# 207357                    JSR     SUC             ;set up colors

                        ;       Test audio-visual.

555C# A900          STV1        LDA     #0
555E# 8597                      STA     STVOC           ;initialize voice indicator

                        ;       Test voice.

5560# A900          STV2        LDA     #0
5562# 8598                      STA     STNOT           ;initialize note counter
5564# A231                      LDX     #low DISL5      ;audio-visual display list
5566# A052                      LDY     #high DISL5
5568# A900                      LDA     #0              ;indicate not keyboard self:
556A# 209E50                    JSR     SDL             ;set up display list

                        ;       Display voice number.

556D# A209                      LDX     #9              ;offset to "VOICE #" text
556F# 205957                    JSR     SSM             ;set screen memory
5572# A597                      LDA     STVOC           ;voice indicator
5574# 4A                        LSR     A               ;voice number
5575# 18                        CLC
5576# 6911                      ADC     #$11            ;adjust for screen memory
5578# 8D0B30                    STA     ST300B          ;voice number display

                        ;       Display staff.

557B# A20F                      LDX     #$0F            ;offset to last byte of sta:

557D# A9FF          STV3        LDA     #$FF            ;color 2
557F# 9D5031                    STA     ST3150,X        ;byte of first line of staf:
5582# 9DB031                    STA     ST31B0,X        ;byte of second line of sta:
5585# 9D1032                    STA     ST3210,X        ;byte of third line of staf:
5588# 9D7032                    STA     ST3270,X        ;byte of fourth line of sta:
558B# 9DD032                    STA     ST32D0,X        ;byte of fifth line of staf:
558E# CA                        DEX
558F# 10EC ^557D                BPL     STV3            ;if not done

                        ;       Display cleft.

5591# A900                      LDA     #0              ;offset to first cleft disp:
5593# 8599                      STA     STCDI           ;cleft display pointer
5595# A90C                      LDA     #2*6
5597# 859A                      STA     STCDA           ;cleft data pointer

5599# A699          STV4        LDX     STCDI           ;cleft display pointer
559B# BD1757                    LDA     TCDA+1,X        ;high address of cleft disp:
559E# A8                        TAY
559F# BD1657                    LDA     TCDA,X          ;low address of cleft displ:
55A2# AA                        TAX
55A3# A59A                      LDA     STCDA           ;cleft data pointer
55A5# 208556                    JSR     DVN
55A8# 18                        CLC
55A9# A59A                      LDA     STCDA           ;cleft data pointer
55AB# 6906                      ADC     #6
55AD# 859A                      STA     STCDA           ;update cleft data pointer
```

```
55AF# E699                  INC     STCDI           ;increment cleft display po:
55B1# E699                  INC     STCDI
55B3# A599                  LDA     STCDI           ;cleft display pointer
55B5# C914                  CMP     #TCDAL          ;length of cleft display ta:
55B7# D0E0 ^5599            BNE     STV4            ;if not done

                    ;       Delay.

55B9# 20B153                JSR     DMW             ;delay a middling while

                    ;       Display and play first note.

55BC# A254                  LDX     #low ST3154
55BE# A031                  LDY     #high ST3154
55C0# A900                  LDA     #0*6
55C2# 208556                JSR     DVN

55C5# A951                  LDA     #$51            ;first note frequency
55C7# 206C56                JSR     SVN

                    ;       Display and play second note.

55CA# A286                  LDX     #low ST3186
55CC# A031                  LDY     #high ST3186
55CE# A900                  LDA     #0*6
55D0# 208556                JSR     DVN

55D3# A95B                  LDA     #$5B            ;second note frequency
55D5# 206C56                JSR     SVN

                    ;       Display and play third note.

55D8# A2F8                  LDX     #low ST30F8
55DA# A030                  LDY     #high ST30F8
55DC# A948                  LDA     #12*6
55DE# 208556                JSR     DVN
55E1# A2C7                  LDX     #low ST30C7
55E3# A030                  LDY     #high ST30C7
55E5# A954                  LDA     #14*6
55E7# 208556                JSR     DVN
55EA# A248                  LDX     #low ST3248
55EC# A032                  LDY     #high ST3248
55EE# A94E                  LDA     #13*6
55F0# 208556                JSR     DVN

55F3# A944                  LDA     #$44            ;third note frequency
55F5# 206C56                JSR     SVN

                    ;       Display and play fourth note.

55F8# A2CA                  LDX     #low ST30CA
55FA# A030                  LDY     #high ST30CA
55FC# A948                  LDA     #12*6
55FE# 208556                JSR     DVN
5601# A21A                  LDX     #low ST321A
5603# A032                  LDY     #high ST321A
5605# A94E                  LDA     #13*6
```

```
5607# 208556                JSR     DVN
560A# A2CA                  LDX     #low ST31CA
560C# A031                  LDY     #high ST31CA
560E# A906                  LDA     #1*6
5610# 208556                JSR     DVN

5613# A93C                  LDA     #$3C            ;fourth note frequency
5615# 206C56                JSR     SVN

                    ;       Display and play fifth note.

5618# A23C                  LDX     #low ST303C
561A# A030                  LDY     #high ST303C
561C# A948                  LDA     #12*6
561E# 208556                JSR     DVN
5621# A28C                  LDX     #low ST318C
5623# A031                  LDY     #high ST318C
5625# A94E                  LDA     #13*6
5627# 208556                JSR     DVN
562A# A23C                  LDX     #low ST313C
562C# A031                  LDY     #high ST313C
562E# A906                  LDA     #1*6
5630# 208556                JSR     DVN

5633# A92D                  LDA     #$2D            ;fifth note frequency
5635# 206C56                JSR     SVN

                    ;       Display and play sixth note.

5638# A29E                  LDX     #low ST309E
563A# A030                  LDY     #high ST309E
563C# A948                  LDA     #12*6
563E# 208556                JSR     DVN
5641# A2EE                  LDX     #low ST31EE
5643# A031                  LDY     #high ST31EE
5645# A94E                  LDA     #13*6
5647# 208556                JSR     DVN

564A# A935                  LDA     #$35            ;sixth note frequency
564C# 206C56                JSR     SVN

                    ;       Delay.

564F# 208553                JSR     DLW             ;delay a long while

                    ;       Advance to next voice.

5652# E697                  INC     STVOC           ;increment voice indicator
5654# E697                  INC     STVOC
5656# A597                  LDA     STVOC           ;voice indicator
5658# C908                  CMP     #8              ;last voice indicator
565A# D007 ^5663            BNE     STV5            ;if all voices not processe:

                    ;       Process test completion.

565C# A582                  LDA     STAUT           ;auto-mode flag
565E# D006 ^5666            BNE     STV6            ;if auto-mode, perform keyb:
```

```
5660# 4C5C55                JMP     STV1            ;repeat audio-visual test

                  ;         Test next voice.

5663# 4C6055      STV5      JMP     STV2            ;test next voice

                  ;         Self-test keyboard.

5666# 208553      STV6      JSR     DLW             ;delay a long while
5669# 4C5054                JMP     STK             ;self-test keyboard


                  **        SVN - Sound Tone
                  *
                  *         ENTRY   JSR     SVN
                  *
                  *         MODS
                  *                 M. W. Colburn   10/26/82
                  *                 1. Bring closer to Coding Standard (object :
                  *                    R. K. Nordin 11/01/83


      = 566C      SVN       =       *               ;entry

                  ;         Sound note.

566C# A497                  LDY     STVOC           ;current voice indicator
566E# 9900D2                STA     AUDF1,Y         ;set frequency
5671# A9A8                  LDA     #$A8            ;pure tone, half volume
5673# 9901D2                STA     AUDC1,Y         ;set control

                  ;         Delay a while.

5676# A698                  LDX     STNOT           ;current note
5678# BDB656                LDA     TNDD,X          ;delay time
567B# AA                    TAX
567C# 208753                JSR     DAW             ;delay a while

                  ;         Increment note counter.

567F# E698                  INC     STNOT           ;increment note counter

                  ;         Exit.

5681# 201055                JSR     SAS             ;silence all sounds
5684# 60                    RTS                     ;return
```

```
                      **      DVN - Display
                      *
                      *       ENTRY   JSR     DVN
                      *
                      *       MODS
                      *               M. W. Colburn   10/26/82
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = 5685          DVN     =       *                 ;entry
5685# 8698                    STX     STTMP2
5687# 849C                    STY     STTMP2+1
5689# AA                      TAX
568A# A000                    LDY     #0
568C# A910                    LDA     #16
568E# 859D                    STA     STTMP3
5690# A906                    LDA     #6
5692# 85A3                    STA     STTMP4

5694# BDBC56          DVN1    LDA     TAVD,X
5697# 119B                    ORA     (STTMP2),Y
5699# 919B                    STA     (STTMP2),Y
569B# 20AA56                  JSR     AST               ;add 16
569E# C69D                    DEC     STTMP3
56A0# D0F2 ^5694              BNE     DVN1

56A2# E69D                    INC     STTMP3
56A4# E8                      INX
56A5# C6A3                    DEC     STTMP4
56A7# D0EB ^5694              BNE     DVN1

56A9# 60                      RTS                       ;return



                      **      AST - Add Sixteen
                      *
                      *       ENTRY   JSR     AST
                      *
                      *       MODS
                      *               M. W. Colburn   10/26/82
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = 56AA          AST     =       *                 ;entry
56AA# 18                      CLC
56AB# A59B                    LDA     STTMP2            ;current low value
56AD# 6910                    ADC     #16               ;add 16
56AF# 859B                    STA     STTMP2            ;new low value
56B1# 9002 ^56B5              BCC     AST1              ;if no carry

56B3# E69C                    INC     STTMP2+1          ;adjust high value

56B5# 60              AST1    RTS                       ;return
```

```
                    **      TNDD - Table of Note Duration Delays


56B6# 20            TNDD    DB        32       ;0 - first note
56B7# 20                    DB        32       ;1 - second note
56B8# 20                    DB        32       ;2 - third note
56B9# 10                    DB        16       ;3 - fourth note
56BA# 10                    DB        16       ;4 - fifth note
56BB# 20                    DB        32       ;5 - sixth note



                    **      TAVD - Table of Audio-visual Test Display Data


56BC# 011F3F7F3E    TAVD    DB        $01,$1F,$3F,$7F,$3E,$1C          ;0
56C2# 0041424C70            DB        $00,$41,$42,$4C,$70,$40          ;1
56C8# 0001020408            DB        $00,$01,$02,$04,$08,$10          ;2
56CE# 0043444848            DB        $00,$43,$44,$48,$48,$48          ;3
56D4# 0044221008            DB        $00,$44,$22,$10,$08,$07          ;4
56DA# 0004080502            DB        $00,$04,$08,$05,$02,$00          ;5
56E0# 0030488884            DB        $00,$30,$48,$88,$84,$84          ;6
56E6# 00888890A0            DB        $00,$88,$88,$90,$A0,$C0          ;7
56EC# 00F0888482            DB        $00,$F0,$88,$84,$82,$82          ;8
56F2# 0082828488            DB        $00,$82,$82,$84,$88,$F0          ;9
56F8# 0000000000            DB        $00,$00,$00,$00,$00,$80          ;10
56FE# 8080808080            DB        $80,$80,$80,$80,$80,$80          ;11
5704# 001C3E7F7E            DB        $00,$1C,$3E,$7F,$7E,$7C          ;12
570A# 4000000000            DB        $40,$00,$00,$00,$00,$00          ;13
5710# 0004040605            DB        $00,$04,$04,$06,$05,$06          ;14



                    **      TCDA - Table of Cleft Display Addresses


5716# C130          TCDA    DW        ST30C1  ;0
5718# 2131                  DW        ST3121  ;1
571A# 8131                  DW        ST3181  ;2
571C# F131                  DW        ST31F1  ;3
571E# 0230                  DW        ST3002  ;4
5720# 6230                  DW        ST3062  ;5
5722# 2231                  DW        ST3122  ;6
5724# 8231                  DW        ST3182  ;7
5726# C230                  DW        ST30C2  ;8
5728# C231                  DW        ST31C2  ;9

        = 0014      TCDAL   =         *-TCDA           ;length
```

```
                       **      SVR - Set Value in Range
                       *
                       *       ENTRY   JSR     SVR
                       *               A = value to set
                       *               X = offset to TARS range
                       *
                       *       EXIT
                       *               A = value set
                       *
                       *       MODS
                       *               M. W. Colburn   10/26/82
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83

        = 572A         SVR     =       *               ;entry

                       ;       Initialize.

572A# 48                       PHA                     ;save value

                       ;       Set address range.

572B# BDDC57                   LDA     TARS,X          ;start of range
572E# 859E                     STA     STADR1
5730# BDDD57                   LDA     TARS+1,X
5733# 859F                     STA     STADR1+1
5735# BDDE57                   LDA     TARS+2,X        ;end of range
5738# 85A0                     STA     STADR2
573A# BDDF57                   LDA     TARS+3,X
573D# 85A1                     STA     STADR2+1

                       ;       Set value in range.

573F# A000                     LDY     #0              ;offset to first byte

5741# 68               SVR1    PLA                     ;saved value
5742# 919E                     STA     (STADR1),Y      ;byte of range
5744# E69E                     INC     STADR1          ;increment low address
5746# D002 ^574A               BNE     SVR2            ;if no carry

5748# E69F                     INC     STADR1+1        ;adjust high address

574A# 48               SVR2    PHA                     ;save value
574B# A59E                     LDA     STADR1          ;low current address
574D# C5A0                     CMP     STADR2          ;low end of range
574F# D0F0 ^5741               BNE     SVR1            ;if definitely not done

5751# A59F                     LDA     STADR1+1        ;high current address
5753# C5A1                     CMP     STADR2+1        ;high end of range
5755# D0EA ^5741               BNE     SVR1            ;if not done

                       ;       Exit.

5757# 68                       PLA                     ;restore value
5758# 60                       RTS                     ;return
```

```
                        **      SSM - Set Screen Memory
                        *
                        *       ENTRY   JSR     SSM
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


       = 5759           SSM     =       *               ;entry
5759# BD57CA                    LDA     TSTO,X          ;offset to source
575C# A8                        TAY
575D# BDEC57                    LDA     TSTL,X          ;length of source
5760# 859E                      STA     STADR1          ;length
5762# BDF657                    LDA     TSTD,X          ;offset to destination
5765# AA                        TAX

5766# B961CA           SSM1     LDA     TTXT,Y          ;byte of source
5769# 9D0030                    STA     ST3000,X        ;byte of destination
576C# C8                        INY
576D# E8                        INX
576E# C69E                      DEC     STADR1          ;decrement length
5770# D0F4 ^5766                BNE     SSM1            ;if not done

5772# 60                        RTS                     ;return



                        **      SUC - Set Up Colors
                        *
                        *       ENTRY   JSR     SUC
                        *               X = 0, if main screen colors
                        *                 = 1, if memory test colors
                        *                 = 2, if keyboard test colors
                        *                 = 3, if audio-visual test colors
                        *
                        *       EXIT
                        *               COLOR0, COLOR1, COLOR2 and COLOR4 set.
                        *
                        *       CHANGES
                        *               A
                        *
                        *       CALLS
                        *               -none-
                        *
                        *       MODS
                        *               M. W. Colburn   10/26/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


       = 5773           SUC     =       *               ;entry

5773# BD8C57                    LDA     SUCA,X
5776# 8DC402                    STA     COLOR0  ;playfield 0 color
```

```
5779# BD9057             LDA     SUCB,X
577C# 8DC502             STA     COLOR1  ;playfield 1 color

577F# BD9457             LDA     SUCC,X
5782# 8DC602             STA     COLOR2  ;playfield 2 color

5785# BD9857             LDA     SUCD,X
5788# 8DC802             STA     COLOR4  ;background color

578B# 60                 RTS             ;return


578C# 2C          SUCA   DB      $2C     ;0 - main screen playfield 0 color
578D# 0C                 DB      $0C     ;1 - memory test playfield 0 color
578E# 2A                 DB      $2A     ;2 - keyboard test playfield 0 colo:
578F# 18                 DB      $18     ;3 - audio-visual test playfield 0 :

5790# 0F          SUCB   DB      $0F     ;0 - main screen playfield 1 color
5791# 32                 DB      $32     ;1 - memory test playfield 1 color
5792# 0C                 DB      $0C     ;2 - keyboard test playfield 1 colo:
5793# 0E                 DB      $0E     ;3 - audio-visual test playfield 1 :

5794# D2          SUCC   DB      $D2     ;0 - main screen playfield 2 color
5795# D6                 DB      $D6     ;1 - memory test playfield 2 color
5796# 00                 DB      $00     ;2 - keyboard test playfield 2 colo:
5797# 84                 DB      $84     ;3 - audio-visual test playfield 2 :

5798# D2          SUCD   DB      $D2     ;0 - main screen background color
5799# A0                 DB      $A0     ;1 - memory test background color
579A# 30                 DB      $30     ;2 - keyboard test background color
579B# 84                 DB      $84     ;3 - audio-visual test background c:


                  **     TSMC - Table of Screen Memory Character Codes
                  *
                  *      Entry n is the screen memory character code for key:


579C# 2C          TSMC   DB      $2C     ;$00 - L key
579D# 2A                 DB      $2A     ;$01 - J key
579E# 1B                 DB      $1B     ;$02 - semicolon key
579F# 91                 DB      $91     ;$03
57A0# 92                 DB      $92     ;$04
57A1# 2B                 DB      $2B     ;$05 - K key
57A2# 0B                 DB      $0B     ;$06 - plus key
57A3# 0A                 DB      $0A     ;$07 - asterisk key
57A4# 2F                 DB      $2F     ;$08 - O key
57A5# 00                 DB      $00     ;$09
57A6# 30                 DB      $30     ;$0A - P key
57A7# 35                 DB      $35     ;$0B - U key
57A8# B2                 DB      $B2     ;$0C - RETURN key
57A9# 29                 DB      $29     ;$0D - I key
57AA# 0D                 DB      $0D     ;$0E - minus key
57AB# 1D                 DB      $1D     ;$0F - = key
```

```
57AC# 36                      DB        $36       ;$10 - V key
57AD# A8                      DB        $A8       ;$11
57AE# 23                      DB        $23       ;$12 - C key
57AF# 93                      DB        $93       ;$13
57B0# 94                      DB        $94       ;$14
57B1# 22                      DB        $22       ;$15 - B key
57B2# 38                      DB        $38       ;$16 - X key
57B3# 3A                      DB        $3A       ;$17 - Z key
57B4# 14                      DB        $14       ;$18 - 4 key
57B5# 00                      DB        $00       ;$19
57B6# 13                      DB        $13       ;$1A - 3 key
57B7# 16                      DB        $16       ;$1B - 6 key
57B8# 5B                      DB        $5B       ;$1C - ESC key
57B9# 15                      DB        $15       ;$1D - 5 key
57BA# 12                      DB        $12       ;$1E - 2 key
57BB# 11                      DB        $11       ;$1F - 1 key

57BC# 0C                      DB        $0C       ;$20 - comma key
57BD# 00                      DB        $00       ;$21 - space key
57BE# 0E                      DB        $0E       ;$22 - period key
57BF# 2E                      DB        $2E       ;$23 - N key
57C0# 00                      DB        $00       ;$24
57C1# 2D                      DB        $2D       ;$25 - M key
57C2# 0F                      DB        $0F       ;$26 - / key
57C3# A1                      DB        $A1       ;$27 - inverse video key
57C4# 32                      DB        $32       ;$28 - R key
57C5# 00                      DB        $00       ;$29
57C6# 25                      DB        $25       ;$2A - E key
57C7# 39                      DB        $39       ;$2B - Y key
57C8# FF                      DB        $FF       ;$2C - TAB key
57C9# 34                      DB        $34       ;$2D - T key
57CA# 37                      DB        $37       ;$2E - W key
57CB# 31                      DB        $31       ;$2F - Q key

57CC# 19                      DB        $19       ;$30 - 9 key
57CD# 00                      DB        $00       ;$31
57CE# 10                      DB        $10       ;$32 - 0 key
57CF# 17                      DB        $17       ;$33 - 7 key
57D0# A2                      DB        $A2       ;$34 - backspace key
57D1# 18                      DB        $18       ;$35 - 8 key
57D2# 1C                      DB        $1C       ;$36 - < key
57D3# 1E                      DB        $1E       ;$37 - > key
57D4# 26                      DB        $26       ;$38 - F key
57D5# 28                      DB        $28       ;$39 - H key
57D6# 24                      DB        $24       ;$3A - D key
57D7# 00                      DB        $00       ;$3B
57D8# A3                      DB        $A3       ;$3C - CAPS key
57D9# 27                      DB        $27       ;$3D - G key
57DA# 33                      DB        $33       ;$3E - S key
57DB# 21                      DB        $21       ;$3F - A key
```

```
                        **      TARS - Table of Address Ranges to Set

   57DC# 0030FF3E       TARS    DW      ST3000,ST3000+$0EFF     ;0 - screen memory
   57E0# 20302430               DW      ST3020,ST3020+4         ;1 - memory test fi:
   57E4# 24302830               DW      ST3024,ST3024+4         ;2 - memory test se:
   57E8# 00302030               DW      ST3000,ST3000+32        ;3 - main screen bo:


                        **      TSTL - Table of Self-test Text Lengths

   57EC# 13             TSTL    DB      TXT0L   ;0 - length of "MEMORY TEST   ROM" :
   57ED# 03                     DB      TXT1L   ;1 - length of "RAM" text
   57EE# 13                     DB      TXT2L   ;2 - length of "KEYBOARD TEST" text
   57EF# 13                     DB      TXT3L   ;3 - length of "S P A C E   B A R" :
   57F0# 04                     DB      TXT4L   ;4 - length of "SH" text
   57F1# 04                     DB      TXT5L   ;5 - length of "SH" text
   57F2# 03                     DB      TXT6L   ;6 - length of "B S" text
   57F3# A8                     DB      TXT7L   ;7 - length of keyboard text
   57F4# 03                     DB      TXT8L   ;8 - length of control key text
   57F5# 07                     DB      TXT9L   ;9 - length of "VOICE #" text


                        **      TSTD - Table of Self-test Text Destination Offsets

   57F6# 00             TSTD    DB      ST3000-ST3000   ;0 - offset to "MEMORY TEST:
   57F7# 28                     DB      ST3028-ST3000   ;1 - offset to "RAM" text
   57F8# 00                     DB      ST3000-ST3000   ;2 - offset to "KEYBOARD TE:
   57F9# 87                     DB      ST3087-ST3000   ;3 - offset to "S P A C E  :
   57FA# 92                     DB      ST3092-ST3000   ;4 - offset to "SH" text
   57FB# AB                     DB      ST30AB-ST3000   ;5 - offset to "SH" text
   57FC# 4C                     DB      ST304C-ST3000   ;6 - offset to "B S" text
   57FD# 22                     DB      ST3022-ST3000   ;7 - offset to keyboard tex:
   57FE# 72                     DB      ST3072-ST3000   ;8 - offset to control key :
   57FF# 04                     DB      ST3004-ST3000   ;9 - offset to "VOICE #" te:
```

```
 5800#           ***     (C) Copyright 1978 Shepardson Microsystems, Inc.


 5800#                   FIX     $D800


                 ***     FPP - Floating Point Package
                 *
                 *       FPP is a collection of routines for floating point
                 *       computations.  A floating point number is represent:
                 *       in 6 bytes:
                 *
                 *       Byte 0
                 *               Bit 7           Sign of mantissa
                 *               Bits 0 - 6      BCD exponent, biased by $40
                 *
                 *       Bytes 1 - 5             BCD mantissa
                 *
                 *       MODS
                 *               Shepardson Microsystems
                 *
                 *               Produce 2K version.
                 *               M. Lorenzen     09/06/81


 D800                    FIX     AFP


                 **      AFP - Convert ASCII to Floating Point
                 *
                 *       ENTRY   JSR     AFP
                 *               INBUFF = line buffer pointer
                 *               CIX = offset to first byte of number
                 *
                 *       EXIT
                 *               C clear, if valid number
                 *               C set, if invalid number
                 *
                 *       NOTES
                 *               Problem: bytes wasted by check for "-", nea:
                 *
                 *       MODS
                 *               Original Author Unknown
                 *               1. Bring closer to Coding Standard (object :
                 *                  R. K. Nordin 11/01/83


                 ;AFP    =       *               ;entry
```

```
                        ;       Initialize.

 D800   20A1DB                  JSR     SLB     ;skip leading blanks

                        ;       Check for number.

 D803   20B8DB                  JSR     TVN     ;test for valid number character
 D806   B039 ^D841              BCS     AFP5    ;if not number character

                        ;       Set initial values.

 D808   A2ED                    LDX     #EEXP   ;exponent
 D80A   A004                    LDY     #4      ;indicate 4 bytes to clear
 D80C   2048DA                  JSR     ZXLY
 D80F   A2FF                    LDX     #$FF
 D811   86F1                    STX     DIGRT   ;number of digits after decimal poi;
 D813   2044DA                  JSR     ZFR0    ;zero FR0
 D816   F004 ^D81C              BEQ     AFP2    ;get first character

                        ;       Indicate not first character.

 D818   A9FF            AFP1    LDA     #$FF    ;indicate not first character
 D81A   85F0                    STA     FCHFLG  ;first character flag

                        ;       Get next character.

 D81C   2094DB          AFP2    JSR     GNC     ;get next character
 D81F   B021 ^D842              BCS     AFP6    ;if character not numeric

                        ;       Process numeric character.

 D821   48                      PHA                             ;save digit
 D822   A6D5                    LDX     FR0M                    ;first byte
 D824   D011 ^D837              BNE     AFP3                    ;if not zero
 D826   20E8DB                  JSR     S0L                     ;shift FR0 left 1 digit
 D829   68                      PLA                             ;saved digit
 D82A   05D9                    ORA     FR0M+FMPREC-1           ;insert into last byte
 D82C   85D9                    STA     FR0M+FMPREC-1           ;update last byte

                        ;       Check for decimal point.

 D82E   A6F1                    LDX     DIGRT   ;number of digits after decimal poi;
 D830   30E6 ^D818              BMI     AFP1    ;if no decimal point, process next ;

                        ;       Increment number of digits after decimal point.

 D832   E8                      INX             ;increment number of digits
 D833   86F1                    STX     DIGRT   ;number of digits after decimal poi;
 D835   D0E1 ^D818              BNE     AFP1    ;process next character

                        ;       Increment exponent, if necessary.

 D837   68              AFP3    PLA             ;clean stack
 D838   A6F1                    LDX     DIGRT   ;number of digits after decimal poi;
 D83A   1002 ^D83E              BPL     AFP4    ;if already have decimal point
```

```
D83C  E6ED                  INC    EEXP    ;increment number of digits more th:

                      ;     Process next character.

D83E  4C18D8        AFP4    JMP    AFP1    ;process next character

                      ;     Exit.

D841  60            AFP5    RTS            ;return

                      ;     Process non-numeric character.

D842  C92E          AFP6    CMP    #'.'
D844  F014 ^D85A            BEQ    AFP8    ;if ".", process decimal point

D846  C945                  CMP    #'E'
D848  F019 ^D863            BEQ    AFP9    ;if "E", process exponent

D84A  A6F0                  LDX    FCHFLG  ;first character flag
D84C  D068 ^D8B6            BNE    AFP16   ;if not first character, process en:

D84E  C92B                  CMP    #'+'
D850  F0C6 ^D818            BEQ    AFP1    ;if "+", process next character

D852  C92D                  CMP    #'-'
D854  F000 ^D856            BEQ    AFP7    ;if "-", process negative sign

                      ;     Process negative sign.

D856  85EE          AFP7    STA    NSIGN   ;sign of number
D858  F0BE ^D818            BEQ    AFP1    ;process next character

                      ;     Process decimal point.

D85A  A6F1          AFP8    LDX    DIGRT   ;number of digits after decimal poi:
D85C  1058 ^D8B6            BPL    AFP16   ;if already have decimal point

D85E  E8                    INX            ;zero
D85F  86F1                  STX    DIGRT   ;number of digits after decimal poi:
D861  F0B5 ^D818            BEQ    AFP1    ;process next character

                      ;     Process exponent.

D863  A5F2          AFP9    LDA    CIX     ;offset to character
D865  85EC                  STA    FRX     ;save offset to character
D867  2094D8                JSR    GNC     ;get next character
D86A  B037 ^D8A3            BCS    AFP13   ;if not numeric

                      ;     Process numeric character in exponent.

D86C  AA            AFP10   TAX            ;first character of exponent
D86D  A5ED                  LDA    EEXP    ;number of digits more than 9
D86F  48                    PHA            ;save number of digits more than 9
D870  86ED                  STX    EEXP    ;first character of exponent

                      ;     Process second character of exponent.
```

```
D872  2094D8              JSR     GNC     ;get next character
D875  B017 ^D88E          BCS     AFP11   ;if not numeric, no second digit

D877  48                  PHA             ;save second digit
D878  A5ED                LDA     EEXP    ;first digit
D87A  0A                  ASL     A       ;2 times first digit
D87B  85ED                STA     EEXP    ;2 times first digit
D87D  0A                  ASL     A       ;4 times first digit
D87E  0A                  ASL     A       ;8 times first digit
D87F  65ED                ADC     EEXP    ;add 2 times first digit
D881  85ED                STA     EEXP    ;save 10 times first digit
D883  68                  PLA             ;saved second digit
D884  18                  CLC
D885  65ED                ADC     EEXP    ;insert in exponent
D887  85ED                STA     EEXP    ;update exponent

                  ;       Process third character of exponent.

D889  A4F2                LDY     CIX     ;offset to third character
D88B  209DD8              JSR     ICX     ;increment offset

D88E  A5EF        AFP11   LDA     ESIGN   ;sign of exponent
D890  F009 ^D89B          BEQ     AFP12   ;if no sign on exponent

                  ;       Process negative exponent.

D892  A5ED                LDA     EEXP    ;exponent
D894  49FF                EOR     #$FF    ;complement exponent
D896  18                  CLC
D897  6901                ADC     #1      ;add 1 for 2's complement
D899  85ED                STA     EEXP    ;update exponent

                  ;       Add in number of digits more than 9.

D89B  68          AFP12   PLA             ;saved number of digits more than 9
D89C  18                  CLC
D89D  65ED                ADC     EEXP    ;add exponent
D89F  85ED                STA     EEXP    ;update exponent
D8A1  D013 ^D8B6          BNE     AFP16   ;process end of input

                  ;       Process non-numeric in exponent.

D8A3  C92B        AFP13   CMP     #'+'
D8A5  F006 ^D8AD          BEQ     AFP14   ;if "+", process next character

D8A7  C92D                CMP     #'-'
D8A9  D007 ^D8B2          BNE     AFP15

D8AB  85EF                STA     ESIGN   ;save sign of exponent

                  ;       Process next character.

D8AD  2094D8      AFP14   JSR     GNC     ;get next character
D8B0  90BA ^D86C          BCC     AFP10   ;if numeric, process numeric charac:

                  ;       Process other non-numeric in exponent.
```

```
D8B2  A5EC          AFP15   LDA     FRX       ;saved offset
D8B4  85F2                  STA     CIX       ;restore offset

                    ;       Process end of input.

D8B6  C6F2          AFP16   DEC     CIX       ;decrement offset
D8B8  A5ED                  LDA     EEXP      ;exponent
D8BA  A6F1                  LDX     DIGRT     ;number of digits after decimal poi:
D8BC  3005 ^D8C3            BMI     AFP17     ;if no decimal point

D8BE  F003 ^D8C3            BEQ     AFP17     ;if no digits after decimal point

D8C0  38                    SEC
D8C1  E5F1                  SBC     DIGRT     ;subtract number of digits after de:

D8C3  48            AFP17   PHA               ;save adjusted exponent
D8C4  2A                    ROL     A         ;set C with sign of exponent
D8C5  68                    PLA               ;saved adjusted exponent
D8C6  6A                    ROR     A         ;shift right
D8C7  85ED                  STA     EEXP      ;save power of 100
D8C9  9003 ^D8CE            BCC     AFP18     ;if no carry, process even number

D8CB  20E8D8                JSR     SOL       ;shift FR0 left 1 digit

D8CE  A5ED          AFP18   LDA     EEXP      ;exponent
D8D0  18                    CLC
D8D1  6944                  ADC     #$40+4    ;add bias plus 4 for normalization
D8D3  85D4                  STA     FR0       ;save exponent

D8D5  2000DC                JSR     NORM      ;normalize number
D8D8  B008 ^D8E5            BCS     AFP20     ;if error

                    ;       Check sign of number.

D8DA  A6EE                  LDX     NSIGN     ;sign of number
D8DC  F006 ^D8E4            BEQ     AFP19     ;if sign of number not negative

                    ;       Process negative number.

D8DE  A5D4                  LDA     FR0       ;first byte of mantissa
D8E0  0980                  ORA     #$80      ;indicate negative
D8E2  85D4                  STA     FR0       ;update first byte of mantissa

                    ;       Exit.

D8E4  18            AFP19   CLC               ;indicate valid number

D8E5  60            AFP20   RTS               ;return
```

```
D8E6                          FIX     FASC


                        **      FASC - Convert Floating Point Number to ASCII
                        *
                        *       ENTRY   JSR     FASC
                        *               FR0 - FR0+5 = number to convert
                        *
                        *       EXIT
                        *               INBUFF = pointer to start of number
                        *               High order bit of last charecter set
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

                        ;FASC   =       *       ;entry

                        ;       Initialize.

D8E6  2051DA                    JSR     ILP     ;initialize line buffer pointer
D8E9  A930                      LDA     #'0'
D8EB  8D7F05                    STA     LBPR2   ;put "0" in front of line buffer

                        ;       Check for E format required.

D8EE  A5D4                      LDA     FR0     ;exponent
D8F0  F028 ^D91A                BEQ     FASC2   ;if exponent zero, number zero

D8F2  297F                      AND     #$7F    ;clear sign
D8F4  C93F                      CMP     #$40-1  ;bias-1
D8F6  9028 ^D920                BCC     FASC3   ;if exponent < bias-1, E format req:

D8F8  C945                      CMP     #$40+5  ;bias+5
D8FA  B024 ^D920                BCS     FASC3   ;if >= bias+5, E format required

                        ;       Process E format not required.

D8FC  38                        SEC
D8FD  E93F                      SBC     #$40-1  ;subtract bias-1, yielding decimal :
D8FF  2070DC                    JSR     COA     ;convert FR0 to ASCII
D902  20A4DC                    JSR     FNZ     ;find last non-zero character
D905  0980                      ORA     #$80    ;set high order bit
D907  9D8005                    STA     LBUFF,X ;update last character
D90A  AD8005                    LDA     LBUFF   ;first character
D90D  C92E                      CMP     #'.'
D90F  F003 ^D914                BEQ     FASC1   ;if decimal point

D911  4C88D9                    JMP     FASC10

D914  20C1DC            FASC1   JSR     DLP     ;decrement line buffer pointer
D917  4C9CD9                    JMP     FASC11  ;perform final adjustment
```

```
                       ;        Process zero.

D91A  A980         FASC2   LDA     #$80+'0'         ;"0" with high order bit se:
D91C  8D8005               STA     LBUFF            ;put zero character in line:
D91F  60                   RTS                      ;return

                       ;        Process E format required.

D920  A901         FASC3   LDA     #1
D922  2070DC               JSR     COA      ;convert FR0 to ASCII
D925  20A4DC               JSR     FNZ      ;find last non-zero character
D928  E8                   INX              ;increment offset to last character
D929  86F2                 STX     CIX      ;save offset to last character

                       ;        Adjust exponent.

D92B  A5D4                 LDA     FR0      ;exponent
D92D  0A                   ASL     A        ;double exponent
D92E  38                   SEC
D92F  E980                 SBC     #$40*2   ;subtract 2 times bias

                       ;        Check first character for "0"..

D931  AE8005               LDX     LBUFF    ;first character
D934  E030                 CPX     #'0'
D936  F017 ^D94F           BEQ     FASC5    ;if "0"

                       ;        Put decimal after first character.

D938  AE8105               LDX     LBUFF+1 ;second character
D93B  AC8205               LDY     LBUFF+2 ;decimal point
D93E  8E8205               STX     LBUFF+2 ;decimal point
D941  8C8105               STY     LBUFF+1 ;third character
D944  A6F2                 LDX     CIX      ;offset
D946  E002                 CPX     #2       ;former offset to decimal point
D948  D002 ^D94C           BNE     FASC4    ;if offset pointed to second charac:

D94A  E6F2                 INC     CIX      ;increment offset

D94C  18           FASC4   CLC
D94D  6901                 ADC     #1       ;adjust exponent for movement of de:

                       ;        Convert exponent to ASCII.

D94F  85ED         FASC5   STA     EEXP     ;exponent
D951  A945                 LDA     #'E'
D953  A4F2                 LDY     CIX      ;offset
D955  209FDC               JSR     SAL      ;store ASCII character in line buff:
D958  84F2                 STY     CIX      ;save offset
D95A  A5ED                 LDA     EEXP     ;exponent
D95C  100B ^D969           BPL     FASC6    ;if exponent positive

D95E  A900                 LDA     #0
D960  38                   SEC
D961  E5ED                 SBC     EEXP     ;complement exponent
D963  85ED                 STA     EEXP     ;update exponent
D965  A92D                 LDA     #'-'
```

```
D967  D002 ^D96B            BNE     FASC7    ;store "-"

D969  A928         FASC6    LDA     #'+'

D96B  209FDC       FASC7    JSR     SAL      ;store ASCII character in line buff:
D96E  A200                  LDX     #0       ;initial number of 10's
D970  A5ED                  LDA     EEXP     ;exponent

D972  38           FASC8    SEC
D973  E90A                  SBC     #10      ;subtract 10
D975  9003 ^D97A            BCC     FASC9    ;if < 0, done

D977  E8                    INX              ;increment number of 10's
D978  D0F8 ^D972            BNE     FASC8    ;continue

D97A  18           FASC9    CLC
D97B  690A                  ADC     #10      ;add back 10
D97D  48                    PHA              ;save remainder
D97E  8A                    TXA              ;number of 10's
D97F  209DDC                JSR     SNL      ;store number in line buffer
D982  68                    PLA              ;saved remainder
D983  0980                  ORA     #$80     ;set high order bit
D985  209DDC                JSR     SNL      ;store number in line buffer

                   ;        Perform final adjustment.

D988  AD8005       FASC10   LDA     LBUFF    ;first character
D98B  C930                  CMP     #'0'
D98D  D00D ^D99C            BNE     FASC11

                   ;        Increment pointer to point to non-zero character.

D98F  18                    CLC
D990  A5F3                  LDA     INBUFF          ;line buffer pointer
D992  6901                  ADC     #1              ;add 1
D994  85F3                  STA     INBUFF          ;update line buffer pointer
D996  A5F4                  LDA     INBUFF+1
D998  6900                  ADC     #0
D99A  85F4                  STA     INBUFF+1

                   ;        Check for positive exponent.

D99C  A5D4         FASC11   LDA     FR0             ;exponent
D99E  1009 ^D9A9            BPL     FASC12          ;if exponent positive, exit

                   ;        Process negative exponent.

D9A0  20C1DC                JSR     DLP             ;decrement line buffer poin:
D9A3  A000                  LDY     #0              ;offset to first character
D9A5  A92D                  LDA     #'-'
D9A7  91F3                  STA     (INBUFF),Y      ;put "-" in line buffer

                   ;        Exit.

D9A9  60           FASC12   RTS                     ;return
```

```
D9AA                           FIX     IFP



                       **      IFP - Convert Integer to Floating Point Number
                       *
                       *       ENTRY   JSR     IFP
                       *               FR0 - FR0+1 = integer to convert
                       *
                       *       EXIT
                       *               FR0 - FR0+5 = floating point number
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83


                       ;IFP    =       *               ;entry

                       ;       Initialize.

D9AA   A5D4                    LDA     FR0             ;low integer
D9AC   85F8                    STA     ZTEMP4+1        ;save low integer
D9AE   A5D5                    LDA     FR0+1           ;high integer
D9B0   85F7                    STA     ZTEMP4          ;save high integer
D9B2   2044DA                  JSR     ZFR0            ;zero FR0

                       ;       Convert to floating point.

D9B5   F8                      SED
D9B6   A010                    LDY     #16             ;number of bits in integer

D9B8   06F8            IFP1    ASL     ZTEMP4+1        ;shift integer
D9BA   26F7                    ROL     ZTEMP4          ;shift integer, setting C i:

D9BC   A203                    LDX     #3              ;offset to last possible by:

D9BE   B5D4            IFP2    LDA     FR0,X           ;byte of number
D9C0   75D4                    ADC     FR0,X           ;double byte, adding in car:
D9C2   95D4                    STA     FR0,X           ;update byte of number
D9C4   CA                      DEX
D9C5   D0F7 ^D9BE              BNE     IFP2            ;if not done

D9C7   88                      DEY                     ;decrement count of integer:
D9C8   D0EE ^D9B8              BNE     IFP1            ;if not done

D9CA   D8                      CLD

                       ;       Set exponent.

D9CB   A942                    LDA     #$40+2          ;indicate decimal after las:
D9CD   85D4                    STA     FR0             ;exponent

                       ;       Exit.
```

```
D9CF  4C00DC                  JMP     NORM            ;normalize, return



D9D2                          FIX     FPI



                     **       FPI - Convert Floating Point Number to Integer
                     *
                     *        ENTRY   JSR     FPI
                     *                FR0 - FR0+5 = floating point number
                     *
                     *        EXIT
                     *                C set, if error
                     *                C clear, if no error
                     *                FR0 - FR0+1 = integer
                     *
                     *        MODS
                     *                Original Author Unknown
                     *                1. Bring closer to Coding Standard (object :
                     *                   R. K. Nordin 11/01/83

                     ;FPI     =       *               ;entry

                     ;        Initialize.

D9D2  A900                    LDA     #0
D9D4  85F7                    STA     ZTEMP4          ;zero integer
D9D6  85F8                    STA     ZTEMP4+1

                     ;        Check exponent.

D9D8  A5D4                    LDA     FR0             ;exponent
D9DA  3066 ^DA42              BMI     FPI4            ;if sign of exponent is neg:

D9DC  C943                    CMP     #$40+3          ;bias+3
D9DE  B062 ^DA42              BCS     FPI4            ;if number too big, error

D9E0  38                      SEC
D9E1  E940                    SBC     #$40            ;subtract bias
D9E3  903F ^DA24              BCC     FPI2            ;if number less than 1, tes:

                     ;        Compute number of digits to convert.

D9E5  6900                    ADC     #0              ;add carry
D9E7  0A                      ASL     A               ;2 times exponent-$40+1
D9E8  85F5                    STA     ZTEMP1          ;number of digits to conver:

                     ;        Convert.

D9EA  205ADA         FPI1     JSR     SIL             ;shift integer left
D9ED  B053 ^DA42              BCS     FPI4            ;if number too big, error
```

```
D9EF   A5F7                  LDA      ZTEMP4             ;2 times integer
D9F1   85F9                  STA      ZTEMP3             ;save 2 times integer
D9F3   A5F8                  LDA      ZTEMP4+1
D9F5   85FA                  STA      ZTEMP3+1
D9F7   205ADA                JSR      SIL                ;shift integer left
D9FA   B046 ^DA42            BCS      FPI4               ;if number too big, error

D9FC   205ADA                JSR      SIL                ;shift integer left
D9FF   B041 ^DA42            BCS      FPI4               ;if number too big, error

DA01   18                    CLC
DA02   A5F8                  LDA      ZTEMP4+1           ;8 times integer
DA04   65FA                  ADC      ZTEMP3+1           ;add 2 times integer
DA06   85F8                  STA      ZTEMP4+1           ;10 times integer
DA08   A5F7                  LDA      ZTEMP4
DA0A   65F9                  ADC      ZTEMP3
DA0C   85F7                  STA      ZTEMP4
DA0E   B032 ^DA42            BCS      FPI4               ;if overflow, error

DA10   20B9DC                JSR      GND                ;get next digit
DA13   18                    CLC
DA14   65F8                  ADC      ZTEMP4+1           ;insert digit
DA16   85F8                  STA      ZTEMP4+1
DA18   A5F7                  LDA      ZTEMP4
DA1A   6900                  ADC      #0                 ;add carry
DA1C   B024 ^DA42            BCS      FPI4               ;if overflow, error

DA1E   85F7                  STA      ZTEMP4
DA20   C6F5                  DEC      ZTEMP1             ;decrement count of digits :
DA22   D0C6 ^D9EA            BNE      FPI1               ;if not done

                      ;      Check for round required.

DA24   20B9DC         FPI2   JSR      GND                ;get next digit
DA27   C905                  CMP      #5
DA29   900D ^DA38            BCC      FPI3               ;if digit less than 5, do n:

                      ;      Round.

DA2B   18                    CLC
DA2C   A5F8                  LDA      ZTEMP4+1
DA2E   6901                  ADC      #1                 ;add 1 to round
DA30   85F8                  STA      ZTEMP4+1
DA32   A5F7                  LDA      ZTEMP4
DA34   6900                  ADC      #0
DA36   85F7                  STA      ZTEMP4

                      ;      Return integer.

DA38   A5F8           FPI3   LDA      ZTEMP4+1           ;low integer
DA3A   85D4                  STA      FR0                ;low integer result
DA3C   A5F7                  LDA      ZTEMP4             ;high integer
DA3E   85D5                  STA      FR0+1              ;high integer result
DA40   18                    CLC                         ;indicate success
DA41   60                    RTS                         ;return

                      ;      Return error.
```

```
DA42  38            FPI4    SEC                         ;indicate error
DA43  60                    RTS                         ;return


DA44                        FIX     ZFR0


                    **      ZFR0 - Zero FR0
                    *
                    *       ENTRY   JSR     ZFR0
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

                    ;ZFR0   =       *       ;entry

DA44  A2D4                  LDX     #FR0    ;indicate zero FR0
                    ;       JMP     ZF1     ;zero floating point number, return


DA46                        FIX     ZF1


                    **      ZF1 - Zero Floating Point Number
                    *
                    *       ENTRY   JSR     ZF1
                    *               X = offset to register
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

                    ;ZF1    =       *       ;entry

DA46  A006                  LDY     #6      ;number of bytes to zero
                    ;       JMP     ZXLY    ;zero bytes, return
```

```
                        **      ZXLY - Zero Page Zero Location X for Length Y
                        *
                        *       ENTRY   JSR     ZXLY
                        *               X = offset
                        *               Y = length
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = DA48             ZXLY    =       *       ;entry

DA48  A900                      LDA     #0

DA4A  9500              ZXLY1   STA     $0000,X ;zero byte
DA4C  E8                        INX
DA4D  88                        DEY
DA4E  D0FA ^DA4A                BNE     ZXLY1   ;if not done

DA50  60                        RTS             ;return



                        **      ILP - Initialize Line Buffer Pointer
                        *
                        *       ENTRY   JSR     ILP
                        *
                        *       EXIT
                        *               INBUFF - INBUFF+1 = line buffer address
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = DA51             ILP     =       *               ;entry
DA51  A905                      LDA     #high LBUFF     ;high buffer address
DA53  85F4                      STA     INBUFF+1        ;high line buffer pointer
DA55  A980                      LDA     #low LBUFF      ;low buffer address
DA57  85F3                      STA     INBUFF          ;low line buffer pointer
DA59  60                        RTS                     ;return
```

```
                             **      SIL - Shift Integer Left
                             *
                             *       ENTRY   JSR     SIL
                             *               ZTEMP4 - ZTEMP4+1 = number (high, low) to s:
                             *
                             *       EXIT
                             *               ZTEMP4 - ZTEMP4+1 shifted left 1
                             *
                             *       MODS
                             *               Original Author Unknown
                             *               1. Bring closer to Coding Standard (object :
                             *                  R. K. Nordin 11/01/83

          = DA5A             SIL     =       *                 ;entry
DA5A   18                            CLC
DA5B   26F8                          ROL     ZTEMP4+1          ;shift low
DA5D   26F7                          ROL     ZTEMP4            ;shift high
DA5F   60                            RTS                       ;return


DA60                                 FIX     FSUB



                             **      FSUB - Perform Floating Point Subtract
                             *
                             *       FSUB subtracts FR1 from FR0.
                             *
                             *       ENTRY   JSR     FSUB
                             *               FR0 - FR0+5 = minuend
                             *               FR1 - FR1+5 = subtrahend
                             *
                             *       EXIT
                             *               C set, if error
                             *               C clear, if no error
                             *               FR0 - FR0+5 = difference
                             *
                             *       MODS
                             *               Original Author Unknown
                             *               1. Bring closer to Coding Standard (object :
                             *                  R. K. Nordin 11/01/83

                             ;FSUB   =       *       ;entry

                             ;       Complement sign of subtrahend and add.

DA60   A5E0                          LDA     FR1     ;subtrahend exponent
DA62   4980                          EOR     #$80    ;complement sign of subtrahend
DA64   85E0                          STA     FR1     ;update subtrahend exponent
                             ;       JMP     FADD    ;perform add, return
```

```
DA66                    FIX     FADD




                **      FADD - Perform Floating Point Add
                *
                *       ENTRY   JSR     FADD
                *               FR0 - FR0+5 = augend
                *               FR1 - FR1+5 = addend
                *
                *       EXIT
                *               C set, if error
                *               C clear, if no error
                *               FR0 - FR0+5 = sum
                *
                *       MODS
                *               Original Author Unknown
                *               1. Bring closer to Coding Standard (object :
                *                  R. K. Nordin 11/01/83


                ;FADD   =       *       ;entry

                ;       Initialize.

DA66  A5E0      FADD1   LDA     FR1     ;exponent of addend
DA68  297F              AND     #$7F    ;clear sign of addend mantissa
DA6A  85F7              STA     ZTEMP4  ;save addend exponent
DA6C  A5D4              LDA     FR0     ;exponent of augend
DA6E  297F              AND     #$7F    ;clear sign of augend mantissa
DA70  38                SEC
DA71  E5F7              SBC     ZTEMP4  ;subtract addend exponent
DA73  1010 ^DA85        BPL     FADD3   ;if augend exponent >= addend expon:

                ;       Swap augend and addend.

DA75  A205              LDX     #FPREC-1        ;offset to last byte

DA77  B5D4      FADD2   LDA     FR0,X           ;byte of augend
DA79  B4E0              LDY     FR1,X           ;byte of addend
DA7B  95E0              STA     FR1,X           ;move byte of augend to add:
DA7D  98                TYA
DA7E  95D4              STA     FR0,X           ;move byte of addend to aug:
DA80  CA                DEX
DA81  10F4 ^DA77        BPL     FADD2           ;if not done

DA83  30E1 ^DA66        BMI     FADD1           ;re-initialize

                ;       Check alignment.

DA85  F007 ^DA8E FADD3  BEQ     FADD4   ;if exponent difference zero, alrea:

DA87  C905              CMP     #FMPREC ;mantissa precision
DA89  B019 ^DAA4        BCS     FADD6   ;if exponent difference < mantissa :

                ;       Align.
```

```
DA8B  203EDC                JSR     S1R      ;shift FR1 right

                      ;     Check for like signs of mantissas.

DA8E  F8          FADD4     SED
DA8F  A5D4                  LDA     FR0      ;augend exponent
DA91  45E0                  EOR     FR1      ;EOR with addend exponent
DA93  301E ^DAB3            BMI     FADD8    ;if signs differ, subtract

                      ;     Add.

DA95  A204                  LDX     #FMPREC-1        ;offset to last byte of man:
DA97  18                    CLC

DA98  B5D5        FADD5     LDA     FR0M,X           ;byte of augend mantissa
DA9A  75E1                  ADC     FR1M,X           ;add byte of addend mantiss:
DA9C  95D5                  STA     FR0M,X           ;update byte of result mant:
DA9E  CA                    DEX
DA9F  10F7 ^DA98            BPL     FADD5            ;if not done

DAA1  D8                    CLD
DAA2  B003 ^DAA7            BCS     FADD7            ;if carry, process carry

                      ;     Exit.

DAA4  4C00DC      FADD6     JMP     NORM             ;normalize, return

                      ;     Process carry.

DAA7  A901        FADD7     LDA     #1               ;indicate shift 1
DAA9  203ADC                JSR     S0R              ;shift FR0 right
DAAC  A901                  LDA     #1               ;carry
DAAE  85D5                  STA     FR0M             ;set carry in result

                      ;     Exit.

DAB0  4C00DC                JMP     NORM             ;normalize, return

                      ;     Subtract.

DAB3  A204        FADD8     LDX     #FMPREC-1        ;offset to last byte of man:
DAB5  38                    SEC

DAB6  B5D5        FADD9     LDA     FR0M,X           ;byte of augend mantissa
DAB8  F5E1                  SBC     FR1M,X           ;subtract byte of addend ma:
DABA  95D5                  STA     FR0M,X           ;update byte of result mant:
DABC  CA                    DEX
DABD  10F7 ^DAB6            BPL     FADD9            ;if not done

DABF  9004 ^DAC5            BCC     FADD10           ;if borrow, process borrow

                      ;     Exit.

DAC1  D8                    CLD
DAC2  4C00DC                JMP     NORM             ;normalize, return
```

```
                     ;       Process borrow.

DAC5  A5D4           FADD10  LDA     FR0             ;result exponent
DAC7  4980                   EOR     #$80            ;complement sign of result
DAC9  85D4                   STA     FR0             ;update result exponent

DACB  38                     SEC
DACC  A204                   LDX     #FMPREC-1       ;offset to last byte of man:

DACE  A900           FADD11  LDA     #0
DAD0  F5D5                   SBC     FR0M,X          ;complement byte of result :
DAD2  95D5                   STA     FR0M,X          ;update byte of result mant:
DAD4  CA                     DEX
DAD5  10F7 ^DACE             BPL     FADD11          ;if not done

                     ;       Exit.

DAD7  D8                     CLD
DAD8  4C00DC                 JMP     NORM            ;normalize, return



DADB                         FIX     FMUL



                     **      FMUL - Perform Floating Point Multiply
                     *
                     *       ENTRY   JSR     FMUL
                     *               FR0 - FR0+5 = multiplicand
                     *               FR1 - FR1+5 = multiplier
                     *
                     *       EXIT
                     *               C set, if error
                     *               C clear, if no error
                     *               FR0 - FR0+5 = product
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


                     ;FMUL   =       *       ;entry

                     ;       Check for zero multiplicand.

DADB  A5D4                   LDA     FR0     ;multiplicand exponent
DADD  F045 ^DB24             BEQ     FMUL8   ;if multiplicand exponent zero, res:

                     ;       Check for zero multiplier.

DADF  A5E0                   LDA     FR1     ;multiplier exponent
DAE1  F03E ^DB21             BEQ     FMUL7   ;if multiplier exponent zero, resul:
```

```
DAE3  20CFDC                  JSR     SUE        ;set up exponent
DAE6  38                      SEC
DAE7  E940                    SBC     #$40       ;subtract bias
DAE9  38                      SEC                ;add 1
DAEA  65E0                    ADC     FR1        ;add multiplier exponent
DAEC  3038 ^DB26              BMI     FMUL9      ;if overflow, error

                        ;     Set up.

DAEE  20E0DC                  JSR     SUP        ;set up

                        ;     Compute number of times to add multiplicand.

DAF1  A5DF              FMUL1 LDA     FRE+FPREC-1     ;last byte of FRE
DAF3  290F                    AND     #$0F            ;extract low order digit
DAF5  85F6                    STA     ZTEMP1+1

                        ;     Check for completion.

DAF7  C6F6              FMUL2 DEC     ZTEMP1+1        ;decrement counter
DAF9  3006 ^DB01              BMI     FMUL3           ;if done

DAFB  2001DD                  JSR     FRA10           ;add FR1 to FR0
DAFE  4CF7DA                  JMP     FMUL2           ;continue

                        ;     Compute number of times to add 10 times multiplicans

DB01  A5DF              FMUL3 LDA     FRE+FPREC-1     ;last byte of FRE
DB03  4A                      LSR     A
DB04  4A                      LSR     A
DB05  4A                      LSR     A
DB06  4A                      LSR     A               ;high order digit
DB07  85F6                    STA     ZTEMP1+1

                        ;     Check for completion.

DB09  C6F6              FMUL4 DEC     ZTEMP1+1        ;decrement counter
DB0B  3006 ^DB13              BMI     FMUL5           ;if done

DB0D  2005DD                  JSR     FRA20           ;add FR2 to FR0
DB10  4C09DB                  JMP     FMUL4           ;continue

                        ;     Set up for next set of adds.

DB13  2062DC            FMUL5 JSR     S0ER            ;shift FR0/FRE right

                        ;     Decrement counter and test for completion.

DB16  C6F5                    DEC     ZTEMP1          ;decrement
DB18  D0D7 ^DAF1              BNE     FMUL1           ;if not done

                        ;     Set exponent.

DB1A  A5ED              FMUL6 LDA     EEXP            ;exponent
DB1C  85D4                    STA     FR0             ;result exponent
DB1E  4C04DC                  JMP     NOE             ;normalize, return
```

```
                    ;       Return zero result.

DB21  2044DA        FMUL7   JSR     ZFRO            ;zero FR0

                    ;       Return no error.

DB24  18            FMUL8   CLC                     ;indicate no error
DB25  60                    RTS                     ;return

                    ;       Return error.

DB26  38            FMUL9   SEC                     ;indicate error
DB27  60                    RTS                     ;return



DB28                        FIX     FDIV



                    **      FDIV - Perform Floating Point Divide
                    *
                    *       ENTRY   JSR     FDIV
                    *               FR0 - FR0+5 = dividend
                    *               FR1 - FR1+5 = divisor
                    *
                    *       EXIT
                    *               C clear, if no error
                    *               C set, if error
                    *               FR0 - FR0+5 = quotient
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


                    ;FDIV   =       *       ;entry

                    ;       Check for zero divisor.

DB28  A5E0                  LDA     FR1     ;divisor exponent
DB2A  F0FA ^DB26            BEQ     FMUL9   ;if divisor exponent zero, error

                    ;       Check for zero dividend.

DB2C  A5D4                  LDA     FR0     ;dividend exponent
DB2E  F0F4 ^DB24            BEQ     FMUL8   ;if dividend exponent zero, result :

DB30  20CFDC                JSR     SUE     ;set up exponent
DB33  38                    SEC
DB34  E5E0                  SBC     FR1     ;subtract divisor exponent
DB36  18                    CLC
DB37  6940                  ADC     #$40    ;add bias
DB39  30EB ^DB26            BMI     FMUL9   ;if overflow, error
```

```
DB3B  20E0DC             JSR    SUP     ;set up
DB3E  E6F5               INC    ZTEMP1  ;divide requires extra pass
DB40  4C4EDB             JMP    FDIV3   ;skip shift

                   ;     Shift FR0/FRE left one byte.

DB43  A200         FDIV1 LDX    #0              ;offset to first byte to sh

DB45  B5D5         FDIV2 LDA    FR0+1,X         ;byte to shift
DB47  95D4               STA    FR0,X           ;byte of destination
DB49  E8                 INX
DB4A  E00C               CPX    #FMPREC*2+2     ;number of bytes to shift
DB4C  D0F7 ^DB45         BNE    FDIV2           ;if not done

                   ;     Subtract 2 times divisor from dividend.

DB4E  A005         FDIV3 LDY    #FPREC-1        ;offset to last byte
DB50  38                 SEC
DB51  F8                 SED

DB52  B9DA00       FDIV4 LDA    FRE,Y           ;byte of dividend
DB55  F9E600             SBC    FR2,Y           ;subtract byte of 2*divisor
DB58  99DA00             STA    FRE,Y           ;update byte of dividend
DB5B  88                 DEY
DB5C  10F4 ^DB52         BPL    FDIV4           ;if not done

DB5E  D8                 CLD
DB5F  9004 ^DB65         BCC    FDIV5           ;if difference < 0

DB61  E6D9               INC    QTEMP           ;increment
DB63  D0E9 ^DB4E         BNE    FDIV3           ;continue

                   ;     Adjust.

DB65  200FDD       FDIV5 JSR    FRA2E   ;add FR2 to FR0

                   ;     Shift last byte of quotient left one digit.

DB68  06D9               ASL    QTEMP
DB6A  06D9               ASL    QTEMP
DB6C  06D9               ASL    QTEMP
DB6E  06D9               ASL    QTEMP

                   ;     Subtract divisor from dividend.

DB70  A005         FDIV6 LDY    #FPREC-1        ;offset to last byte
DB72  38                 SEC
DB73  F8                 SED

DB74  B9DA00       FDIV7 LDA    FRE,Y           ;byte of dividend
DB77  F9E000             SBC    FR1,Y           ;subtract byte of divisor
DB7A  99DA00             STA    FRE,Y           ;update byte of dividend
DB7D  88                 DEY
DB7E  10F4 ^DB74         BPL    FDIV7           ;if not done

DB80  D8                 CLD
```

```
DB81  9004 ^DB87              BCC     FDIV8            ;if difference < 0

DB83  E6D9                    INC     QTEMP            ;increment
DB85  D0E9 ^DB70              BNE     FDIV6            ;continue

                      ;       Adjust.

DB87  2009DD          FDIV8   JSR     FRA1E    ;add FR1 to FR0
DB8A  C6F5                    DEC     ZTEMP1   ;decrement
DB8C  D0B5 ^DB43              BNE     FDIV1    ;if not done

                      ;       Clear exponent.

DB8E  2062DC                  JSR     S0ER     ;shift  FR0/FRE right

                      ;       Exit.

DB91  4C1ADB                  JMP     FMUL6



                      **      GNC - Get Next Character
                      *
                      *       ENTRY   JSR     GNC
                      *               INBUFF - INBUFF+1 = line buffer pointer
                      *               CIX = offset to character
                      *
                      *       EXIT
                      *               C set, if character not numeric
                      *               A = non-numeric character
                      *               C clear, if character numeric
                      *               CIX = offset to next character
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = DB94          GNC     =       *                ;entry
DB94  204FDB                  JSR     TNC              ;test for numeric character
DB97  A4F2                    LDY     CIX              ;offset
DB99  9002 ^DB9D              BCC     ICX              ;if numeric, increment offs:

DB9B  B1F3                    LDA     (INBUFF),Y       ;character
                      ;       JMP     ICX              ;increment offset, return
```

```
                        **      ICX - Increment Character Offset
                        *
                        *       ENTRY   JSR     ICX
                        *               Y = offset
                        *
                        *       EXIT
                        *               CIX = offset to next character
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = DB9D          ICX     =       *       ;entry
DB9D    C8                      INY             ;increment offset
DB9E    84F2                    STY     CIX     ;offset
DBA0    60                      RTS             ;return


                        **      SLB - Skip Leading Blanks
                        *
                        *       ENTRY   JSR     SLB
                        *               INBUFF - INBUFF+1 = line buffer pointer
                        *               CIX = offset
                        *
                        *       EXIT
                        *               CIX = offset to first non-blank character
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = DBA1          SLB     =       *               ;entry

                        ;       Initialize.

DBA1    A4F2                    LDY     CIX             ;offset to character
DBA3    A920                    LDA     #' '

                        ;       Search for first non-blank character.

DBA5    D1F3            SLB1    CMP     (INBUFF),Y      ;character
DBA7    D003 ^DBAC              BNE     SLB2            ;if non-blank character

DBA9    C8                      INY
DBAA    D0F9 ^DBA5              BNE     SLB1            ;if not done

                        ;       Exit.

DBAC    84F2            SLB2    STY     CIX             ;offset to first non-blank :
DBAE    60                      RTS                     ;return
```

```
                        **      TNC - Test for Numeric Character
                        *
                        *       ENTRY   JSR     TNC
                        *               INBUFF - INBUFF+1 = line buffer pointer
                        *               CIX = offset
                        *
                        *       EXIT
                        *               C set, if numeric
                        *               C clear if non-numeric
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = DBAF          TNC     =       *               ;entry
DBAF  A4F2                      LDY     CIX             ;offset
DBB1  B1F3                      LDA     (INBUFF),Y      ;character
DBB3  38                        SEC
DBB4  E930                      SBC     #'0'
DBB6  9018 ^DBD0                BCC     TVN2            ;if < "0", return failure

DBB8  C90A                      CMP     #'9'-'0'+1      ;return success or failure
DBBA  60                        RTS                     ;return



                        **      TVN - Test for Valid Number Character
                        *
                        *       ENTRY   JSR     TVN
                        *
                        *       EXIT
                        *               C set, if not number
                        *               C clear, if number
                        *
                        *       NOTES
                        *               Problem: bytes wasted by BCC TVN5.
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = DBBB          TVN     =       *       ;entry

                        ;       Initialize.

DBBB  A5F2                      LDA     CIX     ;offset
DBBD  48                        PHA             ;save offset

                        ;       Check next character.

DBBE  2094DB                    JSR     GNC     ;get next character
DBC1  901F ^DBE2                BCC     TVN5    ;if numeric, return success
```

```
DBC3  C92E                  CMP     #'.'
DBC5  F014 ^DBDB            BEQ     TVN4     ;if ".", check next character

DBC7  C92B                  CMP     #'+'
DBC9  F007 ^DBD2            BEQ     TVN3     ;if "+", check next character

DBCB  C92D                  CMP     #'-'
DBCD  F003 ^DBD2            BEQ     TVN3     ;if "-", check next character

                    ;       Clean stack.

DBCF  68            TVN1    PLA              ;clean stack

                    ;       Return failure.

DBD0  38            TVN2    SEC              ;indicate failure
DBD1  60                    RTS              ;return

                    ;       Check character after "+" or "-".

DBD2  2094DB        TVN3    JSR     GNC      ;get next character
DBD5  9008 ^DBE2            BCC     TVN5     ;if numeric, return success

DBD7  C92E                  CMP     #'.'
DBD9  D0F4 ^DBCF            BNE     TVN1     ;if not ".", return failure

                    ;       Check character after ".".

DBDB  2094DB        TVN4    JSR     GNC      ;get next character
DBDE  9002 ^DBE2            BCC     TVN5     ;if numeric, return success

DBE0  B0ED ^DBCF            BCS     TVN1     ;return failure

                    ;       Return success.

DBE2  68            TVN5    PLA              ;saved offset
DBE3  85F2                  STA     CIX      ;restore offset
DBE5  18                    CLC              ;indicate success
DBE6  60                    RTS              ;return




                    **      S2L - Shift FR2 Left One Digit
                    *
                    *       ENTRY   JSR     S2L
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = DBE7        S2L     =       *        ;entry
DBE7  A2E7                  LDX     #FR2+1   ;indicate shift of FR2 mantissa
```

```
DBE9  D002 ^DBED           BNE     SML      ;shift mantissa left 1 digit, retur:


                     **     SOL - Shift FR0 Left One Digit
                     *
                     *      ENTRY   JSR     SOL
                     *
                     *      MODS
                     *              Original Author Unknown
                     *              1. Bring closer to Coding Standard (object :
                     *                 R. K. Nordin 11/01/83


      = DBEB        SOL     =       *        ;entry
DBEB  A205                  LDX     #FR0M    ;indicate shift of FR0 mantissa
                     ;      JMP     SML      ;shift mantissa left 1 digit, retur:


                     **     SML - Shift Mantissa Left One Digit
                     *
                     *      ENTRY   JSR     SML
                     *
                     *      EXIT
                     *              FRX = excess digit
                     *
                     *      MODS
                     *              Original Author Unknown
                     *              1. Bring closer to Coding Standard (object :
                     *                 R. K. Nordin 11/01/83


      = DBED        SML     =       *        ;entry
DBED  A004                  LDY     #4       ;number of bits to shift

DBEF  18            SML2    CLC
DBF0  3604                  ROL     $0004,X ;shift 5th byte left 1 bit
DBF2  3603                  ROL     $0003,X ;shift 4th byte left 1 bit
DBF4  3602                  ROL     $0002,X ;shift 3rd byte left 1 bit
DBF6  3601                  ROL     $0001,X ;shift 2nd byte left 1 bit
DBF8  3600                  ROL     $0000,X ;shift 1st byte left 1 bit
DBFA  26EC                  ROL     FRX      ;shift excess digit left 1 bit
DBFC  88                    DEY
DBFD  D0F0 ^DBEF            BNE     SML2     ;if not done

DBFF  60                    RTS              ;return
```

```
                          **      NORM - Normalize FR0
                          *
                          *       ENTRY   JSR     NORM
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

         = DC00          NORM    =       *               ;entry
DC00  A200                       LDX     #0
DC02  86DA                       STX     FRE             ;byte to shift in
                          ;      JMP     N0E             ;normalize FR0/FRE, return



                          **      N0E - Normalize FR0/FRE
                          *
                          *       ENTRY   JSR     N0E
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83

         = DC04          N0E     =       *               ;entry
DC04  A204                       LDX     #FMPREC-1       ;mantissa size
DC06  A5D4                       LDA     FR0             ;exponent
DC08  F02E ^DC38                 BEQ     N0E5            ;if exponent zero, number i:

DC0A  A5D5               N0E1    LDA     FR0M            ;first byte of mantissa
DC0C  D01A ^DC28                 BNE     N0E3            ;if not zero, no shift

                          ;       Shift mantissa left 1 byte.

DC0E  A000                       LDY     #0              ;offset to first byte of ma:

DC10  B9D600             N0E2    LDA     FR0M+1,Y        ;byte to shift
DC13  99D500                     STA     FR0M,Y          ;byte of destination
DC16  C8                         INY
DC17  C005                       CPY     #FMPREC         ;size of mantissa
DC19  90F5 ^DC10                 BCC     N0E2            ;if not done

                          ;       Decrement exponent and check for completion.

DC1B  C6D4                       DEC     FR0             ;decrement exponent
DC1D  CA                         DEX
DC1E  D0EA ^DC0A                 BNE     N0E1            ;if not done

                          ;       Check first byte of mantissa.

DC20  A5D5                       LDA     FR0M    ;first byte of mantissa
DC22  D004 ^DC28                 BNE     N0E3    ;if mantissa not zero
```

```
                        ;       Zero exponent.

DC24  85D4                      STA     FR0       ;zero exponent
DC26  18                        CLC
DC27  60                        RTS               ;return

                        ;       Check for overflow.

DC28  A5D4              NOE3    LDA     FR0       ;exponent
DC2A  297F                      AND     #$7F      ;clear sign
DC2C  C971                      CMP     #$40+49   ;bias+49
DC2E  9001 ^DC31                BCC     NOE4      ;if exponent < 49, no overflow

                        ;       Return error.

                        ;       SEC               ;indicate error
DC30  60                        RTS               ;return

                        ;       Check for underflow.

DC31  C90F              NOE4    CMP     #$40-49
DC33  B003 ^DC36                BCS     NOE5      ;if exponent >= -49, no underflow

                        ;       Zero result.

DC35  2044DA                    JSR     ZFR0      ;zero FR0

                        ;       Exit.

DC38  18                NOE5    CLC               ;indicate no error
DC39  60                        RTS               ;return



                        **      S0R - Shift FR0 Right
                        *
                        *       ENTRY   JSR     S0R
                        *               A = shift count
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = DC3A            S0R     =       *         ;entry
DC3A  A2D4                      LDX     #FR0      ;indicate shift of FR0
DC3C  D002 ^DC40                BNE     SRR       ;shift register right, return
```

```
                        **      S1R - Shift FR1 Right
                        *
                        *       ENTRY    JSR     S1R
                        *                A = shift count
                        *
                        *       MODS
                        *                Original Author Unknown
                        *                1. Bring closer to Coding Standard (object :
                        *                   R. K. Nordin 11/01/83


       = DC3E           S1R     =        *       ;entry
DC3E   A2E0                     LDX      #FR1    ;indicate shift of FR1
                        ;       JMP      SRR     ;shift register right, return



                        **      SRR - Shift Register Right
                        *
                        *       ENTRY    JSR     SRR
                        *                X = offset to register.
                        *                A = shift count
                        *
                        *       MODS
                        *                Original Author Unknown
                        *                1. Bring closer to Coding Standard (object :
                        *                   R. K. Nordin 11/01/83


       = DC40           SRR     =        *                 ;entry
DC40   86F9                     STX      ZTEMP3            ;register
DC42   85F7                     STA      ZTEMP4            ;shift count
DC44   85F8                     STA      ZTEMP4+1          ;save shift count

DC46   A004             SRR1    LDY      #FMPREC-1         ;mantissa size-1

DC48   B504             SRR2    LDA      $0004,X           ;byte to shift
DC4A   9505                     STA      $0005,X           ;byte of destination
DC4C   CA                       DEX
DC4D   88                       DEY
DC4E   D0F8 ^DC48               BNE      SRR2              ;if not done

DC50   A900                     LDA      #0
DC52   9505                     STA      $0005,X           ;first byte of mantissa
DC54   A6F9                     LDX      ZTEMP3            ;register
DC56   C6F7                     DEC      ZTEMP4            ;decrement shift count
DC58   D0EC ^DC46               BNE      SRR1              ;if not done

                        ;       Adjust exponent.

DC5A   B500                     LDA      $0000,X           ;exponent
DC5C   18                       CLC
DC5D   65F8                     ADC      ZTEMP4+1          ;subtract shift count
DC5F   9500                     STA      $0000,X           ;update exponent
DC61   60                       RTS                        ;return
```

```
                           **      S0ER - Shift FR0/FRE Right
                           *
                           *       ENTRY    JSR      S0ER
                           *
                           *       MODS
                           *                Original Author Unknown
                           *                1. Bring closer to Coding Standard (object :
                           *                   R. K. Nordin 11/01/83


         = DC62            S0ER    =        *                 ;entry
DC62  A20A                         LDX      #FMPREC*2         ;number of bytes to shift

DC64  85D4                 S0ER1   LDA      FR0,X             ;byte to shift
DC66  95D5                         STA      FR0+1,X           ;byte of destination
DC68  CA                           DEX
DC69  10F9 ^DC64                   BPL      S0ER1             ;if not done

DC6B  A900                         LDA      #0
DC6D  85D4                         STA      FR0               ;shift in 0
DC6F  60                           RTS                        ;return



                           **      COA - Convert FR0 to ASCII
                           *
                           *       ENTRY    JSR      COA
                           *                A = decimal point position
                           *
                           *       MODS
                           *                Original Author Unknown
                           *                1. Bring closer to Coding Standard (object :
                           *                   R. K. Nordin 11/01/83


         = DC70            COA     =        *        ;entry

                           ;       Initialize.

DC70  85F7                         STA      ZTEMP4   ;decimal point position counter
DC72  A200                         LDX      #0       ;offset to first byte of FROM
DC74  A000                         LDY      #0       ;offset to first byte of LBUF

                           ;       Convert next byte.

DC76  2093DC               COA1    JSR      TDP      ;test for decimal point
DC79  38                           SEC
DC7A  E901                         SBC      #1       ;decrement deciaml point position
DC7C  85F7                         STA      ZTEMP4   ;update deciaml point position coun:

                           ;       Convert first digit of next byte.

DC7E  85D5                         LDA      FROM,X   ;byte
DC80  4A                           LSR      A
DC81  4A                           LSR      A
DC82  4A                           LSR      A
```

```
DC83  4A                    LSR     A         ;first digit
DC84  209DDC                JSR     SNL       ;store number in line buffer

                    ;       Convert second digit of next byte.

DC87  B5D5                  LDA     FROM,X    ;byte
DC89  290F                  AND     #$0F      ;extract second digit
DC8B  209DDC                JSR     SNL       ;store number in line buffer
DC8E  E8                    INX
DC8F  E005                  CPX     #FMPREC   ;nuber of bytes
DC91  90E3 ^DC76            BCC     COA1      ;if not done

                    ;       Exit.

                    ;       JMP     TDP       ;test for decimal point, return



                    **      TDP - Test for Decimal Point
                    *
                    *       ENTRY   JSR     TDP
                    *               ZTEMP4 = decimal point position counter
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = DC93        TDP     =       *         ;entry

                    ;       Check decimal point position counter.

DC93  A5F7                  LDA     ZTEMP4    ;decimal point position counter
DC95  D005 ^DC9C            BNE     TDP1      ;if not decimal point position, exi:

                    ;       Insert decimal point.

DC97  A92E                  LDA     #'.'
DC99  209FDC                JSR     SAL       ;store ASCII character in line buff:

                    ;       Exit.

DC9C  60            TDP1    RTS               ;return
```

```
                    **      SNL - Store Number in Line Buffer
                    *
                    *       ENTRY   JSR     SNL
                    *               A = digit to store
                    *               Y = offset
                    *
                    *       EXIT
                    *               ASCII digit placed in line buffer
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = DC9D        SNL     =       *       ;entry
DC9D  0930                  ORA     #$30    ;convert digit to ASCII
                    ;       JMP     SAL     ;store ASCII character in line buff:


                    **      SAL - Store ASCII Character in Line Buffer
                    *
                    *       ENTRY   JSR     SAL
                    *               Y = offset
                    *               A = character
                    *
                    *       EXIT
                    *               Character placed in line buffer
                    *               Y = incremented offset
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = DC9F        SAL     =       *       ;entry
DC9F  998005                STA     LBUFF,Y ;store character in line buffer
DCA2  C8                    INY             ;increment offset
DCA3  60                    RTS             ;return
```

```
                    **      FNZ - Find Last Non-zero Character in Line Buffer
                    *
                    *       FNZ returns the last non-zero character.  If the la:
                    *       non-zero character is ".", FNZ returns the characte:
                    *       preceding the ".".  If no other non-zero character :
                    *       encountered, FNZ returns the first character.
                    *
                    *       ENTRY   JSR     FNZ
                    *
                    *       EXIT
                    *               A = character
                    *               X = offset to character
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = DCA4        FNZ     =       *       ;entry

                    ;       Initialize.

DCA4  A20A                  LDX     #10     ;offset to last possible character

                    ;       Check next character.

DCA6  BD8005        FNZ1    LDA     LBUFF,X ;character
DCA9  C92E                  CMP     #'.'
DCAB  F007 ^DCB4            BEQ     FNZ2    ;if ".", return preceding character

DCAD  C930                  CMP     #'0'
DCAF  D007 ^DCB8            BNE     FNZ3    ;if not "0", exit

                    ;       Decrement offset and check for completion.

DCB1  CA                    DEX
DCB2  D0F2 ^DCA6            BNE     FNZ1    ;if not done

                    ;       Return character preceding "." or first character.

DCB4  CA            FNZ2    DEX             ;offset to character
DCB5  BD8005                LDA     LBUFF,X ;character

                    ;       Exit.

DCB8  60            FNZ3    RTS             ;return
```

```
                          **      GND - Get Next Digit
                          *
                          *       ENTRY   JSR     GND
                          *               FR0 - FR0+5 = number
                          *
                          *       EXIT
                          *               A = digit
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


      = DC89      GND     =       *          ;entry
DCB9  20EBDB              JSR     SOL        ;shift FR0 left 1 digit
DCBC  A5FC                LDA     FRX        ;excess digit
DCBE  290F                AND     #$0F       ;extract low order digit
DCC0  60                  RTS                ;return



                          **      DLP - Decrement Line Buffer Pointer
                          *
                          *       ENTRY   JSR     DLP
                          *               INBUFF - INBUFF+1 = line buffer pointer
                          *
                          *       EXIT
                          *               INBUFF - INBUFF+1 = incremented line buffer:
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


      = DCC1      DLP     =       *                    ;entry
DCC1  38                  SEC
DCC2  A5F3                LDA     INBUFF               ;line buffer pointer
DCC4  E901                SBC     #1                   ;subtract 1
DCC6  85F3                STA     INBUFF               ;update line buffer pointer
DCC8  A5F4                LDA     INBUFF+1
DCCA  E900                SBC     #0
DCCC  85F4                STA     INBUFF+1
DCCE  60                  RTS                          ;return
```

```
                        **      SUE - Set Up Exponent for Multiply or Divide
                        *
                        *       ENTRY   JSR     SUE
                        *
                        *       EXIT
                        *               A = FR0 exponent (without sign)
                        *               FR1 = FR1 exponent (without sign)
                        *               FRSIGN = sign of result
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = DCCF        SUE     =       *       ;entry
DCCF  A5D4                  LDA     FR0     ;FR0 exponent
DCD1  45E0                  EOR     FR1     ;EOR with FR1 exponent
DCD3  2980                  AND     #$80    ;extract sign
DCD5  85EE                  STA     FRSIGN  ;sign of result
DCD7  06E0                  ASL     FR1     ;shift out FR1 sign
DCD9  46E0                  LSR     FR1     ;FR1 exponent without sign
DCDB  A5D4                  LDA     FR0     ;FR0 exponent
DCDD  297F                  AND     #$7F    ;FR0 exponent without sign
DCDF  60                    RTS             ;return



                        **      SUP - Set Up for Multiply or Divide
                        *
                        *       ENTRY   JSR     SUP
                        *               A = exponent
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = DCE0        SUP     =       *       ;entry
DCE0  05EE                  ORA     FRSIGN  ;place sign in exponent
DCE2  85ED                  STA     EEXP    ;exponent
DCE4  A900                  LDA     #0
DCE6  85D4                  STA     FR0     ;clear FR0 exponent
DCE8  85E0                  STA     FR1     ;clear FR0 exponent
DCEA  2028DD                JSR     M12     ;move FR1 to FR2
DCED  20E7DB                JSR     S2L     ;shift FR2 left 1 digit
DCF0  A5EC                  LDA     FRX     ;excess digit
DCF2  290F                  AND     #$0F    ;extract low order digit
DCF4  85E6                  STA     FR2     ;shift in low order digit
DCF6  A905                  LDA     #FMPREC ;mantissa size
DCF8  85F5                  STA     ZTEMP1  ;mantissa size
DCFA  2034DD                JSR     M0E     ;move FR0 to FRE
DCFD  2044DA                JSR     ZFR0    ;zero FR0
DD00  60                    RTS             ;return
```

```
                                **      FRA10 - Add FR1 to FR0
                                *
                                *       ENTRY   JSR     FRA10
                                *               FR0 - FR0+5 = augend
                                *               FR1 - FR1+5 = addend
                                *
                                *       EXIT
                                *               FR0 - FR0+5 = sum
                                *
                                *       MODS
                                *               Original Author Unknown
                                *               1. Bring closer to Coding Standard (object :
                                *                  R. K. Nordin 11/01/83

     = DD01         FRA10       =       *                   ;entry
DD01  A209                      LDX     #FR0+FPREC-1        ;offset to last byte of FR0
DD03  D006 ^DD0B                BNE     F1R



                                **      FRA20 - Add FR2 to FR0
                                *
                                *       ENTRY   JSR     FRA20
                                *               FR0 - FR0+5 = augend
                                *               FR2 - FR2+5 = addend
                                *
                                *       EXIT
                                *               FR0 - FR0+5 = sum
                                *
                                *       MODS
                                *               Original Author Unknown
                                *               1. Bring closer to Coding Standard (object :
                                *                  R. K. Nordin 11/01/83

     = DD05         FRA20       =       *                   ;entry
DD05  A209                      LDX     #FR0+FPREC-1        ;offset to last byte of FR0
DD07  D008 ^DD11                BNE     F2R



                                **      FRA1E - Add FR1 to FRE
                                *
                                *       ENTRY   JSR     FRA1E
                                *               FRE - FRE+5 = augend
                                *               FR1 - FR1+5 = addend
                                *
                                *       EXIT
                                *               FRE - FRE+5 = sum
                                *
                                *       MODS
                                *               Original Author Unknown
                                *               1. Bring closer to Coding Standard (object :
                                *                  R. K. Nordin 11/01/83
     = DD09         FRA1E     =       *                 ;entry
DD09  A2DF                    LDX     #FRE+FPREC-1      ;offset to last byte of FRE
                    ;         JMP     F1R               ;add FR1 to register, retur:



                    **        F1R - Add FR1 to Register
                    *
                    *         ENTRY   JSR     F1R
                    *                 X = offset to last byte of augend register
                    *                 FR1 - FR1+5 = addend
                    *
                    *         EXIT
                    *                 Sum in augend register
                    *
                    *         MODS
                    *                 Original Author Unknown
                    *                 1. Bring closer to Coding Standard (object :
                    *                    R. K. Nordin 11/01/83

     = DD0B         F1R       =       *                 ;entry
DD0B  A0E5                    LDY     #FR1+FPREC-1      ;offset to last byte of FR1
DD0D  D004 ^DD13              BNE     FARR



                    **        FRA2E - Add FR2 to FRE
                    *
                    *         ENTRY   JSR     FRA2E
                    *                 FRE - FRE+5 = augend
                    *                 FR2 - FR2+5 = addend
                    *
                    *         EXIT
                    *                 FRE - FRE+5 = sum
                    *
                    *         MODS
                    *                 Original Author Unknown
                    *                 1. Bring closer to Coding Standard (object :
                    *                    R. K. Nordin 11/01/83

     = DD0F         FRA2E     =       *                 ;entry
DD0F  A2DF                    LDX     #FRE+FPREC-1      ;offset to last byte of FRE
                    ;         JMP     F2R
```

```
                        **      F2R - Add FR2 to Register
                        *
                        *       ENTRY   JSR     F2R
                        *               X = offset to last byte of augend register
                        *               FR2 - FR2+5 = addend
                        *
                        *       EXIT
                        *               Sum in augend register
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = DD11          F2R     =       *               ;entry
DD11    A0EB                    LDY     #FR2+FPREC-1    ;offset to last byte of FR2
                        ;       JMP     FARR



                        **      FARR - Add Register to Register
                        *
                        *       ENTRY   JSR     FARR
                        *               X = offset to last byte of augend register
                        *               Y = offset to last byte of addend register
                        *
                        *       EXIT
                        *               Sum in augend register
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = DD13          FARR    =       *               ;entry

                        ;       Initialize.

DD13    A905                    LDA     #FPREC-1        ;floating point number size:
DD15    85F7                    STA     ZTEMP4          ;byte count
DD17    18                      CLC
DD18    F8                      SED

                        ;       Add.

DD19    B500            FARR1   LDA     $0000,X         ;byte of augend
DD1B    790000                  ADC     $0000,Y         ;add byte of addend
DD1E    9500                    STA     $0000,X         ;update byte of augend
DD20    CA                      DEX
DD21    88                      DEY
DD22    C6F7                    DEC     ZTEMP4          ;decrement byte count
DD24    10F3 ^DD19              BPL     FARR1           ;if not done

                        ;       Exit.
```

```
DD26  D8                        CLD
DD27  60                        RTS                         ;return



                    **          M12 - Move FR1 to FR2
                    *
                    *           ENTRY   JSR     M12
                    *                   FR1 - FR1+5 = number to move
                    *
                    *           EXIT
                    *                   FR2 - FR2+5 = moved number
                    *
                    *           MODS
                    *                   Original Author Unknown
                    *                   1. Bring closer to Coding Standard (object :
                    *                      R. K. Nordin 11/01/83

      = DD28        M12         =       *                   ;entry
DD28  A005                      LDY     #FPREC-1            ;offset to last byte

DD2A  B9E000        M121        LDA     FR1,Y               ;byte of source
DD2D  99E600                    STA     FR2,Y               ;byte of destination
DD30  88                        DEY
DD31  10F7 ^DD2A                BPL     M121                ;if not done

DD33  60                        RTS                         ;return



                    **          M0E - Move FR0 to FRE
                    *
                    *           ENTRY   JSR     M0E
                    *                   FR0 - FR0+5 = number to move
                    *
                    *           EXIT
                    *                   FRE - FRE+5 = moved number
                    *
                    *           MODS
                    *                   Original Author Unknown
                    *                   1. Bring closer to Coding Standard (object :
                    *                      R. K. Nordin 11/01/83

      = DD34        M0E         =       *                   ;entry
DD34  A005                      LDY     #FPREC-1            ;offset to last byte

DD36  B9D400        M0E1        LDA     FR0,Y               ;byte of source
DD39  99DA00                    STA     FRE,Y               ;byte of destination
DD3C  88                        DEY
DD3D  10F7 ^DD36                BPL     M0E1                ;if not done
```

```
DD3F  60                  RTS                        ;return



DD40                      FIX      PLYEVL



                  **      PLYEVL - Evaluate Polynomial
                  *
                  *       Y = A(0)+A(1)*X+A(2)*X^2+...+A(N)*X^N
                  *
                  *       ENTRY    JSR      PLYEVL
                  *                X = low address of coefficient table
                  *                Y = high address of coefficient table
                  *                FR0 - FR0+5 = X argument
                  *                A = N+1
                  *
                  *       EXIT
                  *                FR0 - FR0+5 = Y result
                  *
                  *       MODS
                  *                Original Author Unknown
                  *                1. Bring closer to Coding Standard (object :
                  *                   R. K. Nordin 11/01/83


                  ;PLYEVL =        *                     ;entry

DD40  86FE                STX      FPTR2             ;save pointer to coefficien:
DD42  84FF                STY      FPTR2+1
DD44  85EF                STA      PLYCNT            ;degree
DD46  A2E0                LDX      #low PLYARG
DD48  A005                LDY      #high PLYARG
DD4A  20A7DD              JSR      FSTOR             ;save argument
DD4D  20B6DD              JSR      FMOVE             ;move argument to FR1
DD50  A6FE                LDX      FPTR2
DD52  A4FF                LDY      FPTR2+1
DD54  2089DD              JSR      FLD0R             ;initialize sum in FR0
DD57  C6EF                DEC      PLYCNT            ;decrement degree
DD59  F02D ^DD88          BEQ      PLY3              ;if complete, exit

DD5B  20DBDA      PLY1    JSR      FMUL              ;argument times current sum
DD5E  B028 ^DD88          BCS      PLY3              ;if overflow

DD60  18                  CLC
DD61  A5FE                LDA      FPTR2             ;current low coefficient ad:
DD63  6906                ADC      #FPREC            ;add floating point number :
DD65  85FE                STA      FPTR2             ;update low coefficient add:
DD67  9006 ^DD6F          BCC      PLY2              ;if no carry

DD69  A5FF                LDA      FPTR2+1           ;current high coefficceint :
DD6B  6900                ADC      #0                ;adjust high coefficient ad:
DD6D  85FF                STA      FPTR2+1           ;update high coefficient ad:
```

```
DD6F  A6FE        PLY2    LDX     FPTR2           ;low coefficient address
DD71  A4FF                LDY     FPTR2+1         ;high coefficient address
DD73  2098DD              JSR     FLD1R           ;get next coefficient
DD76  2066DA              JSR     FADD            ;add coefficient to argumen:
DD79  B00D ^DD88          BCS     PLY3            ;if overflow

DD7B  C6EF                DEC     PLYCNT          ;decrement degree
DD7D  F009 ^DD88          BEQ     PLY3            ;if complete, exit

DD7F  A2E0                LDX     #low PLYARG     ;low argument address
DD81  A005                LDY     #high PLYARG    ;high argument address
DD83  2098DD              JSR     FLD1R           ;get argument
DD86  30D3 ^DD5B          BMI     PLY1            ;continue

DD88  60          PLY3    RTS                     ;return



DD89                      FIX     FLD0R



                  **      FLD0R - Load FR0
                  *
                  *       ENTRY   JSR     FLD0R
                  *               X = low pointer
                  *               Y = high pointer
                  *
                  *       EXIT
                  *               FR0 loaded
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


                  ;FLD0R  =       *               ;entry
DD89  86FC                STX     FLPTR           ;low pointer
DD8B  84FD                STY     FLPTR+1         ;high pointer
                  ;       JMP     FLD0P           ;load FR0, return
```

```
DD8D                           FIX     FLD0P



                       **      FLD0P - Load FR0
                       *
                       *       ENTRY   JSR     FLD0P
                       *               FLPTR - FLPTR+1 = pointer
                       *
                       *       EXIT
                       *               FR0 loaded
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83

                       ;FLD0P  =       *                ;entry

DD8D  A005                     LDY     #FPREC-1         ;offset to last byte

DD8F  B1FC             FLD01   LDA     (FLPTR),Y        ;byte of source
DD91  99D400                   STA     FR0,Y            ;byte of destination
DD94  88                       DEY
DD95  10F8 ^DD8F               BPL     FLD01            ;if not done

DD97  60                       RTS                      ;return



DD98                           FIX     FLD1R



                       **      FLD1R - Load FR1
                       *
                       *       ENTRY   JSR     FLD1R
                       *               X = low pointer
                       *               Y = high pointer
                       *
                       *       EXIT
                       *               FR1 loaded
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83

                       ;FLD1R  =       *                ;entry
DD98  86FC                     STX     FLPTR            ;low pointer
DD9A  84FD                     STY     FLPTR+1          ;high pointer
                       ;       JMP     FLD1P            ;load FR1, return
```

```
DD9C                         FIX     FLD1P



                     **      FLD1P - Load FR1
                     *
                     *       ENTRY   JSR     FLD1P
                     *               FLPTR - FLPTR+1 = pointer
                     *
                     *       EXIT
                     *               FR1 loaded
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83

                     ;FLD1P  =       *               ;entry

DD9C  A005                   LDY     #FPREC-1        ;offset to last byte

DD9E  B1FC           FLD11   LDA     (FLPTR),Y       ;byte of source
DDA0  99E000                 STA     FR1,Y           ;byte of destination
DDA3  88                     DEY
DDA4  10F8 ^DD9E             BPL     FLD11           ;if not done

DDA6  60                     RTS                     ;return



DDA7                         FIX     FSTOR



                     **      FSTOR - Store FR0
                     *
                     *       ENTRY   JSR     FSTOR
                     *               FR0 - FR0+5 = number
                     *               X = low pointer
                     *               Y = high pointer
                     *
                     *       EXIT
                     *               FR0 stored
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83

                     ;FSTOR  =       *               ;entry
DDA7  86FC                   STX     FLPTR           ;low pointer
DDA9  84FD                   STY     FLPTR+1         ;high pointer
```

```
                        ;       JMP     FSTOP           ;store FR0, return


DDAB                            FIX     FSTOP


                        **      FSTOP - Store FR0
                        *
                        *       ENTRY   JSR     FSTOP
                        *               FR0 - FR0+5 = number
                        *               FLPTR - FLPTR+1 = pointer
                        *
                        *       EXIT
                        *               FR0 stored
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

                        ;FSTOP  =       *               ;entry

DDAB  A005                      LDY     #FPREC-1        ;offset to last byte

DDAD  B9D400            FST01   LDA     FR0,Y           ;byte of source
DDB0  91FC                      STA     (FLPTR),Y       ;byte of destination
DDB2  88                        DEY
DDB3  10F8 ^DDAD                BPL     FST01           ;if not done

DDB5  60                        RTS                     ;return


DDB6                            FIX     FMOVE


                        **      FMOVE - Move FR0 to FR1
                        *
                        *       ENTRY   JSR     FMOVE
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

                        ;FMOVE  =       *               ;entry

DDB6  A205                      LDX     #FPREC-1        ;offset to last byte
```

```
DDB8  B5D4         FM01    LDA     FR0,X             ;byte of source
DDBA  95E0                 STA     FR1,X             ;byte of destination
DDBC  CA                   DEX
DDBD  10F9 ^DDB8           BPL     FM01              ;if not done

DDBF  60                   RTS                       ;return



DDC0                       FIX     EXP


                   **      EXP - Compute Power of e
                   *
                   *       ENTRY   JSR     EXP
                   *
                   *       MODS
                   *               Original Author Unknown
                   *               1. Bring closer to Coding Standard (object :
                   *                  R. K. Nordin 11/01/83

                   ;EXP    =       *                 ;entry

                   ;       Initialize.

DDC0  A289                 LDX     #low LOG10E       ;base 10 logarithm of e
DDC2  A0DE                 LDY     #high LOG10E
DDC4  2098DD               JSR     FLD1R             ;load FR1

                   ;       Compute X*LOG10(E).

DDC7  20DBDA               JSR     FMUL              ;multiply
DDCA  B07F ^DE4B           BCS     EXP6              ;if overflow, error

                   ;       Compute result = 10^(X*LOG10(E)).

                   ;       JMP     EXP10             ;compute power of 10, retur:
```

```
DDCC                        FIX     EXP10



                  **        EXP10 - Compute Power of 10
                  *
                  *         ENTRY   JSR     EXP10
                  *
                  *         MODS
                  *                 Original Author Unknown
                  *                 1. Bring closer to Coding Standard (object :
                  *                    R. K. Nordin 11/01/83


                  ;EXP10    =       *               ;entry

                  ;         Initialize.

DDCC  A900                  LDA     #0
DDCE  85F1                  STA     XFMFLG          ;zero integer part
DDD0  A5D4                  LDA     FR0
DDD2  85F0                  STA     SGNFLG          ;save argument sign
DDD4  297F                  AND     #$7F            ;extract absolute value
DDD6  85D4                  STA     FR0             ;update argument

                  ;         Check for argument less than 1.

DDD8  38                    SEC
DDD9  E940                  SBC     #$40            ;subtract bias
DDDB  3026 ^DE03            BMI     EXP1            ;if argument < 1

                  ;         Extract integer and fractional parts of exponent.

DDDD  C904                  CMP     #FPREC-2
DDDF  106A ^DE46            BPL     EXP6            ;if argument too big, error

DDE1  A2E6                  LDX     #low FPSCR
DDE3  A005                  LDY     #high FPSCR
DDE5  20A7DD                JSR     FSTOR           ;save argument
DDE8  20D2D9                JSR     FPI             ;convert argument to intege:
DDEB  A5D4                  LDA     FR0
DDED  85F1                  STA     XFMFLG          ;save interger part
DDEF  A5D5                  LDA     FR0+1           ;most significant byte of i:
DDF1  D058 ^DE46            BNE     EXP6            ;if integer part too large,:

DDF3  20AAD9                JSR     IFP             ;convert integer part to fl:
DDF6  20B6DD                JSR     FMOVE
DDF9  A2E6                  LDX     #low FPSCR
DDFB  A005                  LDY     #high FPSCR
DDFD  2089DD                JSR     FLDOR           ;argument
DE00  2060DA                JSR     FSUB            ;subtract to get fractional:

                  ;         Compute 10 to fractional exponent.

DE03  A90A        EXP1      LDA     #NPCOEF
DE05  A24D                  LDX     #low P10COF
```

```
DE07  A0DE                  LDY     #high P10COF
DE09  2040DD                JSR     PLYEVL          ;P(X)
DE0C  2086DD                JSR     FMOVE
DE0F  20DBDA                JSR     FMUL            ;P(X)*P(X)

                     ;      Check integer part.

DE12  A5F1                  LDA     XFMFLG          ;integer part
DE14  F023 ^DE39            BEQ     EXP4            ;if integer part zero

                     ;      Compute 10 to integer part.

DE16  18                    CLC
DE17  6A                    ROR     A               ;integer part divided by 2
DE18  85E0                  STA     FR1             ;exponent
DE1A  A901                  LDA     #1              ;assume mantissa 1
DE1C  9002 ^DE20            BCC     EXP2            ;if integer part even

DE1E  A910                  LDA     #$10            ;substitute mantissa 10

DE20  85E1           EXP2   STA     FR1M            ;mantissa
DE22  A204                  LDX     #FMPREC-1       ;offset to last byte of man:
DE24  A900                  LDA     #0

DE26  95E2           EXP3   STA     FR1M+1,X        ;zero byte of mantissa
DE28  CA                    DEX
DE29  10FB ^DE26            BPL     EXP3            ;if not done

DE2B  A5E0                  LDA     FR1             ;exponent
DE2D  18                    CLC
DE2E  6940                  ADC     #$40            ;add bias
DE30  B019 ^DE4B            BCS     EXP6            ;if too big, error

DE32  3017 ^DE4B            BMI     EXP6            ;if underflow, error

DE34  85E0                  STA     FR1             ;10 to integer part

                     ;      Compute product of 10 to integer part and 10 to fra:

DE36  20DBDA                JSR     FMUL            ;multiply to get result

                     ;      Invert result if argument < 0.

DE39  A5F0           EXP4   LDA     SGNFLG          ;argument sign
DE3B  100D ^DE4A            BPL     EXP5            ;if argument >= 0

DE3D  2086DD                JSR     FMOVE
DE40  A28F                  LDX     #low FONE
DE42  A0DE                  LDY     #high FONE
DE44  2089DD                JSR     FLD0R           ;load FR0
DE47  2028DB                JSR     FDIV            ;divide to get result

                     ;      Exit.

DE4A  60             EXP5   RTS                     ;return

                     ;      Return error.
```

```
DE4B  38            EXP6    SEC                        ;indicate error
DE4C  60                    RTS                        ;return



                    **      P10COF - Power of 10 Coefficients


DE4D  3D17941900    P10COF  DB      $3D,$17,$94,$19,$00,$00 ;0.0000179419
DE53  3D57330500            DB      $3D,$57,$33,$05,$00,$00 ;0.0000573305
DE59  3E05547662            DB      $3E,$05,$54,$76,$62,$00 ;0.0005547662
DE5F  3E32196227            DB      $3E,$32,$19,$62,$27,$00 ;0.0032176227
DE65  3F01686030            DB      $3F,$01,$68,$60,$30,$36 ;0.0168603036
DE6B  3F07320327            DB      $3F,$07,$32,$03,$27,$41 ;0.0732032741
DE71  3F25433456            DB      $3F,$25,$43,$34,$56,$75 ;0.2543345675
DE77  3F66273730            DB      $3F,$66,$27,$37,$30,$50 ;0.6627373050
DE7D  4001151292            DB      $40,$01,$15,$12,$92,$55 ;1.15129255
DE83  3F99999999            DB      $3F,$99,$99,$99,$99,$99 ;0.9999999999

      = 000A        NPCOEF  =       [*-P10COF]/FPREC



                    **      LOG10E - Base 10 Logarithm of e


DE89  3F43429448    LOG10E  DB      $3F,$43,$42,$94,$48,$19 ;base 10 logarithm :



                    **      FONE - 1.0


DE8F  4001000000    FONE    DB      $40,$01,$00,$00,$00,$00 ;1.0



                    **      XFORM - Transform
                    *
                    *       Z = (X-C)/(X+C)
                    *
                    *       ENTRY   JSR     XFORM
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = DE95        XFORM   =       *                  ;entry
DE95  86FE                  STX     FPTR2
DE97  84FF                  STY     FPTR2+1
```

```
DE99  A2E0                     LDX     #low PLYARG
DE9B  A005                     LDY     #high PLYARG
DE9D  20A7DD                   JSR     FSTOR                  ;save argument
DEA0  A6FE                     LDX     FPTR2
DEA2  A4FF                     LDY     FPTR2+1
DEA4  2098DD                   JSR     FLD1R                  ;load FR1
DEA7  2066DA                   JSR     FADD                   ;X+C
DEAA  A2E6                     LDX     #low FPSCR
DEAC  A005                     LDY     #high FPSCR
DEAE  20A7DD                   JSR     FSTOR                  ;store FR0
DEB1  A2E0                     LDX     #low PLYARG
DEB3  A005                     LDY     #high PLYARG
DEB5  2089DD                   JSR     FLD0R                  ;load FR0
DEB8  A6FE                     LDX     FPTR2
DEBA  A4FF                     LDY     FPTR2+1
DEBC  2098DD                   JSR     FLD1R                  ;load FR1
DEBF  2060DA                   JSR     FSUB                   ;X-C
DEC2  A2E6                     LDX     #low FPSCR
DEC4  A005                     LDY     #high FPSCR
DEC6  2098DD                   JSR     FLD1R                  ;load FR1
DEC9  20280B                   JSR     FDIV                   ;divide to get result
DECC  60                       RTS                            ;return


DECD                           FIX     LOG


                      **       LOG - Compute Base e Logarithm
                      *
                      *        ENTRY   JSR     LOG
                      *                FR0 - FR0+5 = argument
                      *
                      *        MODS
                      *                Original Author Unknown
                      *                1. Bring closer to Coding Standard (object :
                      *                   R. K. Nordin 11/01/83


                      ;LOG     =       *       ;entry

DECD  A901                     LDA     #1      ;indicate base e logarithm
DECF  D002 ^DED3               BNE     LOGS    ;compute logertihm, return
```

```
DED1                         FIX     LOG10



                     **      LOG10 - Compute Base 10 Logarithm
                     *
                     *       ENTRY   JSR     LOG10
                     *               FR0 - FR0+5 = argument
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


                     ;LOG10  =       *       ;entry

DED1  A900                   LDA     #0      ;indicate base 10 logartihm
                     ;       JMP     LOGS    ;compute logarithm, return



                     **      LOGS - Compute Logarithm
                     *
                     *       ENTRY   JSR     LOGS
                     *               A = 0, if base 10 logarithm
                     *                 = 1, if base e logartihm
                     *               FR0 - FR0+5 = argument
                     *
                     *       EXIT
                     *               C set, if error
                     *               C clear, if no error
                     *               FR0 - FR0+5 = result
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = DED3         LOGS    =       *       ;entry

                     ;       Initialize.

DED3  85F0                   STA     SGNFLG  ;save logarithm base indicator

                     ;       Check argument.

DED5  A5D4                   LDA     FR0     ;argument exponent
DED7  F005 ^DEDE             BEQ     LOGS1   ;if argument zero, error

DED9  3003 ^DEDE             BMI     LOGS1   ;if argument negative, error

                     ;       X = F*(10^Y), 1<F<10
                     ;       10^Y HAS SAME EXP BYTE AS X
```

```
                        ;          & MANTISSA BYTE = 1 OR 10

DEDB  4CF6DF                       JMP      LOGQ

                        ;          Return error.

DEDE  38                LOGS1      SEC                  ;indicate error
DEDF  60                           RTS                  ;return



                        **         LOGC - Complete Computation of Logarithm
                        *
                        *          ENTRY    JSR      LOGC
                        *                   SGNFLG = 0, if base 10 logarithmr
                        *                          = 1, if base e logarithm
                        *
                        *          NOTES
                        *                   Problem: logic is convoluted because LOGQ c:
                        *                   was moved.
                        *
                        *          MODS
                        *                   Original Author Unknown
                        *                   1. Bring closer to Coding Standard (object :
                        *                      R. K. Nordin 11/01/83


        = DEE0          LOGC       =        *                   ;entry

                        ;          Initialize.

DEE0  E940                         SBC      #$40
DEE2  0A                           ASL      A
DEE3  85F1                         STA      XFMFLG              ;save Y
DEE5  A5D5                         LDA      FR0+1
DEE7  29F0                         AND      #$F0
DEE9  D004 ^DEEF                   BNE      LOGC2

DEEB  A901                         LDA      #1                  ;mantissa is 1
DEED  D004 ^DEF3                   BNE      LOGC3               ;set mantissa

DEEF  E6F1              LOGC2      INC      XFMFLG              ;increment Y
DEF1  A910                         LDA      #$10                ;mantissa is 10

DEF3  85E1              LOGC3      STA      FR1M                ;mantissa
DEF5  A204                         LDX      #FMPREC-1           ;offset to last byte of man:
DEF7  A900                         LDA      #0

DEF9  95E2              LOGC4      STA      FR1M+1,X            ;zero byte of mantissa
DEFB  CA                           DEX
DEFC  10FB ^DEF9                   BPL      LOGC4               ;if not done

DEFE  2028DB                       JSR      FDIV                ;X = X/(10^Y), S.B. IN (1,1:

                        ;          Compute LOG10(X), 1 <= X <= 10.
```

```
DF01  A266                 LDX     #low SQR10
DF03  A0DF                 LDY     #high SQR10
DF05  2095DE               JSR     XFORM          ;Z = (X-C)/(X+C); C*C = 10
DF08  A2E6                 LDX     #low FPSCR
DF0A  A005                 LDY     #high FPSCR
DF0C  20A7DD               JSR     FSTOR          ;SAVE Z
DF0F  20B6DD               JSR     FMOVE
DF12  20DBDA               JSR     FMUL           ;Z*Z
DF15  A90A                 LDA     #NLCOEF
DF17  A272                 LDX     #low LGCOEF
DF19  A0DF                 LDY     #high LGCOEF
DF1B  2040DD               JSR     PLYEVL         ;P(Z*Z)
DF1E  A2E6                 LDX     #low FPSCR
DF20  A005                 LDY     #high FPSCR
DF22  2098DD               JSR     FLD1R          ;load FR1
DF25  20DBDA               JSR     FMUL           ;Z*P(Z*Z)
DF28  A26C                 LDX     #low FHALF
DF2A  A0DF                 LDY     #high FHALF
DF2C  2098DD               JSR     FLD1R
DF2F  2066DA               JSR     FADD           ;0.5 + Z*P(Z*Z)
DF32  20B6DD               JSR     FMOVE
DF35  A900                 LDA     #0
DF37  85D5                 STA     FR0+1
DF39  A5F1                 LDA     XFMFLG
DF3B  85D4                 STA     FR0
DF3D  1007 ^DF46           BPL     LOGC5

DF3F  49FF                 EOR     #-$01          ;complement sign
DF41  18                   CLC
DF42  6901                 ADC     #1
DF44  85D4                 STA     FR0

DF46  20AAD9       LOGC5   JSR     IFP            ;convert integer to floatin
DF49  24F1                 BIT     XFMFLG
DF4B  1006 ^DF53           BPL     LOGC6

DF4D  A980                 LDA     #$80
DF4F  05D4                 ORA     FR0
DF51  85D4                 STA     FR0            ;update exponent

DF53  2066DA       LOGC6   JSR     FADD           ;LOG(X) = LOG(X)+Y

                   ;       Check base of logarithm.

DF56  A5F0                 LDA     SGNFLG         ;logarithm base indicator
DF58  F00A ^DF64           BEQ     LOGC7          ;if LOG10 (not LOG)

                   ;       Compute base e logarithm.

DF5A  A289                 LDX     #low LOG10E    ;base 10 logarithm of e
DF5C  A0DE                 LDY     #high LOG10E
DF5E  2098DD               JSR     FLD1R          ;load FR1
DF61  2028DB               JSR     FDIV           ;result is LOG(X) divided b

                   ;       Exit.

DF64  18           LOGC7   CLC                    ;indicate success
```

```
DF65  60                        RTS                          ;return



                     **         SQR10 - Square Root of 10


DF66  4003162277     SQR10      DB         $40,$03,$16,$22,$77,$66 ;square root of 10



                     **         FHALF - 0.5


DF6C  3F50000000     FHALF      DB         $3F,$50,$00,$00,$00,$00 ;0.5



                     **         LGCOEF - Logartihm Coefficients


DF72  3F49155711     LGCOEF     DB         $3F,$49,$15,$57,$11,$08 ;0.4915571108
DF78  BF51704947                DB         $BF,$51,$70,$49,$47,$08 ;-0.5170494708
DF7E  3F39205761                DB         $3F,$39,$20,$57,$61,$95 ;0.3920576195
DF84  BF04396303                DB         $BF,$04,$39,$63,$03,$55 ;-0.0439630355
DF8A  3F10093012                DB         $3F,$10,$09,$30,$12,$64 ;0.1009301264
DF90  3F09390804                DB         $3F,$09,$39,$08,$04,$60 ;0.0939080460
DF96  3F12425847                DB         $3F,$12,$42,$58,$47,$42 ;0.1242584742
DF9C  3F17371206                DB         $3F,$17,$37,$12,$06,$08 ;0.1737120608
DFA2  3F28952971                DB         $3F,$28,$95,$29,$71,$17 ;0.2895297117
DFA8  3F86858896                DB         $3F,$86,$85,$88,$96,$44 ;0.8685889644

      = 000A         NLCOEF     =          [*-LGCOEF]/FPREC



                     **         ATCOEF - Arctangent Coefficients
                     *
                     *          NOTES
                     *                     Problem: not used.


DFAE  3E16054449                DB         $3E,$16,$05,$44,$49,$00 ;0.001605444900
DFB4  BE95683845                DB         $BE,$95,$68,$38,$45,$00 ;-0.009568384500
DFBA  3F02687994                DB         $3F,$02,$68,$79,$94,$16 ;0.0268799416
DFC0  BF04927890                DB         $BF,$04,$92,$78,$90,$80 ;-0.0492789080
DFC6  3F07031520                DB         $3F,$07,$03,$15,$20,$00 ;0.0703152000
DFCC  BF08922912                DB         $BF,$08,$92,$29,$12,$44 ;-0.0892291244
DFD2  3F11084009                DB         $3F,$11,$08,$40,$09,$11 ;0.1108400911
DFD8  BF14283156                DB         $BF,$14,$28,$31,$56,$04 ;-0.1428315604
DFDE  3F19999877                DB         $3F,$19,$99,$98,$77,$44 ;0.1999987744
DFE4  BF33333331                DB         $BF,$33,$33,$33,$31,$13 ;-0.3333333113
DFEA  3F99999999                DB         $3F,$99,$99,$99,$99,$99 ;0.9999999999
```

```
DFF0  3F78539816          DB      $3F,$78,$53,$98,$16,$34 ;pi/4 = arctan 1


                    **      LOGQ - Continue Computation of Loagarithm
                    *
                    *       ENTRY   JSR     LOGQ
                    *
                    *       NOTES
                    *               Problem: logic is convoluted because this c:
                    *               moved.
                    *               Problem: for readability, this might be rel:
                    *               before tables.
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = DFF6        LOGQ    =       *       ;entry
DFF6  A5D4                  LDA     FR0
DFF8  85E0                  STA     FR1
DFFA  38                    SEC
DFFB  4CE0DE                JMP     LOGC    ;complete computation of logarithm,:
```

```
DFFE                              FIX     DCSORG



                        **        Domestic Character Set


E000  0000000000                  DB      $00,$00,$00,$00,$00,$00,$00,$00 ;$00 - spac:
E008  0018181818                  DB      $00,$18,$18,$18,$18,$00,$18,$00 ;$01 - !
E010  0066666600                  DB      $00,$66,$66,$66,$00,$00,$00,$00 ;$02 - "
E018  0066FF6666                  DB      $00,$66,$FF,$66,$66,$FF,$66,$00 ;$03 - #
E020  183E603C06                  DB      $18,$3E,$60,$3C,$06,$7C,$18,$00 ;$04 - $
E028  00666C1830                  DB      $00,$66,$6C,$18,$30,$66,$46,$00 ;$05 - %
E030  1C361C386F                  DB      $1C,$36,$1C,$38,$6F,$66,$3B,$00 ;$06 - &
E038  0018181800                  DB      $00,$18,$18,$18,$00,$00,$00,$00 ;$07 - '
E040  000E1C1818                  DB      $00,$0E,$1C,$18,$18,$1C,$0E,$00 ;$08 - (
E048  0070381818                  DB      $00,$70,$38,$18,$18,$38,$70,$00 ;$09 - )
E050  00663CFF3C                  DB      $00,$66,$3C,$FF,$3C,$66,$00,$00 ;$0A - aste:
E058  0018187E18                  DB      $00,$18,$18,$7E,$18,$18,$00,$00 ;$0B - plus
E060  0000000000                  DB      $00,$00,$00,$00,$00,$18,$18,$30 ;$0C - comm:
E068  0000007E00                  DB      $00,$00,$00,$7E,$00,$00,$00,$00 ;$0D - minu:
E070  0000000000                  DB      $00,$00,$00,$00,$00,$18,$18,$00 ;$0E - peri:
E078  00060C1830                  DB      $00,$06,$0C,$18,$30,$60,$40,$00 ;$0F - /

E080  003C666E76                  DB      $00,$3C,$66,$6E,$76,$66,$3C,$00 ;$10 - 0
E088  0018381818                  DB      $00,$18,$38,$18,$18,$18,$7E,$00 ;$11 - 1
E090  003C660C18                  DB      $00,$3C,$66,$0C,$18,$30,$7E,$00 ;$12 - 2
E098  007E0C180C                  DB      $00,$7E,$0C,$18,$0C,$66,$3C,$00 ;$13 - 3
E0A0  000C1C3C6C                  DB      $00,$0C,$1C,$3C,$6C,$7E,$0C,$00 ;$14 - 4
E0A8  007E607C06                  DB      $00,$7E,$60,$7C,$06,$66,$3C,$00 ;$15 - 5
E0B0  003C607C66                  DB      $00,$3C,$60,$7C,$66,$66,$3C,$00 ;$16 - 6
E0B8  007E060C18                  DB      $00,$7E,$06,$0C,$18,$30,$30,$00 ;$17 - 7
E0C0  003C663C66                  DB      $00,$3C,$66,$3C,$66,$66,$3C,$00 ;$18 - 8
E0C8  003C663E06                  DB      $00,$3C,$66,$3E,$06,$0C,$38,$00 ;$19 - 9
E0D0  0000181800                  DB      $00,$00,$18,$18,$00,$18,$18,$00 ;$1A - colo:
E0D8  0000181800                  DB      $00,$00,$18,$18,$00,$18,$18,$30 ;$1B - semi:
E0E0  060C183018                  DB      $06,$0C,$18,$30,$18,$0C,$06,$00 ;$1C - <
E0E8  00007E0000                  DB      $00,$00,$7E,$00,$00,$7E,$00,$00 ;$1D - =
E0F0  6030180C18                  DB      $60,$30,$18,$0C,$18,$30,$60,$00 ;$1E - >
E0F8  003C660C18                  DB      $00,$3C,$66,$0C,$18,$00,$18,$00 ;$1F - ?

E100  003C666E6E                  DB      $00,$3C,$66,$6E,$6E,$60,$3E,$00 ;$20 - @
E108  00183C6666                  DB      $00,$18,$3C,$66,$66,$7E,$66,$00 ;$21 - A
E110  007C667C66                  DB      $00,$7C,$66,$7C,$66,$66,$7C,$00 ;$22 - B
E118  003C666060                  DB      $00,$3C,$66,$60,$60,$66,$3C,$00 ;$23 - C
E120  00786C6666                  DB      $00,$78,$6C,$66,$66,$6C,$78,$00 ;$24 - D
E128  007E607C60                  DB      $00,$7E,$60,$7C,$60,$60,$7E,$00 ;$25 - E
E130  007E607C60                  DB      $00,$7E,$60,$7C,$60,$60,$60,$00 ;$26 - F
E138  003E60606E                  DB      $00,$3E,$60,$60,$6E,$66,$3E,$00 ;$27 - G
E140  0066667E66                  DB      $00,$66,$66,$7E,$66,$66,$66,$00 ;$28 - H
E148  007E181818                  DB      $00,$7E,$18,$18,$18,$18,$7E,$00 ;$29 - I
E150  0006060606                  DB      $00,$06,$06,$06,$06,$66,$3C,$00 ;$2A - J
E158  00666C7878                  DB      $00,$66,$6C,$78,$78,$6C,$66,$00 ;$2B - K
E160  0060606060                  DB      $00,$60,$60,$60,$60,$60,$7E,$00 ;$2C - L
E168  0063777F6B                  DB      $00,$63,$77,$7F,$6B,$63,$63,$00 ;$2D - M
E170  0066767E7E                  DB      $00,$66,$76,$7E,$7E,$6E,$66,$00 ;$2E - N
```

```
E178  003C666666          DB        $00,$3C,$66,$66,$66,$66,$3C,$00 ;$2F - O

E180  007C66667C          DB        $00,$7C,$66,$66,$7C,$60,$60,$00 ;$30 - P
E188  003C666666          DB        $00,$3C,$66,$66,$66,$6C,$36,$00 ;$31 - Q
E190  007C66667C          DB        $00,$7C,$66,$66,$7C,$6C,$66,$00 ;$32 - R
E198  003C603C06          DB        $00,$3C,$60,$3C,$06,$06,$3C,$00 ;$33 - S
E1A0  007E181818          DB        $00,$7E,$18,$18,$18,$18,$18,$00 ;$34 - T
E1A8  0066666666          DB        $00,$66,$66,$66,$66,$66,$7E,$00 ;$35 - U
E1B0  0066666666          DB        $00,$66,$66,$66,$66,$3C,$18,$00 ;$36 - V
E1B8  0063636B7F          DB        $00,$63,$63,$6B,$7F,$77,$63,$00 ;$37 - W
E1C0  0066663C3C          DB        $00,$66,$66,$3C,$3C,$66,$66,$00 ;$38 - X
E1C8  0066663C18          DB        $00,$66,$66,$3C,$18,$18,$18,$00 ;$39 - Y
E1D0  007E0C1830          DB        $00,$7E,$0C,$18,$30,$60,$7E,$00 ;$3A - Z
E1D8  001E181818          DB        $00,$1E,$18,$18,$18,$18,$1E,$00 ;$3B - [
E1E0  0040603018          DB        $00,$40,$60,$30,$18,$0C,$06,$00 ;$3C - \
E1E8  0078181818          DB        $00,$78,$18,$18,$18,$18,$78,$00 ;$3D - ]
E1F0  00081C3663          DB        $00,$08,$1C,$36,$63,$00,$00,$00 ;$3E - ^
E1F8  0000000000          DB        $00,$00,$00,$00,$00,$00,$FF,$00 ;$3F - unde:

E200  00367F7F3E          DB        $00,$36,$7F,$7F,$3E,$1C,$08,$00 ;$40 - hear:
E208  1818181F1F          DB        $18,$18,$18,$1F,$1F,$18,$18,$18 ;$41 - mid :
E210  0303030303          DB        $03,$03,$03,$03,$03,$03,$03,$03 ;$42 - righ:
E218  181818F8F8          DB        $18,$18,$18,$F8,$F8,$00,$00,$00 ;$43 - low :
E220  181818F8F8          DB        $18,$18,$18,$F8,$F8,$18,$18,$18 ;$44 - mid :
E228  000000F8F8          DB        $00,$00,$00,$F8,$F8,$18,$18,$18 ;$45 - up r:
E230  03070E1C38          DB        $03,$07,$0E,$1C,$38,$70,$E0,$C0 ;$46 - righ:
E238  C0E070381C          DB        $C0,$E0,$70,$38,$1C,$0E,$07,$03 ;$47 - left:
E240  0103070F1F          DB        $01,$03,$07,$0F,$1F,$3F,$7F,$FF ;$48 - righ:
E248  000000000F          DB        $00,$00,$00,$00,$0F,$0F,$0F,$0F ;$49 - low :
E250  80C0E0F0F8          DB        $80,$C0,$E0,$F0,$F8,$FC,$FE,$FF ;$4A - left:
E258  0F0F0F0F00          DB        $0F,$0F,$0F,$0F,$00,$00,$00,$00 ;$4B - up r:
E260  F0F0F0F000          DB        $F0,$F0,$F0,$F0,$00,$00,$00,$00 ;$4C - up l:
E268  FFFF000000          DB        $FF,$FF,$00,$00,$00,$00,$00,$00 ;$4D - top :
E270  0000000000          DB        $00,$00,$00,$00,$00,$00,$FF,$FF ;$4E - bott:
E278  00000000F0          DB        $00,$00,$00,$00,$F0,$F0,$F0,$F0 ;$4F - low :

E280  001C1C7777          DB        $00,$1C,$1C,$77,$77,$08,$1C,$00 ;$50 - club:
E288  0000001F1F          DB        $00,$00,$00,$1F,$1F,$18,$18,$18 ;$51 - up l:
E290  000000FFFF          DB        $00,$00,$00,$FF,$FF,$00,$00,$00 ;$52 - mid :
E298  181818FFFF          DB        $18,$18,$18,$FF,$FF,$18,$18,$18 ;$53 - mid :
E2A0  00003C7E7E          DB        $00,$00,$3C,$7E,$7E,$7E,$3C,$00 ;$54 - soli:
E2A8  00000000FF          DB        $00,$00,$00,$00,$FF,$FF,$FF,$FF ;$55 - bott:
E2B0  C0C0C0C0C0          DB        $C0,$C0,$C0,$C0,$C0,$C0,$C0,$C0 ;$56 - left:
E2B8  000000FFFF          DB        $00,$00,$00,$FF,$FF,$18,$18,$18 ;$57 - up m:
E2C0  181818FFFF          DB        $18,$18,$18,$FF,$FF,$00,$00,$00 ;$58 - low :
E2C8  F0F0F0F0F0          DB        $F0,$F0,$F0,$F0,$F0,$F0,$F0,$F0 ;$59 - left:
E2D0  1818181F1F          DB        $18,$18,$18,$1F,$1F,$00,$00,$00 ;$5A - low :
E2D8  786078607E          DB        $78,$60,$78,$60,$7E,$18,$1E,$00 ;$5B - disp:
E2E0  00183C7E18          DB        $00,$18,$3C,$7E,$18,$18,$18,$00 ;$5C - up a:
E2E8  001818187E          DB        $00,$18,$18,$18,$7E,$3C,$18,$00 ;$5D - down:
E2F0  0018307E30          DB        $00,$18,$30,$7E,$30,$18,$00,$00 ;$5E - left:
E2F8  00180C7E0C          DB        $00,$18,$0C,$7E,$0C,$18,$00,$00 ;$5F - righ:

E300  00183C7E7E          DB        $00,$18,$3C,$7E,$7E,$3C,$18,$00 ;$60 - diam:
E308  00003C063E          DB        $00,$00,$3C,$06,$3E,$66,$3E,$00 ;$61 - a
E310  0060607C66          DB        $00,$60,$60,$7C,$66,$66,$7C,$00 ;$62 - b
E318  00003C6060          DB        $00,$00,$3C,$60,$60,$60,$3C,$00 ;$63 - c
```

```
E320  0006063E66       DB      $00,$06,$06,$3E,$66,$66,$3E,$00 ;$64 - d
E328  00003C667E       DB      $00,$00,$3C,$66,$7E,$60,$3C,$00 ;$65 - e
E330  000E183E18       DB      $00,$0E,$18,$3E,$18,$18,$18,$00 ;$66 - f
E338  00003E6666       DB      $00,$00,$3E,$66,$66,$3E,$06,$7C ;$67 - g
E340  0060607C66       DB      $00,$60,$60,$7C,$66,$66,$66,$00 ;$68 - h
E348  0018003818       DB      $00,$18,$00,$38,$18,$18,$3C,$00 ;$69 - i
E350  0006000606       DB      $00,$06,$00,$06,$06,$06,$06,$3C ;$6A - j
E358  0060606C78       DB      $00,$60,$60,$6C,$78,$6C,$66,$00 ;$6B - k
E360  0038181818       DB      $00,$38,$18,$18,$18,$18,$3C,$00 ;$6C - l
E368  0000667F7F       DB      $00,$00,$66,$7F,$7F,$6B,$63,$00 ;$6D - m
E370  00007C6666       DB      $00,$00,$7C,$66,$66,$66,$66,$00 ;$6E - n
E378  00003C6666       DB      $00,$00,$3C,$66,$66,$66,$3C,$00 ;$6F - o

E380  00007C6666       DB      $00,$00,$7C,$66,$66,$7C,$60,$60 ;$70 - p
E388  00003E6666       DB      $00,$00,$3E,$66,$66,$3E,$06,$06 ;$71 - q
E390  00007C6660       DB      $00,$00,$7C,$66,$60,$60,$60,$00 ;$72 - r
E398  00003E603C       DB      $00,$00,$3E,$60,$3C,$06,$7C,$00 ;$73 - s
E3A0  00187E1818       DB      $00,$18,$7E,$18,$18,$18,$0E,$00 ;$74 - t
E3A8  0000666666       DB      $00,$00,$66,$66,$66,$66,$3E,$00 ;$75 - u
E3B0  0000666666       DB      $00,$00,$66,$66,$66,$3C,$18,$00 ;$76 - v
E3B8  0000636B7F       DB      $00,$00,$63,$6B,$7F,$3E,$36,$00 ;$77 - w
E3C0  0000663C18       DB      $00,$00,$66,$3C,$18,$3C,$66,$00 ;$78 - x
E3C8  0000666666       DB      $00,$00,$66,$66,$66,$3E,$0C,$78 ;$79 - y
E3D0  00007E0C18       DB      $00,$00,$7E,$0C,$18,$30,$7E,$00 ;$7A - z
E3D8  00183C7E7E       DB      $00,$18,$3C,$7E,$7E,$18,$3C,$00 ;$7B - spad:
E3E0  1818181818       DB      $18,$18,$18,$18,$18,$18,$18,$18 ;$7C - |
E3E8  007E787C6E       DB      $00,$7E,$78,$7C,$6E,$66,$06,$00 ;$7D - disp:
E3F0  0818387838       DB      $08,$18,$38,$78,$38,$18,$08,$00 ;$7E - disp:
E3F8  10181C1E1C       DB      $10,$18,$1C,$1E,$1C,$18,$10,$00 ;$7F - disp:
```

```
E400                          FIX     EDITRV



                    **        EDITRV - Editor Handler Vector Table


E400  93EF                    DW      EOP-1     ;perform editor OPEN
E402  2DF2                    DW      ECL-1     ;perform editor CLOSE
E404  49F2                    DW      EGB-1     ;perform editor GET-BYTE
E406  AFF2                    DW      EPB-1     ;perform editor PUT-BYTE
E408  1DF2                    DW      SST-1     ;perform editor STATUS (screen STAT:
E40A  2CF2                    DW      ESP-1     ;perform editor SPECIAL
E40C  4C6EEF                  JMP     SIN       ;initialize editor (initialize scre:
E40F  00                      DB      0         ;reserved



E410                          FIX     SCRENV



                    **        SCRENV - Screen Handler Vector Table


E410  8DEF                    DW      SOP-1     ;perform screen OPEN
E412  2DF2                    DW      ECL-1     ;perform screen CLOSE (editor CLOSE:
E414  7FF1                    DW      SGB-1     ;perform screen GET-BYTE
E416  A3F1                    DW      SPB-1     ;perform screen PUT-BYTE
E418  1DF2                    DW      SST-1     ;perform screen STATUS
E41A  AEF9                    DW      SSP-1     ;perform screen SPECIAL
E41C  4C6EEF                  JMP     SIN       ;initialize screen
E41F  00                      DB      0         ;reserved



E420                          FIX     KEYBDV



                    **        KEYBDV - Keyboard Handler Vector Table


E420  1DF2                    DW      SST-1     ;perform keyboard OPEN (screen STAT:
E422  1DF2                    DW      SST-1     ;perform keyboard CLOSE (screen STA:
E424  FCF2                    DW      KGB-1     ;perform keyboard GET-BYTE
E426  2CF2                    DW      ESP-1     ;perform keyboard SPECIAL (editor S:
E428  1DF2                    DW      SST-1     ;perform keyboard STATUS (screen ST:
E42A  2CF2                    DW      ESP-1     ;perform keyboard SPECIAL (editor S:
E42C  4C6EEF                  JMP     SIN       ;initialize keyboard (initialize sc:
E42F  00                      DB      0         ;reserved
```

```
E430                        FIX     PRINTV



                   **       PRINTV - Printer Handler Vector Table


E430  C1FE                  DW      POP-1    ;perform printer OPEN
E432  06FF                  DW      PCL-1    ;perform printer CLOSE
E434  C0FE                  DW      PSP-1    ;perform printer SPECIAL
E436  CAFE                  DW      PPB-1    ;perform printer PUT-BYTE
E438  A2FE                  DW      PST-1    ;perform printer STATUS
E43A  C0FE                  DW      PSP-1    ;perform printer SPECIAL
E43C  4C99FE                JMP     PIN      ;initialize printer
E43F  00                    DB      0        ;reserved



E440                        FIX     CASETV



                   **       CASETV - Cassette Handler Vector Table


E440  E5FC                  DW      COP-1    ;perform cassette OPEN
E442  CEFD                  DW      CCL-1    ;perform cassette CLOSE
E444  79FD                  DW      CGB-1    ;perform cassette GET-BYTE
E446  B3FD                  DW      CPB-1    ;perform cassette PUT-BYTE
E448  C8FD                  DW      CST-1    ;perform cassette STATUS
E44A  E4FC                  DW      CSP-1    ;perform cassette SPECIAL
E44C  4CDBFC                JMP     CIN      ;initialize cassette
E44F  00                    DB      0        ;reserved
```

```
E450                  **       Jump Vectors


E450                           FIX      DINITV
E450   4CA3C6                  JMP      IDIO      ;initialize DIO

E453                           FIX      DSKINV
E453   4CB3C6                  JMP      DIO       ;perform DIO

E456                           FIX      CIOV
E456   4CDFE4                  JMP      CIO       ;perform CIO

E459                           FIX      SIOV
E459   4C33C9                  JMP      PIO       ;perform PIO

E45C                           FIX      SETVBV
E45C   4C72C2                  JMP      SVP       ;set VBLANK parameters

E45F                           FIX      SYSVBV
E45F   4CE2C0                  JMP      IVNM      ;process immediate VBLANK NMI

E462                           FIX      XITVBV
E462   4C8AC2                  JMP      DVNM      ;process deferred VBLANK NMI

E465                           FIX      SIOINV
E465   4C5CE9                  JMP      ISIO      ;initialize SIO

E468                           FIX      SENDEV
E468   4C17EC                  JMP      ESS       ;enable SIO SEND

E46B                           FIX      INTINV
E46B   4C0CC0                  JMP      IIH       ;initialize interrupt handler

E46E                           FIX      CIOINV
E46E   4CC1E4                  JMP      ICIO      ;initialize CIO

E471                           FIX      BLKBDV
E471   4C23F2                  JMP      PPD       ;perform power-up display

E474                           FIX      WARMSV
E474   4C90C2                  JMP      PWS       ;perform warmstart

E477                           FIX      COLDSV
E477   4CC8C2                  JMP      PCS       ;perform coldstart

E47A                           FIX      RBLOKV
E47A   4C8DFD                  JMP      RCB       ;read cassette block

E47D                           FIX      CSOPIV
E47D   4CF7FC                  JMP      OCI       ;open cassette for input

E480                           FIX      PUPDIV
E480   4C23F2                  JMP      PPD       ;perform power-up display

E483                           FIX      SLFTSV
E483   4C0050                  JMP      STH       ;self-test hardware
```

```
E486                         FIX     PHENTV
E486   4C8CEE                JMP     PHE       ;perform peripheral handler entry

E489                         FIX     PHUNLV
E489   4C15E9                JMP     PHU       ;perform peripheral handler unlink:

E48C                         FIX     PHINIV
E48C   4C98E6                JMP     PHI       ;perform peripheral handler initial:
```

```
E48F                        FIX     GPDVV



                    **      GPDVV - Generic Parallel Device Handler Vector Tabl:


E48F  90C9                  DW      GOP-1    ;perform generic parallel device OP:
E491  95C9                  DW      GCL-1    ;perform generic parallel device CL:
E493  9AC9                  DW      GGB-1    ;perform generic parallel device GE:
E495  9FC9                  DW      GPB-1    ;perform generic parallel device PU:
E497  A4C9                  DW      GST-1    ;perform generic parallel device ST:
E499  A9C9                  DW      GSP-1    ;perform generic parallel device SP:
E49B  4C0CC9                JMP     GIN      ;initialize generic parallel device
```

```
E49E                          FIX     $E4C0



                       **     E4C0 - $E4C0 Patch
                       *
                       *      For compatibility with OS Revision B, return.

E4C0  60                      RTS                 ;return
```

```
E4C1                **      ICIO - Initialize CIO
                    *
                    *       ENTRY   JSR     ICIO
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = E4C1         ICIO    =       *                 ;entry

                    ;       Initialize IOCB's.

E4C1  A200                  LDX     #0                ;index of first IOCB

E4C3  A9FF          ICIO1   LDA     #IOCFRE           ;IOCB free indicator
E4C5  9D4003                STA     ICHID,X           ;set IOCB free
E4C8  A9DB                  LDA     #low [IIN-1]
E4CA  9D4603                STA     ICPTL,X           ;initialize PUT-BYTE routin:
E4CD  A9E4                  LDA     #high [IIN-1]
E4CF  9D4703                STA     ICPTH,X
E4D2  8A                    TXA                       ;index of current IOCB
E4D3  18                    CLC
E4D4  6910                  ADC     #IOCBSZ           ;add IOCB size
E4D6  AA                    TAX                       ;index of next IOCB
E4D7  C980                  CMP     #MAXIOC           ;index of first invalid IOC:
E4D9  90E8 ^E4C3            BCC     ICIO1             ;if not done

E4DB  60                    RTS                       ;return



                    **      IIN - Indicate IOCB Not Open Error
                    *
                    *       ENTRY   JSR     IIN
                    *
                    *       EXIT
                    *               Y = IOCB Not Open error code
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = E4DC         IIN     =       *       ;entry
E4DC  A085                  LDY     #NOTOPN ;IOCB not open error
E4DE  60                    RTS             ;return
```

```
                        **      CIO - Central Input/Output
                        *
                        *       ENTRY   JSR     CIO
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = E4DF          CIO     =       *       ;entry

                        ;       Initialize.

E4DF  852F                      STA     CIOCHR  ;save possible output byte value
E4E1  862E                      STX     ICIDNO  ;save IOCB index

                        ;       Check IOCB index validity.

E4E3  8A                        TXA             ;IOCB index
E4E4  290F                      AND     #$0F    ;index modulo 16
E4E6  D004 ^E4EC                BNE     CIO1    ;if IOCB not multiple of 16, error

E4E8  E080                      CPX     #MAXIOC ;index of first invalid IOCB
E4EA  9005 ^E4F1                BCC     CIO2    ;if index within range

                        ;       Indicate Invalid IOCB Index error.

E4EC  A086              CIO1    LDY     #BADIOC ;invalid IOCB index error
E4EE  4C70E6                    JMP     SSC     ;set status and complete operation,:

                        ;       Move part of IOCB to zero page IOCB.

E4F1  A000              CIO2    LDY     #0              ;offset to first byte of pa:

E4F3  BD4003            CIO3    LDA     IOCB,X          ;byte of IOCB
E4F6  992000                    STA     IOCBAS,Y        ;byte of zero page IOCB
E4F9  E8                        INX
E4FA  C8                        INY
E4FB  C00C                      CPY     #ICSPRZ-IOCBAS  ;offset to first undesired :
E4FD  90F4 ^E4F3                BCC     CIO3            ;if not done

                        ;       Check for provisionally open IOCB.

E4FF  A520                      LDA     ICHIDZ  ;handler ID
E501  C97F                      CMP     #$7F    ;provisionally open indicator
E503  D015 ^E51A                BNE     PCC     ;if not provisionally open, perform:

                        ;       Check for CLOSE command.

E505  A522                      LDA     ICCOMZ  ;command
E507  C90C                      CMP     #CLOSE
E509  F071 ^E57C                BEQ     XCL     ;if CLOSE command

                        ;       Check handler load flag.

E50B  ADE902                    LDA     HNDLOD
```

```
E50E  D005 ^E515              BNE     LHO     ;if handler load desired

                        ;     Indicate nonexistent device error.

                        ;     JMP     IND     ;indicate nonexistent device error,:



                        **    IND - Indicate Nonexistent Device Error
                        *
                        *     ENTRY   JSR     IND
                        *
                        *     MODS
                        *             Original Author Unknown
                        *             1. Bring closer to Coding Standard (object :
                        *                R. K. Nordin 11/01/83

      = E510            IND   =       *       ;entry
E510  A082                    LDY     #NONDEV ;nonexistent device error

E512  4C70E6            IND1  JMP     SSC     ;set status and complete operation,:



                        **    LHO - Load Peripheral Handler for OPEN
                        *
                        *     ENTRY   JSR     LHO
                        *
                        *     MODS
                        *             Original Author Unknown
                        *             1. Bring closer to Coding Standard (object :
                        *                R. K. Nordin 11/01/83

      = E515            LHO   =       *       ;entry
E515  2029CA                  JSR     PHL     ;load and initialize peripheral han:
E518  30F8 ^E512              BMI     IND1    ;if error

                        ;     JMP     PCC     ;perform CIO command, return
                        **      PCC - Perform CIO Command
                        *
                        *       ENTRY   JSR     PCC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E51A          PCC     =       *       ;entry

                        ;       Check command validity.

E51A  A084                      LDY     #NVALID         ;assume invalid code
E51C  A522                      LDA     ICCOMZ          ;command
E51E  C903                      CMP     #OPEN           ;first valid command
E520  9025 ^E547                BCC     XOP1            ;if command invalid

E522  A8                        TAY                     ;command

E523  C00E                      CPY     #SPECIL         ;last valid command
E525  9002 ^E529                BCC     PCC1            ;if valid

E527  A00E                      LDY     #SPECIL         ;substitute SPECIAL command

                        ;       Obtain vector offset.

E529  8417              PCC1    STY     ICCOMT          ;save command
E52B  B92AE7                    LDA     TCVO-3,Y        ;vector offset for command
E52E  F00F ^E53F                BEQ     XOP             ;if OPEN command, process

                        ;       Perform command.

E530  C902                      CMP     #2
E532  F048 ^E57C                BEQ     XCL             ;if CLOSE command, process

E534  C908                      CMP     #8
E536  B05F ^E597                BCS     XSS             ;if STATUS or SPECIAL comma:

E538  C904                      CMP     #4
E53A  F076 ^E5B2                BEQ     XGT             ;if GET command, process

E53C  4C1EE6                    JMP     XPT             ;process PUT command, proce:
```

```
                    **      XOP - Execute OPEN Command
                    *
                    *       ENTRY   JSR     XOP
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = E53F         XOP     =       *       ;entry

                    ;       Check IOCB free.

E53F  A520                  LDA     ICHIDZ  ;handler ID
E541  C9FF                  CMP     #IOCFRE ;IOCB free indicator
E543  F005 ^E54A            BEQ     XOP2    ;if IOCB free

                    ;       Process error.

E545  A081                  LDY     #PRVOPN ;IOCB previously open error

E547  4C70E6        XOP1    JMP     SSC     ;set status and complete operation,:

                    ;       Check handler load.

E54A  ADE902        XOP2    LDA     HNDLOD
E54D  D027 ^E576            BNE     PPO     ;if user wants unconditional poll

                    ;       Search handler table.

E54F  20FFE6                JSR     SHT     ;search handler table
E552  B022 ^E576            BCS     PPO     ;if not found, poll

                    ;       Initialize status.

E554  A900                  LDA     #0
E556  8DEA02                STA     DVSTAT  ;clear status
E559  8DEB02                STA     DVSTAT+1

                    ;       Initialize IOCB.

                    ;       JMP     IIO     ;initialize IOCB for OPEN, return
```

```
                        **      IIO - Initialize IOCB for OPEN
                        *
                        *       ENTRY   JSR     IIO
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = E55C         IIO     =       *       ;entry

                        ;       Compute handler entry point.

E55C  2095E6                    JSR     CEP     ;compute handler entry point
E55F  B0E6 ^E547                BCS     XOP1    ;if error

                        ;       Execute command.

E561  20EAE6                    JSR     EHC     ;execute handler command

                        ;       Set PUT-BYTE routine address in IOCB.

E564  A90B                      LDA     #PUTCHR
E566  8517                      STA     ICCOMT  ;command
E568  2095E6                    JSR     CEP     ;compute handler entry point
E56B  A52C                      LDA     ICSPRZ  ;PUT-BYTE routine address
E56D  8526                      STA     ICPTLZ  ;IOCB PUT-BYTE routine address
E56F  A52D                      LDA     ICSPRZ+1
E571  8527                      STA     ICPTHZ
E573  4C72E6                    JMP     CCO     ;complete CIO operation, return



                        **      PPO - Poll Peripheral for OPEN
                        *
                        *       ENTRY   JSR     PPO
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

         = E576         PPO     =       *       ;entry
E576  20F9EE                    JSR     PHO     ;poll
E579  4C70E6                    JMP     SSC     ;set status and complete operation,:
```

```
                        **      XCL - Execute CLOSE Command
                        *
                        *       ENTRY   JSR     XCL
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = E57C     XCL     =       *               ;entry

                        ;       Initialize.

E57C  A001                      LDY     #SUCCES         ;assume success
E57E  8423                      STY     ICSTAZ          ;status
E580  2095E6                    JSR     CEP             ;compute handler entry poin:
E583  B003 ^E588                BCS     XCL1            ;if error

                        ;       Execute command.

E585  20EAE6                    JSR     EHC             ;execute handler command

                        ;       Close IOCB.

E588  A9FF              XCL1    LDA     #IOCFRE         ;IOCB free indicator
E58A  8520                      STA     ICHIDZ          ;indicate IOCB free
E58C  A9E4                      LDA     #high [IIN-1]
E58E  8527                      STA     ICPTHZ          ;reset initial PUT-BYTE rou:
E590  A9DB                      LDA     #low [IIN-1]
E592  8526                      STA     ICPTLZ
E594  4C72E6                    JMP     CCO             ;complete CIO operation, re:



                        **      XSS - Execute STATUS and SPECIAL Commands
                        *
                        *       ENTRY   JSR     XSS
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = E597     XSS     =       *       ;entry

                        ;       Check IOCB free.

E597  A520                      LDA     ICHIDZ  ;handler ID
E599  C9FF                      CMP     #IOCFRE
E59B  D005 ^E5A2                BNE     XSS1    ;if IOCB not free

                        ;       Open IOCB.

E59D  20FFE6                    JSR     SHT     ;search handler table
```

```
E5A0  B0A5 ^E547              BCS     XOP1     ;if error

                        ;     Execute command.

E5A2  2095E6        XSS1      JSR     CEP      ;compute handler entry point
E5A5  20EAE6                  JSR     EHC      ;execute handler command

                        ;     Restore handler ID, in case IOCB implicitly opened.

E5A8  A62E                    LDX     ICIDNO   ;IOCB index
E5AA  BD4003                  LDA     ICHID,X  ;original handler ID
E5AD  8520                    STA     ICHIDZ   ;restore zero page handler ID
E5AF  4C72E6                  JMP     CCO      ;complete CIO operation, return



                        **    XGT - Execute GET Command
                        *
                        *     ENTRY   JSR      XGT
                        *
                        *     MODS
                        *             Original Author Unknown
                        *             1. Bring closer to Coding Standard (object :
                        *                R. K. Nordin 11/01/83

      = E5B2            XGT   =       *        ;entry

                        ;     Check GET validity.

E5B2  A522                    LDA     ICCOMZ   ;command
E5B4  252A                    AND     ICAX1Z
E5B6  D005 ^E5BD              BNE     XGT2     ;if GET command valid

                        ;     Process error.

E5B8  A083                    LDY     #WRONLY  ;IOCB opened for write only error

E5BA  4C70E6        XGT1      JMP     SSC      ;set status and complete operation,:

                        ;     Compute and check handler entry point.

E5BD  2095E6        XGT2      JSR     CEP      ;compute handler entry point
E5C0  B0F8 ^E5BA              BCS     XGT1     ;if error

                        ;     Check buffer length.

E5C2  A528                    LDA     ICBLLZ            ;buffer length
E5C4  0529                    ORA     ICBLLZ+1
E5C6  D008 ^E5D0              BNE     XGT3              ;if buffer length non-zero

                        ;     Get byte.

E5C8  20EAE6                  JSR     EHC      ;execute handler command
E5CB  852F                    STA     CIOCHR   ;data
E5CD  4C72E6                  JMP     CCO      ;complete CIO operation, return
```

```
                        ;       Fill buffer.

E5D0  20EAE6    XGT3    JSR     EHC             ;execute handler command
E5D3  852F              STA     CIOCHR          ;data
E5D5  3041 ^E618        BMI     XGT7            ;if error, end transfer

E5D7  A000              LDY     #0
E5D9  9124              STA     (ICBALZ),Y      ;byte of buffer
E5DB  20D1E6            JSR     IBP             ;increment buffer pointer
E5DE  A522              LDA     ICCOMZ          ;command
E5E0  2902              AND     #$02
E5E2  D00C ^E5F0        BNE     XGT4            ;if GET RECORD command

                        ;       Check for EOL.

E5E4  A52F              LDA     CIOCHR  ;data
E5E6  C99B              CMP     #EOL
E5E8  D006 ^E5F0        BNE     XGT4    ;if not EOL

                        ;       Process EOL.

E5EA  20BBE6            JSR     DBL     ;decrement buffer length
E5ED  4C18E6            JMP     XGT7    ;clean up

                        ;       Check buffer full.

E5F0  20BBE6    XGT4    JSR     DBL     ;decrement buffer length
E5F3  D0DB ^E5D0        BNE     XGT3    ;if buffer not full, continue

                        ;       Check command.

E5F5  A522              LDA     ICCOMZ  ;command
E5F7  2902              AND     #$02
E5F9  D01D ^E618        BNE     XGT7    ;if GET CHARACTER command, clean up

                        ;       Process GET RECORD.

E5FB  20EAE6    XGT5    JSR     EHC     ;execute handler command
E5FE  852F              STA     CIOCHR  ;data
E600  300A ^E60C        BMI     XGT6    ;if error

                        ;       Check for EOL.

E602  A52F              LDA     CIOCHR  ;data
E604  C99B              CMP     #EOL
E606  D0F3 ^E5FB        BNE     XGT5    ;if not EOL, continue

                        ;       Process end of record.

E608  A989              LDA     #TRNRCD ;truncated record error
E60A  8523              STA     ICSTAZ  ;status

                        ;       Process error.

E60C  20C8E6    XGT6    JSR     DBP             ;decrement buffer pointer
E60F  A000              LDY     #0
```

```
E611  A998                LDA    #EOL
E613  9124                STA    (ICBALZ),Y       ;set EOL in buffer
E615  20D1E6              JSR    IBP              ;increment buffer pointer

                   ;      Clean up.

E618  20D8E6       XGT7   JSR    SFL      ;set final buffer length
E61B  4C72E6              JMP    CCO      ;complete CIO operation, return



                   **     XPT - Execute PUT Command
                   *
                   *      ENTRY   JSR     XPT
                   *
                   *      MODS
                   *              Original Author Unknown
                   *              1. Bring closer to Coding Standard (object :
                   *                 R. K. Nordin 11/01/83


      = E61E       XPT    =       *       ;entry

                   ;      Check PUT validity.

E61E  A522                LDA    ICCOMZ  ;command
E620  252A                AND    ICAX1Z
E622  D005 ^E629          BNE    XPT2    ;if PUT command valid

                   ;      Process error.

E624  A087                LDY    #RDONLY ;IOCB opened for read only error

E626  4C70E6       XPT1   JMP    SSC      ;set status and complete operation,:

                   ;      Compute and check handler entry point.

E629  2095E6       XPT2   JSR    CEP      ;compute handler entry point
E62C  B0F8 ^E626          BCS    XPT1     ;if error

                   ;      Check buffer length.

E62E  A528                LDA    ICBLLZ  ;buffer length
E630  0529                ORA    ICBLLZ+1
E632  D006 ^E63A          BNE    XPT3     ;if buffer length non-zero

                   ;      Put byte.

E634  A52F                LDA    CIOCHR  ;data
E636  E628                INC    ICBLLZ  ;set buffer length to 1
E638  D006 ^E640          BNE    XPT4    ;transfer one byte

                   ;      Transfer data from buffer to handler.

E63A  A000         XPT3   LDY    #0
E63C  B124                LDA    (ICBALZ),Y       ;byte from buffer
```

```
E63E  852F                STA     CIOCHR          ;data

E640  20EAE6      XPT4    JSR     EHC             ;execute handler command
E643  08                  PHP                     ;save status
E644  20D1E6              JSR     IBP             ;increment buffer pointer
E647  20BBE6              JSR     DBL             ;decrement buffer length
E64A  28                  PLP                     ;status
E64B  301D ^E66A          BMI     XPT6            ;if error

                  ;       Check command.

E64D  A522                LDA     ICCOMZ  ;command
E64F  2902                AND     #$02
E651  D006 ^E659          BNE     XPT5    ;if PUT RECORD command

                  ;       Check for EOL.

E653  A52F                LDA     CIOCHR  ;data
E655  C99B                CMP     #EOL
E657  F011 ^E66A          BEQ     XPT6    ;if EOL, clean up

                  ;       Check for buffer empty.

E659  A528        XPT5    LDA     ICBLLZ          ;buffer length
E65B  0529                ORA     ICBLLZ+1
E65D  D0DB ^E63A          BNE     XPT3            ;if buffer not empty, conti:

                  ;       Check command.

E65F  A522                LDA     ICCOMZ  ;command
E661  2902                AND     #$02
E663  D005 ^E66A          BNE     XPT6    ;if PUT CHARACTER command

                  ;       Write EOL.

E665  A99B                LDA     #EOL
E667  20EAE6              JSR     EHC     ;execute handler command

                  ;       Clean up.

E66A  20D8E6      XPT6    JSR     SFL     ;set final buffer length
E66D  4C72E6              JMP     CCO     ;complete CIO operation, return
```

```
                        **      SSC - Set Status and Complete Operation
                        *
                        *       ENTRY   JSR     SSC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E670          SSC     =       *       ;entry
E670    8423                    STY     ICSTAZ  ;status
                        ;       JMP     CCO     ;complete CIO operation, return



                        **      CCO - Complete CIO Operation
                        *
                        *       ENTRY   JSR     CCO
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = E672          CCO     =       *                       ;entry

                        ;       Initialize.

E672    A42E                    LDY     ICIDNO                  ;IOCB index

                        ;       Restore buffer pointer.

E674    B94403                  LDA     ICBAL,Y
E677    8524                    STA     ICBALZ                  ;restore buffer pointer
E679    B94503                  LDA     ICBAH,Y
E67C    8525                    STA     ICBAHZ

                        ;       Move part of zero page IOCB to IOCB.

E67E    A200                    LDX     #0                      ;first byte of zero page IO:
E680    8EE902                  STX     HNDLOD

E683    B520            CCO1    LDA     IOCBAS,X                ;byte of zero page IOCB
E685    994003                  STA     IOCB,Y                  ;byte of IOCB
E688    E8                      INX
E689    C8                      INY
E68A    E00C                    CPX     #ICSPRZ-IOCBAS          ;offset to first undesired :
E68C    90F5 ^E683              BCC     CCO1                    ;if not done

                        ;       Restore A, X and Y.

E68E    A52F                    LDA     CIOCHR                  ;data
E690    A62E                    LDX     ICIDNO                  ;IOCB index
E692    A423                    LDY     ICSTAZ                  ;status
```

```
E694  60                    RTS                        ;return



                      **      CEP - Compute Handler Entry Point
                      *
                      *       ENTRY    JSR     CEP
                      *
                      *       MODS
                      *                Original Author Unknown
                      *                1. Bring closer to Coding Standard (object :
                      *                   R. K. Nordin 11/01/83


      = E695          CEP     =        *               ;entry

                      ;       Check handler ID validity.

E695  A420                    LDY      ICHIDZ          ;handler ID
E697  C022                    CPY      #MAXDEV+1       ;first invalid ID
E699  9004 ^E69F              BCC      CEP1            ;if handler ID within range

                      ;       Process error.

E69B  A085                    LDY      #NOTOPN         ;IOCB not open error
E69D  B01B ^E6BA              BCS      CEP2            ;return

                      ;       Compute entry point.

E69F  B91B03          CEP1    LDA      HATABS+1,Y      ;low address
E6A2  852C                    STA      ICSPRZ
E6A4  B91C03                  LDA      HATABS+2,Y      ;high address
E6A7  852D                    STA      ICSPRZ+1
E6A9  A417                    LDY      ICCOMT          ;command
E6AB  B92AE7                  LDA      TCVO-3,Y        ;vector offset for command
E6AE  A8                      TAY
E6AF  B12C                    LDA      (ICSPRZ),Y      ;low vector address
E6B1  AA                      TAX                      ;low vector address
E6B2  C8                      INY
E6B3  B12C                    LDA      (ICSPRZ),Y      ;high vector address
E6B5  852D                    STA      ICSPRZ+1        ;set high address
E6B7  862C                    STX      ICSPRZ          ;set low address
E6B9  18                      CLC                      ;indicate success

                      ;       Exit.

E6BA  60              CEP2    RTS                      ;return
```

```
                        **      DBL - Decrement Buffer Length
                        *
                        *       ENTRY   JSR     DBL
                        *
                        *       EXIT
                        *               Z set if buffer length = 0
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E6BB          DBL     =       *               ;entry
E6BB    A528                    LDA     ICBLLZ          ;low buffer length
E6BD    D002 ^E6C1              BNE     DBL1            ;if low buffer length non-z:

E6BF    C629                    DEC     ICBLLZ+1        ;decrement high buffer leng:

E6C1    C628            DBL1    DEC     ICBLLZ          ;decrement low buffer lengt:
E6C3    A528                    LDA     ICBLLZ
E6C5    0529                    ORA     ICBLLZ+1        ;indicate buffer length sta:
E6C7    60                      RTS                     ;return



                        **      DBP - Decrement Buffer Pointer
                        *
                        *       ENTRY   JSR     DBP
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E6C8          DBP     =       *               ;entry
E6C8    A524                    LDA     ICBALZ          ;low buffer address
E6CA    D002 ^E6CE              BNE     DBP1            ;if low buffer address non-:

E6CC    C625                    DEC     ICBALZ+1        ;decrement high buffer addr:

E6CE    C624            DBP1    DEC     ICBALZ          ;decrement low buffer addre:
E6D0    60                      RTS                     ;return
```

```
                        **      IBP - Increment Buffer Pointer
                        *
                        *       ENTRY   JSR     IBP
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E6D1          IBP     =       *               ;entry
E6D1    E624                    INC     ICBALZ          ;increment low buffer addre:
E6D3    D002 ^E6D7              BNE     IBP1            ;if low buffer address non-:

E6D5    E625                    INC     ICBALZ+1        ;increment high buffer addr:

E6D7    60              IBP1    RTS                     ;return



                        **      SFL - Set Final Buffer Length
                        *
                        *       ENTRY   JSR     SFL
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E6D8          SFL     =       *               ;entry
E6D8    A62E                    LDX     ICIDNO          ;IOCB index
E6DA    38                      SEC
E6DB    BD4803                  LDA     ICBLL,X         ;initial length
E6DE    E528                    SBC     ICBLLZ          ;subtract byte count
E6E0    8528                    STA     ICBLLZ          ;update length
E6E2    BD4903                  LDA     ICBLH,X
E6E5    E529                    SBC     ICBLLZ+1
E6E7    8529                    STA     ICBLHZ
E6E9    60                      RTS                     ;return



                        **      EHC - Execute Handler Command
                        *
                        *       ENTRY   JSR     EHC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = E6EA          EHC     =       *               ;entry
E6EA    A092                    LDY     #FNCNOT         ;assume function not define:
```

```
E6EC  20F4E6                JSR     IDH             ;invoke device handler
E6EF  8423                  STY     ICSTAZ          ;status
E6F1  C000                  CPY     #0              ;set N accordingly
E6F3  60                    RTS                     ;return



                    **      IDH - Invoke Device Handler
                    *
                    *       ENTRY   JSR     IDH
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = E6F4        IDH     =       *               ;entry
E6F4  AA                    TAX                     ;save A
E6F5  A52D                  LDA     ICSPRZ+1        ;high vector
E6F7  48                    PHA                     ;put high vector on stack
E6F8  A52C                  LDA     ICSPRZ          ;low vector
E6FA  48                    PHA                     ;put low vector on stack
E6FB  8A                    TXA                     ;restore A
E6FC  A62E                  LDX     ICIDNO          ;IOCB index
E6FE  60                    RTS                     ;invoke handler (address on:



                    **      SHT - Search Handler Table
                    *
                    *       ENTRY   JSR     SHT
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = E6FF        SHT     =       *       ;entry

                    ;       Set device number.

E6FF  38                    SEC
E700  A001                  LDY     #1
E702  B124                  LDA     (ICBALZ),Y      ;device number
E704  E931                  SBC     #'1'
E706  3004 ^E70C            BMI     SHT1            ;if number less than  "1"

E708  C909                  CMP     #'9'-'1'+1
E70A  9002 ^E70E            BCC     SHT2            ;if number in range "1" to :

E70C  A900          SHT1    LDA     #0              ;substitute device number ":

E70E  8521          SHT2    STA     ICDNOZ          ;device number (0 through 8:
```

```
E710  E621                    INC     ICDNOZ          ;adjust number to range 1 t:

                    ;         Find device handler.

E712  A000                    LDY     #0              ;offset to device code
E714  B124                    LDA     (ICBALZ),Y      ;device code
                    ;         JMP     FDH             ;find device handler, retur:



                    **        FDH - Find Device Handler
                    *
                    *         ENTRY   JSR     FDH
                    *
                    *         MODS
                    *                 Original Author Unknown
                    *                 1. Bring closer to Coding Standard (object :
                    *                    R. K. Nordin 11/01/83


     = E716         FDH       =       *               ;entry

                    ;         Check device code.

E716  F00C ^E724              BEQ     FDH2            ;if device code null

                    ;         Search handler table for device.

E718  A021                    LDY     #MAXDEV         ;offset to last possible en:

E71A  D91A03        FDH1      CMP     HATABS,Y        ;device code from table
E71D  F009 ^E728              BEQ     FDH3            ;if device found

E71F  88                      DEY
E720  88                      DEY
E721  88                      DEY
E722  10F6 ^E71A              BPL     FDH1            ;if not done

                    ;         Process device not found.

E724  A082          FDH2      LDY     #NONDEV         ;nonexistent device error
E726  38                      SEC                     ;indicate error
E727  60                      RTS                     ;return

                    ;         Set handler ID.

E728  98            FDH3      TYA                     ;offset to device code in t:
E729  8520                    STA     ICHIDZ          ;set handler ID
E72B  18                      CLC                     ;indicate no error
E72C  60                      RTS                     ;return
```

```
                    **      TCVO - Table of Command Vector Offsets
                    *
                    *       Entry n is the vector offset for command n+3.


E72D  00            TCVO    DB      0       ;3 - open
E72E  04                    DB      4       ;4
E72F  04                    DB      4       ;5 - get record
E730  04                    DB      4       ;6
E731  04                    DB      4       ;7 - get byte(s)
E732  06                    DB      6       ;8
E733  06                    DB      6       ;9 - put record
E734  06                    DB      6       ;10
E735  06                    DB      6       ;11 - put byte(s)
E736  02                    DB      2       ;12 - close
E737  08                    DB      8       ;13 - status
E738  0A                    DB      10      ;14 - special
```

```
E739                    **      PHR - Perform Peripheral Handler Loading Initializa:
                        *
                        *       * Performs Power-up Polling, with Handler loading a:
                        *       and Initialization;
                        *       * Performs System Reset Re-initialization of all ha:
                        *
                        *       Input Parameters:
                        *       WARMST (used to distinguish Cold and Warm Start).
                        *
                        *       Output Parameters:
                        *       None.
                        *
                        *       Modified:
                        *       Registers are not saved;
                        *       All kinds of side effects when any handler is loade:
                        *       (potentially MEMLO, DVSTAT thru DVSTAT+3, the DCB,
                        *       CHLINK, ZCHAIN, TEMP1, TEMP2, TEMP3.    This list m:
                        *       not be complete.).
                        *
                        *       ENTRY   JSR     PHR
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = E739            PHR     =       *       ;entry

                        ;       Check for coldstart.

E739  A508                      LDA     WARMST          ;warmstart flag
E73B  F025 ^E762                BEQ     PHR2            ;if coldstart

                        ;       Process warmstart.

E73D  A9E9                      LDA     #low [CHLINK-18]
E73F  854A                      STA     ZCHAIN
E741  A903                      LDA     #high [CHLINK-18]
E743  854B                      STA     ZCHAIN+1

                        ;       Check next link.

E745  A012              PHR1    LDY     #18             ;offset to link
E747  18                        CLC
E748  B14A                      LDA     (ZCHAIN),Y      ;low link
E74A  AA                        TAX
E74B  C8                        INY
E74C  714A                      ADC     (ZCHAIN),Y      ;high link
E74E  F026 ^E776                BEQ     PHR4            ;if forward link null

                        ;       Re-initialize peripheral handler.

E750  B14A                      LDA     (ZCHAIN),Y      ;high link
E752  854B                      STA     ZCHAIN+1
E754  864A                      STX     ZCHAIN
E756  2056C8                    JSR     CLT             ;checksum linkage table
```

```
E759  D01B ^E776          BNE     PHR4            ;if checksum bad

E75B  2094E8              JSR     PHW             ;re-initialize peripheral h:
E75E  B016 ^E776          BCS     PHR4            ;if error

                  ;       Continue with next handler.

E760  90E3 ^E745          BCC     PHR1            ;continue with next handler

                  ;       Process coldstart.

E762  A900        PHR2    LDA     #0
E764  8DFB03              STA     CHLINK          ;clear chain link
E767  8DFC03              STA     CHLINK+1
E76A  A94F                LDA     #$4F            ;send POLL RESET poll
E76C  D02D ^E79B          BNE     PHR7

                  ;       Perform type 3 poll.

E76E  A900        PHR3    LDA     #0
E770  A8                  TAY
E771  208EE7              JSR     PHP
E774  1001 ^E777          BPL     PHR5            ;if poll answered

                  ;       Exit.

E776  60          PHR4    RTS                     ;return

                  ;       Process answered poll.

E777  18          PHR5    CLC
E778  ADE702              LDA     MEMLO
E77B  6DEA02              ADC     DVSTAT
E77E  8D1203              STA     TEMP1
E781  ADE802              LDA     MEMLO+1
E784  6DE802              ADC     DVSTAT+1
E787  8D1303              STA     TEMP1+1         ;(TEMP2 := MEMLO + handler :
E78A  38                  SEC
E78B  ADE502              LDA     MEMTOP
E78E  ED1203              SBC     TEMP1
E791  ADE602              LDA     MEMTOP+1
E794  ED1303              SBC     TEMP1+1         ;(subtract MEMTOP)
E797  B009 ^E7A2          BCS     PHR8            ;if room to load

                  ;       Prepare for another poll.

E799  A94E        PHR6    LDA     #$4E            ;following any load or init:
                                                  ;prepare for another Type 3:
                                                  ;sending a "special" load c:
                                                  ;serial port.

                  ;       Poll.

E79B  A8          PHR7    TAY                     ;Send either "special" load:
E79C  208EE7              JSR     PHP
E79F  4C6EE7              JMP     PHR3            ;go poll again
```

```
                        ;       Load peripheral handler.

E7A2  ADEC02      PHR8  LDA      DVSTAT+2         ;call the loader
E7A5  AEE702            LDX      MEMLO
E7A8  8EEC02            STX      DVSTAT+2         ;(Parameter = load address)
E7AB  AEE802            LDX      MEMLO+1
E7AE  8EED02            STX      DVSTAT+3
E7B1  20DEE7            JSR      LPH              ;load peripheral handler
E7B4  30E3 ^E799        BMI      PHR6             ;if load error, poll again

E7B6  38                SEC                       ;Call for initialize new ha:
E7B7  209EE8            JSR      PHC              ;(Parameter = add size to M:
E7BA  B0DD ^E799        BCS      PHR6             ;if init error, poll again

E7BC  90B0 ^E76E        BCC      PHR3             ;poll again normally



                        **      PHP - Perform Poll
                        *
                        *       Polling subroutine calls SIO for Type 3 or 4 Poll.
                        *
                        *       Input Parameters:
                        *       A       Value for AUX1
                        *       Y       Value for AUX2
                        *
                        *       Output Parameters:
                        *       Y       SIO status from poll
                        *       DVSTAT: Device minimum size (low), if poll answered
                        *       DVSTAT+1: Device minimum size (high), if poll answe:
                        *       DVSTAT+2: Device address for loading, if poll answe:
                        *       DVSTAT+3: Device version number, if poll answered
                        *
                        *       Modified:
                        *       The registers are not saved.
                        *
                        *       Subroutines called:
                        *       SIO (performs poll and returns to PHP's caller).
                        *
                        *       ENTRY   JSR     PHP
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = E7BE      PHP   =        *                ;entry

                        ;       Initialize.

E7BE  48                PHA                       ;save parameter

                        ;       Set up DCB.

E7BF  A209              LDX      #PHPAL-1         ;offset to last byte of DCB:
```

```
E7C1  BDD4E7      PHP1     LDA     PHPA,X          ;byte of DCB data
E7C4  9D0003               STA     DCB,X           ;byte of DCB
E7C7  CA                   DEX
E7C8  10F7 ^E7C1           BPL     PHP1            ;if not done

                  ;        Set parameters in DCB auxiliary bytes.

E7CA  8C0B03               STY     DAUX2
E7CD  68                   PLA
E7CE  8D0A03               STA     DAUX1

                  ;        Perform SIO.

E7D1  4C59E4               JMP     SIOV            ;vector to SIO, return


                  ;        DCB Poll Request Data

E7D4  4F          PHPA     DB      $4F     ;device bus ID
E7D5  01                   DB      1       ;unit number
E7D6  40                   DB      '@'     ;type 3 or 4 poll command
E7D7  40                   DB      $40     ;I/O direction
E7D8  EA02                 DW      DVSTAT  ;buffer
E7DA  1E                   DB      30      ;timeout
E7DB  00                   DB      0
E7DC  0400                 DW      4       ;buffer length

      = 000A      PHPAL    =       *-PHPA  ;length


                  **       LPH - Load Peripheral Handler
                  *
                  *        This subroutine calls the relocating loader to load
                  *        a handler from a peripheral.
                  *
                  *        Input parameters:
                  *        A       Peripheral serial address for load;
                  *        DVSTAT+2: Load address (low)
                  *        DVSTAT+3: Load address (high)
                  *
                  *        Output parameters:
                  *        From the relocating loader.
                  *
                  *        Modified:
                  *        TEMP1, TEMP2, TEMP3,
                  *        DVSTAT+3, DVSTAT+3 (forced even),
                  *        Relocating loader variables and parameters,
                  *        Registers not saved.
                  *
                  *        Subroutines called:
                  *        RLR (relocating loader).
                  *
                  *        ENTRY   JSR     LPH
                  *
```

```
                         *       MODS
                         *               R. S. Scheiman  04/01/82
                         *               1. Bring closer to Coding Standard (object :
                         *                  R. K. Nordin 11/01/83


       = E7DE            LPH     =       *       ;entry

                         ;       Initialize.

E7DE  8D1303                     STA     TEMP2   ;save peripheral address
E7E1  A200                       LDX     #0
E7E3  8E1203                     STX     TEMP1   ;set starting block number
E7E6  CA                         DEX
E7E7  8E1503                     STX     TEMP3   ;set starting byte number

                         ;       Ensure load address even.

E7EA  ADEC02                     LDA     DVSTAT+2        ;low load address
E7ED  6A                         ROR     A
E7EE  9008 ^E7F8                 BCC     LPH1            ;if even

E7F0  EEEC02                     INC     DVSTAT+2        ;increment low load address
E7F3  D003 ^E7F8                 BNE     LPH1            ;if no carry

E7F5  EEED02                     INC     DVSTAT+3        ;increment high load addres:

                         ;       Set up relocating loader parameters.

E7F8  ADEC02             LPH1    LDA     DVSTAT+2        ;load address
E7FB  8DD102                     STA     LOADAD
E7FE  ADED02                     LDA     DVSTAT+3
E801  8DD202                     STA     LOADAD+1
E804  A916                       LDA     #low PHG        ;get-byte routine address
E806  8DCF02                     STA     GBYTEA
E809  A9E8                       LDA     #high PHG
E80B  8DD002                     STA     GBYTEA+1
E80E  A980                       LDA     #$80            ;loader page zero load addr:
E810  8DD302                     STA     ZLOADA

                         ;       Reloacte routine.

E813  4C45C7                     JMP     RLR             ;relocate routine, return
```

```
                        **      PHG - Perform Peripheral Handler GET-BYTE
                        *
                        *       Get a byte subroutine for relocating loader passes
                        *       bytes from peripheral to relocating loader via
                        *       cassette buffer. Calls GNL each time new
                        *       buffer is needed.
                        *
                        *       Input parameters:
                        *       TEMP1: Next block number;
                        *       TEMP2: Peripheral address (for GNL);
                        *       TEMP3: Next byte number (index to CASBUF).
                        *
                        *       Output parameters (for relocating loader):
                        *       Carry bit on indicates error;
                        *       A       Next byte, if no error.
                        *
                        *       Modified:
                        *       Cassette buffer CASBUF;
                        *       TEMP3;
                        *       X, Y not saved.
                        *
                        *       Subroutines called:
                        *       GNL, which calls SIO to get load records.
                        *
                        *       ENTRY   JSR     PHG
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *.                 R. K. Nordin 11/01/83


     = E816             PHG     =       *                       ;entry

                        ;       Check for another byte in buffer.

E816  AE1503                    LDX     TEMP3
E819  E8                        INX
E81A  8E1503                    STX     TEMP3
E81D  F008 ^E827                BEQ     PHG2                    ;if empty, load next block

                        ;       Retrieve next byte.

E81F  AE1503            PHG1    LDX     TEMP3
E822  BD7D03                    LDA     CASBUF-$80,X    ;byte
E825  18                        CLC                             ;indicate no error
E826  60                        RTS                             ;return

                        ;       Load next block and retrieve next byte.

E827  A980              PHG2    LDA     #-128   ;offset to first byte
E829  8D1503                    STA     TEMP3
E82C  2033E8                    JSR     GNL     ;get next load block
E82F  10EE ^E81F                BPL     PHG1    ;if no error, retrieve next byte

                        ;       Process error.
```

```
E831  38                   SEC             ;indicate error
E832  60                   RTS             ;return



                   **      GNL - Get Next Load Block
                   *
                   *       Subroutine to get a load block from the peripheral.
                   *
                   *       Input parameters:
                   *       TEMP1: Block number.
                   *
                   *       Output parameters:
                   *       Negative bit is set by SIO if I/O error occurs.
                   *
                   *       Modified:
                   *       TEMP1;
                   *       the DCB (SIO);
                   *       Registers not saved.
                   *
                   *       Subroutines called:
                   *       SIO.
                   *
                   *       ENTRY   JSR     GNL
                   *
                   *       MODS
                   *               R. S. Scheiman  04/01/82
                   *               1. Bring closer to Coding Standard (object :
                   *                  R. K. Nordin 11/01/83

      = E833      GNL      =       *               ;entry

                   ;       Set up DCB.

E833  A20B                 LDX     #GNLAL-1        ;offset to last DCB data by:

E835  BD51E8      GNL1     LDA     GNLA,X          ;byte of DCB data
E838  9D0003               STA     DCB,X           ;byte of DCB
E83B  CA                   DEX
E83C  10F7 ^E835           BPL     GNL1            ;if not done

                   ;       Set DCB parameters.

E83E  AE1203               LDX     TEMP1           ;block number
E841  8E0A03               STX     DAUX1           ;auxiliary 1
E844  E8                   INX
E845  8E1203               STX     TEMP1           ;next block number
E848  AD1303               LDA     TEMP2           ;device address
E84B  8D0003               STA     DDEVIC          ;device bus ID

                   ;       Perform SIO.

E84E  4C59E4               JMP     SIOV            ;vector to SIO, return
```

```
                        ;       DCB Data

E851  00                GNLA    DB      $00     ;dummy device bus ID
E852  01                        DB      1       ;dummy unit number
E853  26                        DB      '&'     ;load command
E854  40                        DB      $40     ;I/O direction
E855  FD03                      DW      CASBUF  ;buffer
E857  1E                        DB      30      ;timeout
E858  00                        DB      0
E859  8000                      DW      128     ;buffer length
E85B  00                        DB      0       ;auxiliary 1
E85C  00                        DB      0       ;auxiliary 2

    = 000C              GNLAL   =       *-GNLA  ;length


                        **      SHC - Search Handler Chain
                        *
                        *       Forward chain search searches for pointer to handle:
                        *       table whose address matches caller's parameter. If :
                        *       parameter is zero, this routine looks for the point:
                        *       the final linkage table since this table's forward :
                        *       is zero (null).
                        *
                        *       Input parameters:
                        *       A       Linkage table address to match (High)
                        *       Y       Linkage table address to match (Low)
                        *
                        *       Output parameters:
                        *       ZCHAIN points to linkage table whose forward pointe:
                        *               contains the match (if match is found);
                        *               if the match is found just following the li:
                        *               chain base CHLINK, then ZCHAIN points to CH:
                        *               minus 18;
                        *       If match successful, A (High) and X (Low) contain
                        *               matched address (equiv. to A and Y parms.);
                        *       Carry bit is set to indicate no match or checksum v:
                        *               along the chain. [Note: the linkage table p:
                        *               to by ZCHAIN upon return is not checksum ch:
                        *
                        *       Modified:
                        *       TEMP1, TEMP2, ZCHAIN;
                        *       The registers are not saved.
                        *
                        *       Subroutines called:
                        *       CLT.
                        *
                        *       ENTRY   JSR     SHC
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83
```

```
        = E85D      SHC     =       *           ;entry

                    ;       Initialize.

E85D  8C1203                STY     TEMP1
E860  8D1303                STA     TEMP1+1
E863  A9E9                  LDA     #low [CHLINK-18]
E865  854A                  STA     ZCHAIN          ;start ZCHAIN at proper off:
E867  A903                  LDA     #high [CHLINK-18]
E869  854B                  STA     ZCHAIN+1

                    ;       Check for match.

E86B  A012          SHC1    LDY     #18
E86D  B14A                  LDA     (ZCHAIN),Y
E86F  AA                    TAX                     ;low chain pointer
E870  C8                    INY
E871  B14A                  LDA     (ZCHAIN),Y      ;high chain pointer
E873  CD1303                CMP     TEMP2           ;check for match with param:
E876  D007 ^E87F            BNE     SHC2            ;if no match

E878  EC1203                CPX     TEMP1
E87B  D002 ^E87F            BNE     SHC2            ;if no match

                    ;       Exit.

E87D  18                    CLC             ;indicate match
E87E  60                    RTS             ;return

                    ;       Check for end of chain.

E87F  C900          SHC2    CMP     #0      ;end of chain indicator
E881  D006 ^E889            BNE     SHC4    ;if not end of chain

E883  E000                  CPX     #0
E885  D002 ^E889            BNE     SHC4    ;if not end of chain

                    ;       Process end of chain or checksum error.

E887  38            SHC3    SEC             ;return error (checksum or end)
E888  60                    RTS             ;return

                    ;       Set link to new linkage table.

E889  864A          SHC4    STX     ZCHAIN  ;link to new linkage table
E88B  854B                  STA     ZCHAIN+1

E88D  2056CB                JSR     CLT     ;checksum linkage table
E890  D0F5 ^E887            BNE     SHC3    ;if error

                    ;       Continue searching chain.

E892  F0D7 ^E86B            BEQ     SHC1    ;continue searching chain
```

```
**      PHW - Perform Peripheral Handler Warmstart Initiali:
*
*       PHC is the main entry. This performs full initializ:
*               including adding the new linkage table into:
*               table chain;
*       PHW does all initialization except adding to the li:
*               table chain (intended for warm start reinit:
*       PHI is the full initialization entry for calling
*               init from outside the OS.
*
*       The code does the following:
*       1)      Links new handler to end of chain;
*       2)      Calls handler init subroutine in handler;
*       3)      If 2 failed, unlinks handler from chain,
*               and returns with carry;
*       4)      Else, conditionally zeroes handler size ent:
*               handler linkage table (per parameter);
*       5)      Adds handler size entry (possibly zeroed) t:
*       6)      If handler size entry is nonzero, MEMLO is :
*               forced even;
*       7)      Calculates and enters linkage table checksu:
*       8)      Returns with carry clear.
*
*       PHC is called by PHR when loading handlers at cold
*               initialization;and by PHL when loading a ha:
*               application request under CIO;
*       PHW is called by PHR to reinitialize a handler duri:
*               warm-start;
*       PHI is vectored by OS vector at $E49E and is intend:
*               for use by system-level applications which :
*               handlers (ie., AUTORUN.SYS handler loader, :
*
*       Input parameters:
*       PHC:
*               DVSTAT, DVSTAT+1 contain handler size (for
*               handler init, not used by this routine);
*               DVSTAT+2, DVSTAT+3 contain handler linkage :
*               address.
*       PHW:
*               DVSTAT+2, DVSTAT+3 same;
*               DVSTAT, DVSTAT+1 undefined.
*       PHI:
*               A and Y contain handler linkage table addre:
*               they are copied into DVSTAT+3 and DVSTAT+2;
*               DVSTAT, DVSTAT+1 may or may not be signific:
*               any concern about these are up to the progr:
*               of the peripheral handler init routine and :
*               is making use of the non-OS-caller entry PH:
*
*       For PHI and PHC, the Carry bit specifies whether
*               the handler size entry of the linkage table:
*               be zeroed prior to adding to MEMLO: Carry s:
*               do NOT zero this entry.
*
*       Output parameters:
*       Carry indicates error (initialization failed);
*       The registers are not saved.
```

```
                        *
                        *       Modified:
                        *       DVSTAT+2, DVSTAT+3 are modified by PHI;
                        *       ZCHAIN, TEMP1, TEMP2;
                        *       MEMLO, MEMLO+1 conditionally incremented by handler:
                        *
                        *       Subroutines called:
                        *       SHC (to find end of linkage table chain);
                        *       PHU (to unlink handler if init. error);
                        *       CLT (to insert linkage table checksum);
                        *       loaded handler's INIT entry.
                        *
                        *       ENTRY   JSR     PHW
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = E894     PHW     =       *       ;entry
E894  38                SEC             ;indicate not zeroing handler size
E895  08                PHP
E896  B028 ^E8C0        BCS     PHQ     ;initialize handler and update MEML:



                **      PHI - Perfrom Peripheral Handler Initialization wit:
                *
                *       ENTRY   JSR     PHI
                *
                *       MODS
                *               R. S. Scheiman  04/01/82
                *               1. Bring closer to Coding Standard (object :
                *                  R. K. Nordin 11/01/83


     = E898     PHI     =       *               ;entry
E898  8DED02            STA     DVSTAT+3
E89B  8CEC02            STY     DVSTAT+2
                ;       JMP     PHC             ;perfrom coldstart initiali:
                        **      PHC - Perfrom Peripheral Handler Coldstart Initiali:
                        *
                        *       ENTRY   JSR     PHC
                        *
                        *       MODS
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = E89E     PHC     =       *       ;entry

                        ;       Initialize.

E89E  08                        PHP

                        ;       Search for end of chain.

E89F  A900                      LDA     #0      ;indicate searching for end of chai:
E8A1  A8                        TAY
E8A2  205DE8                    JSR     SHC     ;search handler chain
E8A5  B027 ^E8CE                BCS     PHQ1    ;if error, exit

                        ;       Enter at end of chain.

E8A7  A012                      LDY     #18             ;offset
E8A9  ADEC02                    LDA     DVSTAT+2
E8AC  914A                      STA     (ZCHAIN),Y      ;low link
E8AE  AA                        TAX
E8AF  C8                        INY
E8B0  ADED02                    LDA     DVSTAT+3
E8B3  914A                      STA     (ZCHAIN),Y      ;high link
E8B5  864A                      STX     ZCHAIN          ;link to new table
E8B7  854B                      STA     ZCHAIN+1
E8B9  A900                      LDA     #0              ;indicate end of chain
E8BB  914A                      STA     (ZCHAIN),Y      ;low link
E8BD  88                        DEY
E8BE  914A                      STA     (ZCHAIN),Y      ;high link

                        ;       Initialize handler.

                        ;       JMP     PHQ             ;initialize handler, return
```

```
                          **      PHQ - Initialize Handler and Update MEMLO
                          *
                          *       ENTRY    JSR      PHQ
                          *
                          *       MODS
                          *                R. S. Scheiman  04/01/82
                          *                1. Bring closer to Coding Standard (object :
                          *                   R. K. Nordin 11/01/83

      = E8C0              PHQ     =        *                  ;entry

                          ;       Initialize handler.

E8C0  2000E9                      JSR      PHX                ;initialize handler
E8C3  900C ^E8D1                  BCC      PHQ2               ;if no error

                          ;       Process error.

E8C5  ADED02                      LDA      DVSTAT+3
E8C8  ACEC02                      LDY      DVSTAT+2
E8CB  2015E9                      JSR      PHU                ;unlink handler

                          ;       Exit, indicating error.

E8CE  28                  PHQ1    PLP                         ;fix stack
E8CF  38                          SEC                         ;indicate error
E8D0  60                          RTS                         ;return

                          ;       Check for zeroing handler size.

E8D1  28                  PHQ2    PLP
E8D2  B009 ^E8DD                  BCS      PHQ3               ;if not to zero

                          ;       Zero handler size.

E8D4  A900                        LDA      #0
E8D6  A010                        LDY      #16                ;offset
E8D8  914A                        STA      (ZCHAIN),Y         ;zero size
E8DA  C8                          INY
E8DB  914A                        STA      (ZCHAIN),Y

                          ;       Increase MEMLO by size.

E8DD  18                  PHQ3    CLC
E8DE  A010                        LDY      #16                ;offset to size
E8E0  ADE702                      LDA      MEMLO
E8E3  714A                        ADC      (ZCHAIN),Y         ;add low size
E8E5  8DE702                      STA      MEMLO              ;new low MEMLO
E8E8  C8                          INY
E8E9  ADE802                      LDA      MEMLO+1
E8EC  714A                        ADC      (ZCHAIN),Y         ;add high size
E8EE  8DE802                      STA      MEMLO+1            ;new high MEMLO

                          ;       Put checksum in linkage table.

E8F1  A00F                        LDY      #15                ;offset to checksum
```

```
E8F3  A900                LDA    #0
E8F5  914A                STA    (ZCHAIN),Y       ;clear checksum
E8F7  2056CB              JSR    CLT              ;checksum linkage table
E8FA  A00F                LDY    #15              ;offset to checksum
E8FC  914A                STA    (ZCHAIN),Y       ;checksum

                   ;      Exit.

E8FE  18                  CLC                     ;indicate success
E8FF  60                  RTS                     ;return



                   **     PHX - Initialize Handler
                   *
                   *      ENTRY   JSR     PHX
                   *
                   *      MODS
                   *              R. S. Scheiman  04/01/82
                   *              1. Bring closer to Coding Standard (object :
                   *                 R. K. Nordin 11/01/83


      = E900       PHX    =      *                ;entry
E900  18                  CLC
E901  A54A                LDA    ZCHAIN
E903  690C                ADC    #12
E905  8D1203              STA    TEMP1            ;low handler initialization:
E908  A54B                LDA    ZCHAIN+1
E90A  6900                ADC    #0
E90C  8D1303              STA    TEMP1+1          ;high handler initializatio:
E90F  6C1203              JMP    (TEMP1)          ;initialize handler, return
```

```
E912                        FIX     $E912



                  **        E912 - $E912 Patch
                  *
                  *         For compatibilty with OS Revision B, set VBLANK par:

E912  4C72C2                JMP     SVP      ;set VBLANK parameters, return
```

```
E915                  **      PHU - Perform Peripheral Handler Unlinking
                      *
                      *       Handler entry unlinking routine. This routine is ca:
                      *       by the OS handler initialization to unlink a handle:
                      *       initialization fails, or by the handler itself if i:
                      *       the handler unload feature.     This routine is ent:
                      *       OS vector at $E49B.
                      *
                      *       Input parameters:
                      *       A        Address of linkage table to unlink (High);
                      *       Y        Address of linkage table to unlink (Low).
                      *       COLDST: Tested to see if PHU is called during cold :
                      *                if so, chain entry is unlinked even if at M:
                      *
                      *       Output parameters:
                      *       Carry is set to indicate error;in this case,
                      *                no unlinking has occurred.
                      *
                      *       Modified:
                      *       TEMP1, TEMP2;
                      *       ZCHAIN,ZCHAIN+1;
                      *       The forward chain pointer in the predecessor of the:
                      *       table being removed is modified to point to the suc:
                      *       of the removed table if the removal is successful--
                      *       this forward chain pointer may be CHLINK,CHLINK+1.
                      *
                      *       The registers are not saved.
                      *
                      *       Subroutines called:
                      *       SHC, CLT.
                      *
                      *       ENTRY   JSR     PHU
                      *
                      *       MODS
                      *                R. S. Scheiman  04/01/82
                      *                1. Bring closer to Coding Standard (object :
                      *                   R. K. Nordin 11/01/83

      = E915          PHU     =       *                    ;entry

                      ;       Search handler chain.

E915  205DE8                  JSR     SHC                  ;search handler chain
E918  B038 ^E955              BCS     PHU3                 ;if error

                      ;       Perform unlinking.

E91A  A8                      TAY                          ;(save return parameter)
E91B  A54A                    LDA     ZCHAIN               ;save ZCHAIN (points to pre:
E91D  48                      PHA
E91E  A54B                    LDA     ZCHAIN+1
E920  48                      PHA
E921  864A                    STX     ZCHAIN               ;make ZCHAIN point to linka:
E923  844B                    STY     ZCHAIN+1             ;to be removed
E925  AD4402                  LDA     COLDST               ;coldstart flag
E928  D00F ^E939              BNE     PHU1                 ;if coldstart, unconditiona:
```

```
E92A  A010                 LDY     #16             ;check if loaded at MEMLO..:
E92C  18                   CLC                     ;by checking if size is non:
E92D  B14A                 LDA     (ZCHAIN),Y
E92F  C8                   INY
E930  714A                 ADC     (ZCHAIN),Y
E932  D01F ^E953           BNE     PHU2            ;if handler size non-zero

E934  2056C6               JSR     CLT             ;checksum linkage table
E937  D01A ^E953           BNE     PHU2            ;if checksum nonzero, bad c:

E939  A012         PHU1    LDY     #18             ;take link from table being:
E93B  B14A                 LDA     (ZCHAIN),Y
E93D  AA                   TAX
E93E  C8                   INY
E93F  B14A                 LDA     (ZCHAIN),Y
E941  A8                   TAY
E942  68                   PLA             ;Make ZCHAIN point to the predecess:
E943  854B                 STA     ZCHAIN+1
E945  68                   PLA
E946  854A                 STA     ZCHAIN
E948  98                   TYA             ;And put forward link from table be:
E949  A013                 LDY     #19     ;removed into its predecessors link:
E94B  914A                 STA     (ZCHAIN),Y
E94D  88                   DEY
E94E  8A                   TXA
E94F  914A                 STA     (ZCHAIN),Y
E951  18                   CLC             ;indicate success
E952  60                   RTS             ;return

                   ;       Clean stack and process error.

E953  68           PHU2    PLA             ;Error return--restore stack
E954  68                   PLA

                   ;       Process error.

E955  38           PHU3    SEC             ;indicate error
E956  60                   RTS             ;return
```

```
E957                        FIX     $E959



                    **      E959 - $E959 Patch
                    *
                    *       For compatibilty with OS Revision B, perform PIO.

E959  4C33C9                JMP     PIO     ;perform PIO, return
```

```
E95C                 **      ISIO - Initialize SIO
                     *
                     *       ENTRY   JSR     ISIO
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = E95C         ISIO    =       *       ;entry

E95C  A93C                   LDA     #MOTRST
E95E  8D02D3                 STA     PACTL   ;turn off motor

E961  A93C                   LDA     #NCOMHI
E963  8D03D3                 STA     PBCTL   ;raise NOT COMMAND line

E966  A903                   LDA     #$03    ;POKEY out of initialize mode
E968  8D3202                 STA     SSKCTL  ;SKCTL shadow
E96B  8541                   STA     SOUNDR  ;select noisy I/O
E96D  8D0FD2                 STA     SKCTL

E970  60                     RTS             ;return



                     **      SIO - Serial Input/Output
                     *
                     *       ENTRY   JSR     SIO
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = E971         SIO     =       *       ;entry

                     ;       Initialize.

E971  BA                     TSX
E972  8E1803                 STX     STACKP  ;save stack pointer
E975  A901                   LDA     #1      ;critical section indicator
E977  8542                   STA     CRITIC  ;indicate critical section

                     ;       Check device ID.

E979  AD0003                 LDA     DDEVIC  ;device ID
E97C  C960                   CMP     #CASET
E97E  D003 ^E983             BNE     SIO1    ;if not cassette

                     ;       Process cassette.

E980  4C9DEB                 JMP     PCI     ;process cassette I/O, return
```

```
                        ;       Process intelligent device.

E983  A900      SI01    LDA     #0
E985  8D0F03            STA     CASFLG  ;indicate not cassette

E988  A901              LDA     #DRETRI
E98A  8DBD02            STA     DRETRY  ;set device retry count

E98D  A90D      SI02    LDA     #CRETRI
E98F  8D9C02            STA     CRETRY  ;set command frame retry count

                        ;       Send command frame.

E992  A928      SI03    LDA     #low B19200
E994  8D04D2            STA     AUDF3           ;set baud rate to 19200
E997  A900              LDA     #high B19200
E999  8D06D2            STA     AUDF4

                        ;       Set up command buffer.

E99C  18                CLC
E99D  AD0003            LDA     DDEVIC          ;device ID
E9A0  6D0103            ADC     DUNIT           ;add unit number
E9A3  69FF              ADC     #$FF            ;subtract 1
E9A5  8D3A02            STA     CDEVIC          ;device bus ID
E9A8  AD0203            LDA     DCOMND          ;command
E9AB  8D3B02            STA     CCOMND
E9AE  AD0A03            LDA     DAUX1           ;auxiliary information 1
E9B1  8D3C02            STA     CAUX1
E9B4  AD0B03            LDA     DAUX2           ;auxiliary information 2
E9B7  8D3D02            STA     CAUX2

                        ;       Set buffer pointer to command frame buffer.

E9BA  18                CLC
E9BB  A93A              LDA     #low CDEVIC     ;low buffer address
E9BD  8532              STA     BUFRLO          ;low buffer address
E9BF  6904              ADC     #4
E9C1  8534              STA     BFENLO          ;low buffer end address
E9C3  A902              LDA     #high CDEVIC    ;high buffer address
E9C5  8533              STA     BUFRHI          ;high buffer address
E9C7  8535              STA     BFENHI          ;high buffer end address

                        ;       Send command frame to device.

E9C9  A934              LDA     #NCOMLO
E9CB  8D03D3            STA     PBCTL           ;lower NOT COMMAND line
E9CE  20AFEC            JSR     SID             ;send command frame
E9D1  AD3F02            LDA     ERRFLG          ;error flag
E9D4  D003 ^E9D9        BNE     SI04            ;if error received

E9D6  98                TYA                     ;status
E9D7  D008 ^E9E1        BNE     SI05            ;if ACK received

                        ;       Process NAK or timeout.

E9D9  CE9C02    SI04    DEC     CRETRY  ;decrement command frame retry coun
```

```
E9DC  10B4 ^E992          BPL     SI03    ;if retries not exhausted

                    ;     Process command frame retries exhausted.

E9DE  4C22EA              JMP     SI010   ;process error

                    ;     Process ACK.

E9E1  AD0303      SI05    LDA     DSTATS
E9E4  100D ^E9F3          BPL     SI06    ;if no data to send

                    ;     Send data frame to device.

E9E6  A90D                LDA     #CRETRI
E9E8  8D9C02              STA     CRETRY  ;set command frame retry count
E9EB  2087EB              JSR     SBP     ;set buffer pointers
E9EE  20AFEC              JSR     SID     ;send data frame
E9F1  F02F ^EA22          BEQ     SI010   ;if error

                    ;     Wait for complete.

E9F3  209AEC      SI06    JSR     GTO     ;set device timeout
E9F6  A900                LDA     #0
E9F8  8D3F02              STA     ERRFLG  ;clear error flag
E9FB  20C0EC              JSR     STW     ;set timer and wait
E9FE  F012 ^EA12          BEQ     SI08    ;if timeout

                    ;     Process no timeout.

EA00  2C0303              BIT     DSTATS
EA03  7007 ^EA0C          BVS     SI07    ;if more data follows

EA05  AD3F02              LDA     ERRFLG  ;error flag
EA08  D018 ^EA22          BNE     SI010   ;if error

                    ;     Process no error.

EA0A  F01E ^EA2A          BEQ     CSO     ;complete SIO operation

                    ;     Receive data frame from device.

EA0C  2087EB      SI07    JSR     SBP     ;set buffer pointers
EA0F  20FDEA              JSR     REC     ;receive

                    ;     Check error flag.

EA12  AD3F02      SI08    LDA     ERRFLG  ;error flag
EA15  F005 ^EA1C          BEQ     SI09    ;if no error preceded data

                    ;     Process error.

EA17  AD1903              LDA     TSTAT   ;temporary status
EA1A  8530                STA     STATUS  ;status

                    ;     Check status.

EA1C  A530        SI09    LDA     STATUS  ;status
```

```
EA1E  C901                    CMP     #SUCCES
EA20  F008 ^EA2A              BEQ     CSO      ;if successful, complete operation,:

                      ;       Process error.

EA22  CEBD02          SIO10   DEC     DRETRY   ;decrement device retry count
EA25  3003 ^EA2A              BMI     CSO      ;if retries exhausted, complete, re:

                      ;       Retry.

EA27  4C8DE9                  JMP     SIO2     ;retry



                      **      CSO - Complete SIO Operation
                      *
                      *       ENTRY   JSR     CSO
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


      = EA2A          CSO     =       *        ;entry
EA2A  2084EC                  JSR     DSR      ;disable SEND and RECEIVE
EA2D  A900                    LDA     #0       ;not critical section indicator
EA2F  8542                    STA     CRITIC   ;critical section flag
EA31  A430                    LDY     STATUS   ;status
EA33  8C0303                  STY     DSTATS   ;status
EA36  60                      RTS              ;return



                      **      WCA - Wait for Completion or ACK
                      *
                      *       ENTRY   JSR     WCA
                      *
                      *       EXIT
                      *               Y = 0, if failure
                      *                 = $FF, if success
                      *
                      *       NOTES
                      *               Problem: WCA does not handle NAK correctly?:
                      *               Just before WCA3 should be removed.
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83

      = EA37          WCA     =       *                ;entry

                      ;       Initialize.
```

```
EA37  A900                 LDA    #0
EA39  8D3F02               STA    ERRFLG        ;clear error flag

                    ;      Set buffer pointer.

EA3C  18                   CLC
EA3D  A93E                 LDA    #low TEMP     ;low temporary address
EA3F  8532                 STA    BUFRLO        ;low buffer address
EA41  6901                 ADC    #1
EA43  8534                 STA    BFENLO        ;low buffer end address
EA45  A902                 LDA    #high TEMP    ;high temporary address
EA47  8533                 STA    BUFRHI        ;high buffer address
EA49  8535                 STA    BFENHI        ;high buffer end address
EA4B  A9FF                 LDA    #$FF
EA4D  853C                 STA    NOCKSM        ;indicate no checksum follo:
EA4F  20FDEA               JSR    REC           ;receive
EA52  A0FF                 LDY    #$FF          ;assume success
EA54  A530                 LDA    STATUS        ;status
EA56  C901                 CMP    #SUCCES
EA58  D019 ^EA73           BNE    WCA2          ;if failure

EA5A  AD3E02               LDA    TEMP          ;byte received
EA5D  C941                 CMP    #ACK
EA5F  F021 ^EA82           BEQ    WCA4          ;if ACK, exit

EA61  C943                 CMP    #COMPLT
EA63  F01D ^EA82           BEQ    WCA4          ;if complete, exit

EA65  C945                 CMP    #ERROR
EA67  D006 ^EA6F           BNE    WCA1          ;if device did not send bac:

                    ;      Process unrecognized response.

EA69  A990                 LDA    #DERROR
EA6B  8530                 STA    STATUS        ;indicate device error
EA6D  D004 ^EA73           BNE    WCA2          ;check for timeout

                    ;      Process nothing sent back.

EA6F  A98B          WCA1   LDA    #DNACK
EA71  8530                 STA    STATUS        ;indicate NAK

                    ;      Check for timeout.

EA73  A530          WCA2   LDA    STATUS        ;status
EA75  C98A                 CMP    #TIMOUT
EA77  F007 ^EA80           BEQ    WCA3          ;if timeout

                    ;      Process other error.

EA79  A9FF                 LDA    #$FF          ;error indicator
EA7B  8D3F02               STA    ERRFLG        ;indicate error
EA7E  D002 ^EA82           BNE    WCA4          ;exit

                    ;      Indicate failure.
```

```
EA80  A000        WCA3    LDY     #0              ;failure indicator

                  ;       Exit.

EA82  A530        WCA4    LDA     STATUS          ;status
EA84  8D1903              STA     TSTAT           ;temporary status
EA87  60                  RTS                     ;return


                  **      SEN - Send
                  *
                  *       SEN sends a buffer over the serial bus.
                  *
                  *       ENTRY   JSR     SEN
                  *
                  *       NOTES
                  *               Problem: an interrupt may occur before CHKS:
                  *               initialized, causing an incorrect checksum :
                  *               STA CHKSUM should precede STA SEROUT.
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83

      = EA88      SEN     =       *               ;entry

                  ;       Initialize.

EA88  A901                LDA     #SUCCES         ;assume success
EA8A  8530                STA     STATUS          ;status
EA8C  2017EC              JSR     ESS             ;enable SIO SEND
EA8F  A000                LDY     #0
EA91  8431                STY     CHKSUM          ;clear checksum
EA93  843B                STY     CHKSNT          ;clear checksum sent flag
EA95  843A                STY     XMTDON          ;clear transmit-frame done :

                  ;       Initiate TRANSMIT.

EA97  B132                LDA     (BUFRLO),Y      ;first byte from buffer
EA99  8D0DD2              STA     SEROUT          ;serial output register
EA9C  8531                STA     CHKSUM          ;checksum

                  ;       Check BREAK key.

EA9E  A511        SEN1    LDA     BRKKEY
EAA0  D003 ^EAA5          BNE     SEN2            ;if BREAK key not pressed

                  ;       Process BREAK key.

EAA2  4CC7ED              JMP     PBK             ;process BREAK key, return

                  ;       Process BREAK key not pressed.
```

```
EAA5  A53A         SEN2    LDA     XMTDON           ;transmit-frame done flag
EAA7  F0F5 ^EA9E           BEQ     SEN1             ;if transmit-frame not done

                   ;       Exit.

EAA9  2084EC               JSR     DSR              ;disable SEND and RECEIVE
EAAC  60                   RTS                      ;return



                   **      ORIR - Process Serial Output Ready IRQ
                   *
                   *       ENTRY   JMP     ORIR
                   *
                   *       EXIT
                   *               Exits via RTI
                   *
                   *       MODS
                   *               Original Author Unknown
                   *               1. Bring closer to Coding Standard (object :
                   *                  R. K. Nordin 11/01/83

      = EAAD       ORIR    =       *        ;entry

                   ;       Initialize.

EAAD  98                   TYA
EAAE  48                   PHA              ;save Y
EAAF  E632                 INC     BUFRLO   ;increment low buffer pointer
EAB1  D002 ^EAB5           BNE     ORI1     ;if low buffer pointer non-zero

EAB3  E633                 INC     BUFRHI   ;increment high buffer pointer

                   ;       Check end of buffer.

EAB5  A532         ORI1    LDA     BUFRLO   ;buffer address
EAB7  C534                 CMP     BFENLO   ;buffer end address
EAB9  A533                 LDA     BUFRHI
EABB  E535                 SBC     BFENHI
EABD  901C ^EADB           BCC     ORI4     ;if not past end of buffer

                   ;       Process end of buffer.

EABF  A536                 LDA     CHKSNT   ;checksum sent flag
EAC1  D008 ^EACE           BNE     ORI2     ;if checksum already sent

                   ;       Send checksum.

EAC3  A531                 LDA     CHKSUM   ;checksum
EAC5  8D0DD2               STA     SEROUT   ;serial output register
EAC8  A9FF                 LDA     #$FF
EACA  8538                 STA     CHKSNT   ;indicate checksum sent
EACC  D009 ^EAD7           BNE     ORI3

                   ;       Enable TRANSMIT done interrupt.
```

```
EACE  A510          ORI2     LDA      POKMSK
EAD0  0908                   ORA      #$08
EAD2  8510                   STA      POKMSK
EAD4  8D0ED2                 STA      IRQEN

                    ;        Exit.

EAD7  68            ORI3     PLA
EAD8  A8                     TAY               ;restore Y
EAD9  68                     PLA               ;restore A
EADA  40                     RTI               ;return

                    ;        Transmit next byte from buffer.

EADB  A000          ORI4     LDY      #0
EADD  B132                   LDA      (BUFRLO),Y        ;byte from buffer
EADF  8D0DD2                 STA      SEROUT            ;serial output register
EAE2  18                     CLC
EAE3  6531                   ADC      CHKSUM            ;add byte to checksum
EAE5  6900                   ADC      #0
EAE7  8531                   STA      CHKSUM            ;update checksum
EAE9  4CD7EA                 JMP      ORI3              ;exit


                    **       OCIR - Process Serial Output Complete IRQ
                    *
                    *        ENTRY    JMP      OCIR
                    *
                    *        EXIT
                    *                 Exits via RTI
                    *
                    *        MODS
                    *                 Original Author Unknown
                    *                 1. Bring closer to Coding Standard (object :
                    *                    R. K. Nordin 11/01/83

      = EAEC        OCIR     =        *        ;entry

                    ;        Check checksum sent.

EAEC  A53B                   LDA      CHKSNT  ;checksum sent flag
EAEE  F008 ^EAF8             BEQ      OCI1    ;if checksum not yet sent

                    ;        Process checksum sent.

EAF0  853A                   STA      XMTDON  ;indicate transmit-frame done

                    ;        Disable TRANSMIT done interrupt.

EAF2  A510                   LDA      POKMSK
EAF4  29F7                   AND      #$F7
EAF6  8510                   STA      POKMSK
EAF8  8D0ED2                 STA      IRQEN
```

```
                        ;       Exit.

EAFB  68                OCI1    PLA             ;restore A
EAFC  40                        RTI             ;return



                        **      REC - Receive
                        *
                        *       ENTRY   JSR     REC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = EAFD            REC     =       *       ;entry

                        ;       Initialize.

EAFD  A900                      LDA     #0
EAFF  ACOF03                    LDY     CASFLG
EB02  D002 ^EB06                BNE     REC1    ;if cassette

EB04  8531                      STA     CHKSUM  ;initialize checksum

EB06  8538              REC1    STA     BUFRFL  ;clear buffer full flag
EB08  8539                      STA     RECVDN  ;clear receive-frame done flag
EB0A  A901                      LDA     #SUCCES ;assume success
EB0C  8530                      STA     STATUS  ;status
EB0E  2040EC                    JSR     ESR     ;enable SIO RECEIVE
EB11  A93C                      LDA     #NCOMHI
EB13  8D03D3                    STA     PBCTL

                        ;       Check BREAK key.

EB16  A511              REC2    LDA     BRKKEY
EB18  D003 ^EB1D                BNE     REC3    ;if BREAK key not pressed

                        ;       Process BREAK key.

EB1A  4CC7ED                    JMP     PBK     ;process BREAK key, return

                        ;       Process BREAK key not pressed.

EB1D  AD1703            REC3    LDA     TIMFLG  ;timeout flag
EB20  F005 ^EB27                BEQ     ITO     ;if timeout, indicate timeout

                        ;       Process no timeout.

EB22  A539                      LDA     RECVDN  ;receive-frame done flag
EB24  F0F0 ^EB16                BEQ     REC2    ;if receive-frame done, continue

                        ;       Exit.
```

```
EB26  60                    RTS             ;return



                    **      ITO - Indicate Timeout
                    *
                    *       ENTRY   JSR     ITO
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = EB27        ITO     =       *       ;entry
EB27  A98A                  LDA     #TIMOUT ;timeout indicator
EB29  8530                  STA     STATUS  ;indicate timeout
EB2B  60                    RTS             ;return



                    **      IRIR - Process Serial Input Ready IRQ
                    *
                    *       ENTRY   JMP     IRIR
                    *
                    *       EXIT
                    *               Exits via RTI
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = EB2C        IRIR    =       *       ;entry

                    ;       Initialize.

EB2C  98                    TYA
EB2D  48                    PHA             ;save Y
EB2E  AD0FD2                LDA     SKSTAT
EB31  8D0AD2                STA     SKRES   ;reset status register

                    ;       Check for frame error.

EB34  3004 ^EB3A            BMI     IRI1    ;if no frame error

                    ;       Process frame error.

EB36  A08C                  LDY     #FRMERR ;frame error
EB38  8430                  STY     STATUS  ;indicate frame error

                    ;       Check for overrun error.
```

```
EB3A  2920          IRI1     AND     #$20
EB3C  D004 ^EB42             BNE     IRI2      ;if no overrun error

                    ;        Process overrun error.

EB3E  A08E                   LDY     #OVRRUN ;overrun error
EB40  8430                   STY     STATUS  ;indicate overrun error

                    ;        Check for buffer full.

EB42  A538          IRI2     LDA     BUFRFL
EB44  F013 ^EB59             BEQ     IRI5      ;if buffer not yet full

                    ;        Process buffer full.

EB46  AD0DD2                 LDA     SERIN   ;checksum from device
EB49  C531                   CMP     CHKSUM  ;computed checksum
EB4B  F004 ^EB51             BEQ     IRI3      ;if checksums match

                    ;        Process checksum error.

EB4D  A08F                   LDY     #CHKERR ;checksum error
EB4F  8430                   STY     STATUS  ;indicate checksum error

                    ;        Indicate receive-frame done.

EB51  A9FF          IRI3     LDA     #$FF      ;receive-frame done indicator
EB53  8539                   STA     RECVDN  ;indicate receive-frame done

                    ;        Exit.

EB55  68            IRI4     PLA
EB56  A8                     TAY               ;restore Y
EB57  68                     PLA               ;restore A
EB58  40                     RTI               ;return

                    ;        Process buffer not full.

EB59  AD0DD2        IRI5     LDA     SERIN             ;serial input register
EB5C  A000                   LDY     #0
EB5E  9132                   STA     (BUFRLO),Y        ;byte of buffer
EB60  18                     CLC
EB61  6531                   ADC     CHKSUM            ;add byte to checksum
EB63  6900                   ADC     #0
EB65  8531                   STA     CHKSUM            ;update checksum
EB67  E632                   INC     BUFRLO            ;increment low buffer point;
EB69  D002 ^EB6D             BNE     IRI6              ;if low buffer pointer non-;

EB6B  E633                   INC     BUFRHI            ;increment high buffer poin;

                    ;        Check end of buffer.

EB6D  A532          IRI6     LDA     BUFRLO            ;buffer address
EB6F  C534                   CMP     BFENLO            ;buffer end address
EB71  A533                   LDA     BUFRHI
EB73  E535                   SBC     BFENHI
EB75  900E ^EB55             BCC     IRI4              ;if not past end of buffer
```

```
                        ;       Process end of buffer.

EB77  A53C                      LDA     NOCKSM          ;no checksum follows flag
EB79  F006 ^EB81                BEQ     IRI7            ;if checksum will follow

                        ;       Process no checksum will follow.

EB7B  A900                      LDA     #0
EB7D  853C                      STA     NOCKSM          ;clear no checksum follows :
EB7F  F0D0 ^EB51                BEQ     IRI3            ;indicate receive-frame don:

                        ;       Process checksum will follow.

EB81  A9FF              IRI7    LDA     #$FF
EB83  8538                      STA     BUFRFL          ;indicate buffer full
EB85  D0CE ^EB55                BNE     IRI4            ;exit


                        **      SBP - Set Buffer Pointers
                        *
                        *       ENTRY   JSR     SBP
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = EB87            SBP     =       *       ;entry
EB87  18                        CLC
EB88  AD0403                    LDA     DBUFLO
EB8B  8532                      STA     BUFRLO  ;low buffer address
EB8D  6D0803                    ADC     DBYTLO
EB90  8534                      STA     BFENLO  ;low buffer end address
EB92  AD0503                    LDA     DBUFHI
EB95  8533                      STA     BUFRHI  ;high buffer address
EB97  6D0903                    ADC     DBYTHI
EB9A  8535                      STA     BFENHI  ;high buffer end address
EB9C  60                        RTS             ;return
```

```
                          **      PCI - Process Cassette I/O
                          *
                          *       ENTRY   JSR     PCI
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


       = EB9D             PCI     =       *           ;entry

                          ;       Check command type.

EB9D  AD0303                      LDA     DSTATS           ;command type
EBA0  1032 ^EBD4                  BPL     PCI3             ;if READ

                          ;       Write a record.

EBA2  A9CC                        LDA     #low B00600
EBA4  8D04D2                      STA     AUDF3            ;set 600 baud
EBA7  A905                        LDA     #high B00600
EBA9  8D06D2                      STA     AUDF4
EBAC  2017EC                      JSR     ESS              ;enable SIO SEND
EBAF  A662                        LDX     PALNTS           ;PAL/NTSC offset
EBB1  BC15EE                      LDY     WSIRGX,X         ;low short WRITE IRG time
EBB4  AD0803                      LDA     DAUX2            ;IRG type
EBB7  3003 ^EBBC                  BMI     PCI1             ;if short IRG is desired

EBB9  BC11EE                      LDY     WIRGLX,X         ;low long WRITE IRG time

EBBC  A200                PCI1    LDX     #WIRGHI          ;high IRG time
EBBE  20E2ED                      JSR     SSV              ;set SIO VBLANK parameters
EBC1  A934                        LDA     #MOTRGO
EBC3  8D02D3                      STA     PACTL            ;turn on motor

EBC6  AD1703              PCI2    LDA     TIMFLG           ;timeout flag
EBC9  D0FB ^EBC6                  BNE     PCI2             ;if no timeout

EBCB  2087EB                      JSR     SBP              ;set buffer pointers
EBCE  2088EA                      JSR     SEN              ;send
EBD1  4C04EC                      JMP     PCI6             ;exit

                          ;       Read a record.

EBD4  A9FF                PCI3    LDA     #$FF             ;cassette I/O indicator
EBD6  8D0F03                      STA     CASFLG           ;cassette I/O flag

EBD9  A662                        LDX     PALNTS           ;PAL/NTSC offset
EBDB  BC17EE                      LDY     RSIRGX,X         ;low short READ IRG time
EBDE  AD0803                      LDA     DAUX2            ;IRG type
EBE1  3003 ^EBE6                  BMI     PCI4             ;if short IRG desired

EBE3  BC13EE                      LDY     RIRGLX,X         ;low long READ IRG time

EBE6  A200                PCI4    LDX     #RIRGHI          ;high READ IRG time
EBE8  20E2ED                      JSR     SSV              ;set SIO VBLANK parameters
```

```
EBEB  A934                   LDA     #MOTRGO
EBED  8D02D3                 STA     PACTL           ;turn on motor

EBF0  AD1703      PCI5       LDA     TIMFLG          ;timeout flag
EBF3  D0FB ^EBF0             BNE     PCI5            ;if no timeout

EBF5  2087EB                 JSR     SBP             ;set buffer pointers
EBF8  209AEC                 JSR     GTO             ;get device timeout
EBFB  20E2ED                 JSR     SSV             ;set SIO VBLANK parameters
EBFE  203DED                 JSR     SBR             ;set initial baud rate
EC01  20FDEA                 JSR     REC             ;receive

                  ;          Exit.

EC04  AD0B03      PCI6       LDA     DAUX2           ;IRG type
EC07  3005 ^EC0E             BMI     PCI7            ;if doing short IRG

EC09  A93C                   LDA     #MOTRST
EC0B  8D02D3                 STA     PACTL           ;turn off motor

EC0E  4C2AEA      PCI7       JMP     CSO             ;complete SIO operation, re:


                  **         PTE - Process Timer Expiration
                  *
                  *          ENTRY   JSR     PTE
                  *
                  *          MODS
                  *                  Original Author Unknown
                  *                  1. Bring closer to Coding Standard (object :
                  *                     R. K. Nordin 11/01/83


      = EC11      PTE        =       *       ;entry
EC11  A900                   LDA     #0      ;timeout indicator
EC13  8D1703                 STA     TIMFLG  ;timeout flag
EC16  60                     RTS             ;return



                  **         ESS - Enable SIO SEND
                  *
                  *          ENTRY   JSR     ESS
                  *
                  *          MODS
                  *                  Original Author Unknown
                  *                  1. Bring closer to Coding Standard (object :
                  *                     R. K. Nordin 11/01/83


      = EC17      ESS        =       *       ;entry

                  ;          Initialize.
```

```
EC17  A907                  LDA     #$07     ;mask off previous serial bus contr:
EC19  2D3202                AND     SSKCTL
EC1C  0920                  ORA     #$20     ;set SEND mode

                    ;       Check device type.

EC1E  AC0003                LDY     DDEVIC
EC21  C060                  CPY     #CASET
EC23  D00C ^EC31            BNE     ESS1     ;if not cassette

                    ;       Process cassette.

EC25  0908                  ORA     #$08     ;set FSK output
EC27  A007                  LDY     #LOTONE  ;set FSK tone frequencies
EC29  8C02D2                STY     AUDF2
EC2C  A005                  LDY     #HITONE
EC2E  8C00D2                STY     AUDF1

                    ;       Set serial bus control.

EC31  8D3202        ESS1    STA     SSKCTL   ;SKCTL shadow
EC34  8D0FD2                STA     SKCTL
EC37  A9C7                  LDA     #$C7     ;mask off previous serial bus inter:
EC39  2510                  AND     POKMSK   ;and with POKEY IRQ enable
EC3B  0910                  ORA     #$10     ;enable output data needed interrup:
EC3D  4C56EC                JMP     SSR      ;set for SEND, return



                    **      ESR - Enable SIO RECEIVE
                    *
                    *       ENTRY   JSR     ESR
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = EC40        ESR     =       *        ;entry
EC40  A907                  LDA     #$07     ;mask off previous serial bus contr:
EC42  2D3202                AND     SSKCTL
EC45  0910                  ORA     #$10     ;set receive mode asynchronous
EC47  8D3202                STA     SSKCTL   ;SKCTL shadow
EC4A  8D0FD2                STA     SKCTL
EC4D  8D0AD2                STA     SKRES
EC50  A9C7                  LDA     #$C7     ;mask off previous serial bus inter:
EC52  2510                  AND     POKMSK   ;and with POKEY IRQ enable
EC54  0920                  ORA     #$20     ;enable RECEIVE interrupt
                    ;       JMP     SSR      ;set for RECEIVE, return
```

```
                        **      SSR - Set for SEND or RECEIVE
                        *
                        *       ENTRY   JSR     SSR
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = EC56             SSR     =       *       ;entry

                        ;       Initialize.

EC56  8510                      STA     POKMSK  ;update POKEY IRQ enable
EC58  8D0ED2                    STA     IRQEN   ;IRQ enable
EC5B  A928                      LDA     #$28    ;clock ch. 3 with 1.79 MHz, ch. 4 w:
EC5D  8D08D2                    STA     AUDCTL  ;set audio control

                        ;       Set voice controls.

EC60  A206                      LDX     #6      ;offset to last voice control
EC62  A9A8                      LDA     #$A8    ;pure tone, half volume
EC64  A441                      LDY     SOUNDR  ;noisy I/O flag.
EC66  D002 ^EC6A                BNE     SSR1    ;if noisy I/O desired

EC68  A9A0                      LDA     #$A0    ;pure tone, no volume

EC6A  9D01D2            SSR1    STA     AUDC1,X ;set tone and volume
EC6D  CA                        DEX
EC6E  CA                        DEX
EC6F  10F9 ^EC6A                BPL     SSR1    ;if not done

                        ;       Turn off certain voices.

EC71  A9A0                      LDA     #$A0    ;pure tone, no volume
EC73  8D05D2                    STA     AUDC3   ;turn off sound on voice 3
EC76  AC0003                    LDY     DDEVIC  ;device bus ID
EC79  C060                      CPY     #CASET  ;cassette device ID
EC7B  F006 ^EC83                BEQ     SSR2    ;if cassette device

EC7D  8D01D2                    STA     AUDC1   ;turn off sound on voice 1
EC80  8D03D2                    STA     AUDC2   ;turn off sound on voice 2

EC83  60                SSR2    RTS             ;return
```

```
                        **      DSR - Disable SEND and RECEIVE
                        *
                        *       ENTRY   JSR     DSR
                        *
                        *       NOTES
                        *               Problem: NOP may not be necessary.
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = EC84             DSR     =       *       ;entry

                        ;       Disable serial bus interrupts.

EC84  EA                        NOP
EC85  A9C7                      LDA     #$C7    ;mask to clear serial bus interrupt:
EC87  2510                      AND     POKMSK  ;and with POKEY IRQ enable
EC89  8510                      STA     POKMSK  ;update POKEY IRQ enable
EC8B  8D0ED2                    STA     IRQEN   ;IRQ enable

                        ;       Turn off audio volume.

EC8E  A206                      LDX     #6      ;offset to last voice control
EC90  A900                      LDA     #$00    ;no volume

EC92  9D01D2            DSR1    STA     AUDC1,X ;turn off voice
EC95  CA                        DEX
EC96  CA                        DEX
EC97  10F9 ^EC92                BPL     DSR1    ;if not done

EC99  60                        RTS             ;return



                        **      GTO - Get Device Timeout
                        *
                        *       ENTRY   JSR     GTO
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = EC9A             GTO     =       *       ;entry
EC9A  AD0603                    LDA     DTIMLO  ;device timeout
EC9D  6A                        ROR     A
EC9E  6A                        ROR     A
EC9F  A8                        TAY             ;rotated timeout
ECA0  293F                      AND     #$3F    ;lower 6 bits
ECA2  AA                        TAX             ;high timeout
ECA3  98                        TYA             ;rotated timeout
ECA4  6A                        ROR     A
```

```
ECA5  29C0                AND     #$C0      ;upper 2 bits
ECA7  A8                  TAY               ;low timeout
ECA8  60                  RTS               ;return



                    **      TSIH - Table of SIO Interrupt Handlers
                    *
                    *       NOTES
                    *               Problem: not used.


ECA9  2CEB          TSIH  DW      IRIR      ;serial input ready IRQ
ECAB  ADEA                DW      ORIR      ;serial output ready IRQ
ECAD  ECEA                DW      OCIR      ;serial output complete IRQ



                    **      SID - Send to Intelligent Device
                    *
                    *       ENTRY   JSR     SID
                    *
                    *       NOTES
                    *               Problem: bytes wasted by outer delay loop.
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = ECAF        SID   =       *         ;entry

                    ;     Delay.

ECAF  A201                LDX     #1

ECB1  A0FF          SID1  LDY     #255

ECB3  88            SID2  DEY
ECB4  D0FD ^ECB3          BNE     SID2              ;if inner loop not done

ECB6  CA                  DEX
ECB7  D0F6 ^ECB1          BNE     SID1              ;if outer loop not done

                    ;     Send data frame.

ECB9  2088EA              JSR     SEN               ;send

                    ;     Set timer and wait.

ECBC  A002                LDY     #low CTIM         ;frame acknowledge timeout
ECBE  A200                LDX     #high CTIM
                    ;     JMP     STW               ;set timer and wait, return
```

```
                        **      STW - Set Timer and Wait
                        *
                        *       ENTRY   JSR     STW
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = ECC0            STW     =       *       ;entry
ECC0  20E2ED                    JSR     SSV     ;set SIO VBLANK parameters
ECC3  2037EA                    JSR     WCA     ;wait for completion or ACK
ECC6  98                        TYA             ;wait termination status
ECC7  60                        RTS             ;return



                        **      CBR - Compute Baud Rate
                        *
                        *       CBR computes value for POKEY frequency for the baud:
                        *       measured by an interval of the VCOUNT timer.
                        *
                        *       ENTRY   JSR     CBR
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = ECC8            CBR     =       *               ;entry
ECC8  8D1003                    STA     TIMER2          ;save final timer value
ECCB  8C1103                    STY     TIMER2+1
ECCE  202EED                    JSR     AVV             ;adjust VCOUNT value
ECD1  8D1003                    STA     TIMER2          ;save adjusted timer 2 valu:
ECD4  AD0C03                    LDA     TIMER1
ECD7  202EED                    JSR     AVV             ;adjust VCOUNT value
ECDA  8D0C03                    STA     TIMER1          ;save adjusted timer 1 valu:
ECDD  AD1003                    LDA     TIMER2
ECE0  38                        SEC
ECE1  ED0C03                    SBC     TIMER1
ECE4  8D1203                    STA     TEMP1           ;save difference
ECE7  AD1103                    LDA     TIMER2+1
ECEA  38                        SEC
ECEB  ED0D03                    SBC     TIMER1+1
ECEE  A8                        TAY                     ;difference
ECEF  A662                      LDX     PALNTS
ECF1  A900                      LDA     #0
ECF3  38                        SEC
ECF4  FD19EE                    SBC     CONS1X,X

ECF7  18                CBR1    CLC
ECF8  7D19EE                    ADC     CONS1X,X        ;accumulate product
ECFB  88                        DEY
ECFC  10F9 ^ECF7                BPL     CBR1            ;if not done
```

```
ECFE  18                  CLC
ECFF  6D1203              ADC     TEMP1           ;add to get total VCOUNT di:
ED02  A8                  TAY                     ;total VCOUNT difference
ED03  4A                  LSR     A
ED04  4A                  LSR     A
ED05  4A                  LSR     A
ED06  0A                  ASL     A               ;interval divided by 4
ED07  38                  SEC
ED08  E916                SBC     #22             ;adjust offset
ED0A  AA                  TAX                     ;offset
ED0B  98                  TYA                     ;total VCOUNT difference
ED0C  2907                AND     #7              ;extract lower 3 bits of in:
ED0E  A8                  TAY                     ;lower 3 bits of interval
ED0F  A9F5                LDA     #-11

ED11  18           CBR2   CLC
ED12  690B                ADC     #11             ;accumulate interpolation c:
ED14  88                  DEY
ED15  10FA ^ED11          BPL     CBR2            ;if done

ED17  A000                LDY     #0              ;assume no addition correct:
ED19  38                  SEC
ED1A  E907                SBC     #7              ;adjust interpolation const:
ED1C  1001 ^ED1F          BPL     CBR3

ED1E  88                  DEY                     ;indicate addition correcti:

ED1F  18           CBR3   CLC
ED20  7DF9ED              ADC     TPFV,X          ;add constant to table valu:
ED23  8DEE02              STA     CBAUDL          ;low POKEY frequency value
ED26  98                  TYA
ED27  7DFAED              ADC     TPFV+1,X
ED2A  8DEF02              STA     CBAUDH          ;high POKEY frequency value
ED2D  60                  RTS                     ;return



                   **     AVV - Adjust VCOUNT Value
                   *
                   *      ENTRY   JSR     AVV
                   *
                   *      MODS
                   *              Original Author Unknown
                   *              1. Bring closer to Coding Standard (object :
                   *                 R. K. Nordin 11/01/83


      = ED2E       AVV    =       *       ;entry
ED2E  C97C                CMP     #$7C
ED30  3004 ^ED36          BMI     AVV1    ;if >= $7C

ED32  38                  SEC
ED33  E97C                SBC     #$7C
ED35  60                  RTS             ;return
```

```
ED36  18          AVV1    CLC
ED37  A662                LDX     PALNTS
ED39  7D18EE              ADC     CONS2X,X
ED3C  60                  RTS             ;return



                  **      SBR - Set Initial Baud Rate
                  *
                  *       INITIAL BAUD RATE MEASUREMENT -- USED TO SET THE
                  *       BAUD RATE AT THE START OF A RECORD.
                  *
                  *       IT IS ASSUMED THAT THE FIRST TWO BYTES OF EVERY
                  *       RECORD ARE $AA.
                  *
                  *       ENTRY   JSR     SBR
                  *
                  *       NOTES
                  *               Problem: bytes wasted by branch around bran;
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


      = ED3D      SBR     =       *               ;entry

ED3D  A511        SBR1    LDA     BRKKEY
ED3F  D003 ^ED44          BNE     SBR2            ;if BREAK key not pressed

ED41  4CC7ED              JMP     PBK             ;process BREAK key, return

ED44  78          SBR2    SEI
ED45  AD1703              LDA     TIMFLG          ;timeout flag
ED48  D002 ^ED4C          BNE     SBR3            ;if no timeout

ED4A  F025 ^ED71          BEQ     SBR5            ;process timeout

ED4C  AD0FD2      SBR3    LDA     SKSTAT
ED4F  2910                AND     #$10            ;extract start bit
ED51  D0EA ^ED3D          BNE     SBR1            ;if start bit

ED53  8D1603              STA     SAVIO           ;save serial data in
ED56  AE0BD4              LDX     VCOUNT          ;vertical line counter
ED59  A414                LDY     RTCLOK+2        ;low byte of VBLANK clock
ED5B  8E0C03              STX     TIMER1
ED5E  8C0D03              STY     TIMER1+1        ;save initial timer value
ED61  A201                LDX     #1
ED63  8E1503              STX     TEMP3           ;set mode flag
ED66  A00A                LDY     #10             ;10 bits

ED68  A511        SBR4    LDA     BRKKEY
ED6A  F05B ^EDC7          BEQ     PBK             ;if BREAK key pressed, proc;

ED6C  AD1703              LDA     TIMFLG          ;timeout flag
```

```
ED6F  D004 ^ED75          BNE    SBR6          ;if no timeout

ED71  58          SBR5    CLI
ED72  4C27EB              JMP    ITO           ;indicate timeout, return

ED75  AD0FD2      SBR6    LDA    SKSTAT
ED78  2910                AND    #$10          ;extract
ED7A  CD1603              CMP    SAVIO         ;previous serial data in
ED7D  F0E9 ^ED68          BEQ    SBR4          ;if data in not changed

ED7F  8D1603              STA    SAVIO         ;save serial data in
ED82  88                  DEY                  ;decrement bit counter
ED83  D0E3 ^ED68          BNE    SBR4          ;if not done

ED85  CE1503              DEC    TEMP3         ;decrement mode
ED88  300C ^ED96          BMI    SBR7          ;if done with both modes

ED8A  AD0BD4              LDA    VCOUNT
ED8D  A414                LDY    RTCLOK+2
ED8F  20C8EC              JSR    CBR           ;compute baud rate
ED92  A009                LDY    #9            ;9 bits
ED94  D0D2 ^ED68          BNE    SBR4          ;set bit counter

ED96  ADEE02      SBR7    LDA    CBAUDL
ED99  8D04D2              STA    AUDF3
ED9C  ADEF02              LDA    CBAUDH
ED9F  8D06D2              STA    AUDF4         ;set POKEY baud rate
EDA2  A900                LDA    #0
EDA4  8D0FD2              STA    SKSTAT
EDA7  AD3202              LDA    SSKCTL
EDAA  8D0FD2              STA    SKSTAT        ;initialize POKEY serial po:
EDAD  A955                LDA    #$55
EDAF  9132                STA    (BUFRLO),Y    ;first byte of buffer
EDB1  C8                  INY
EDB2  9132                STA    (BUFRLO),Y    ;second byte of buffer
EDB4  A9AA                LDA    #$AA          ;checksum
EDB6  8531                STA    CHKSUM        ;checksum
EDB8  18                  CLC
EDB9  A532                LDA    BUFRLO
EDBB  6902                ADC    #2            ;add 2
EDBD  8532                STA    BUFRLO        ;update low buffer pointer
EDBF  A533                LDA    BUFRHI
EDC1  6900                ADC    #0
EDC3  8533                STA    BUFRHI        ;update high buffer pointer
EDC5  58                  CLI
EDC6  60                  RTS                  ;return
```

```
                       **      PBK - Process BREAK Key
                       *
                       *       ENTRY   JSR     PBK
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83


      = EDC7           PBK     =       *       ;entry
EDC7  2084EC                   JSR     DSR     ;disable SEND and RECEIVE
EDCA  A93C                     LDA     #MOTRST
EDCC  8D0203                   STA     PACTL   ;turn off motor
EDCF  A93C                     LDA     #NCOMHI
EDD1  8D03D3                   STA     PBCTL   ;raise NOT COMMAND line
EDD4  A980                     LDA     #BRKABT ;BREAK abort error
EDD6  8530                     STA     STATUS  ;status
EDD8  AE1803                   LDX     STACKP  ;saved stack pointer
EDDB  9A                       TXS             ;restore stack pointer
EDDC  C611                     DEC     BRKKEY  ;indicate BREAK
EDDE  58                       CLI
EDDF  4C2AEA                   JMP     CSO     ;complete SIO operation, return to :



                       **      SSV - Set SIO VBLANK Parameters
                       *
                       *       ENTRY   JSR     SSV
                       *
                       *       MODS
                       *               Original Author Unknown
                       *               1. Bring closer to Coding Standard (object :
                       *                  R. K. Nordin 11/01/83


      = EDE2           SSV     =       *               ;entry
EDE2  A911                     LDA     #low PTE        ;timer expiration routine a:
EDE4  8D2602                   STA     CDTMA1
EDE7  A9EC                     LDA     #high PTE
EDE9  8D2702                   STA     CDTMA1+1
EDEC  A901                     LDA     #1              ;timer 1
EDEE  78                       SEI
EDEF  205CE4                   JSR     SETVBV          ;set VBLANK parameters
EDF2  A901                     LDA     #1              ;no timeout indicator
EDF4  8D1703                   STA     TIMFLG          ;timeout flag
EDF7  58                       CLI
EDF8  60                       RTS                     ;return
```

```
                    **      TPFV - Table of POKEY Frequency Values
                    *
                    *       TPFV translates VCOUNT interval timer measurements :
                    *       frequency register values.
                    *
                    *       Table entries are AUDF+7.
                    *
                    *       Frequency-out is Frequency-in divided by 2*(AUDF+M):
                    *       Frequency-in = 1.78979 Mhz and M = 7.
                    *
                    *       AUDF+7=(11.365167)*T-out, where T-out is the number:
                    *       (127 used cd soulution) of VCOUNT for one character
                    *       time (10 bit times).


                    ;       DW      636     ;baud rate 1407, VCOUNT interval 56
                    ;       DW      727     ;baud rate 1231, VCOUNT interval 64
                    ;       DW      818     ;baud rate 1094, VCOUNT interval 72
                    ;       DW      909     ;baud rate 985, VCOUNT interval 80

EDF9  E803          TPFV    DW      1000    ;baud rate 895, VCOUNT interval 88
EDFB  4304                  DW      1091    ;baud rate 820, VCOUNT interval 96
EDFD  9E04                  DW      1182    ;baud rate 757, VCOUNT interval 104
EDFF  F904                  DW      1273    ;baud rate 703, VCOUNT interval 112
EE01  5405                  DW      1364    ;baud rate 656, VCOUNT interval 120
EE03  AF05                  DW      1455    ;baud rate 615, VCOUNT interval 128
EE05  0A06                  DW      1546    ;baud rate 579, VCOUNT interval 136
EE07  6506                  DW      1637    ;baud rate 547, VCOUNT interval 144
EE09  C006                  DW      1728    ;baud rate 518, VCOUNT interval 152
EE0B  1A07                  DW      1818    ;baud rate 492, VCOUNT interval 160
EE0D  7507                  DW      1909    ;baud rate 469, VCOUNT interval 168
EE0F  D007                  DW      2000    ;baud rate 447, VCOUNT interval 176

                    ;       DW      2091    ;baud rate 428, VCOUNT interval 184
                    ;       DW      2182    ;baud rate 410, VCOUNT interval 192
                    ;       DW      2273    ;baud rate 394, VCOUNT interval 200
                    ;       DW      2364    ;baud rate 379, VCOUNT interval 208
                    ;       DW      2455    ;baud rate 365, VCOUNT interval 216
                    ;       DW      2546    ;baud rate 352, VCOUNT interval 224
                    ;       DW      2637    ;baud rate 339, VCOUNT interval 232
                    ;       DW      2728    ;baud rate 328, VCOUNT interval 240
                    ;       DW      2819    ;baud rate 318, VCOUNT interval 248
```

```
                        **      NTSC/PAL Constant Tables

EE11  84          WIRGLX  DB      low WIRGLN       ;NTSC low long write IRG
EE12  96                  DB      low WIRGLP       ;PAL low long write IRG

EE13  78          RIRGLX  DB      low RIRGLN       ;NTSC low long read IRG
EE14  64                  DB      low RIRGLP       ;PAL low long read IRG

EE15  0F          WSIRGX  DB      low WSIRGN       ;NTSC low short write IRG
EE16  0D                  DB      low WSIRGP       ;PAL low short write IRG

EE17  0A          RSIRGX  DB      low RSIRGN       ;NTSC low short read IRG
EE18  08                  DB      low RSIRGP       ;PAL low short read IRG

EE19  83          CONS1X  DB      131              ;NTSC
EE1A  9C                  DB      156              ;PAL

EE1B  07          CONS2X  DB      7                ;NTSC
EE1C  20                  DB      32               ;PAL
```

```
EE1D                 **      TSMA - Table of Screen Memory Allocation
                     *
                     *       Entry n is the number of $40-byte blocks to allocat:
                     *       graphics mode n.
                     *
                     *       NOTES
                     *               Problem: For readability, this, and other t:
                     *               this area, could be moved closer to the oth:
                     *               the Keyboard, Editor and Screen Handler (ju:
                     *               the EF6B patch).


EE1D  18             TSMA    DB      24      ;0
EE1E  10                     DB      16      ;1
EE1F  0A                     DB      10      ;2
EE20  0A                     DB      10      ;3
EE21  10                     DB      16      ;4
EE22  1C                     DB      28      ;5
EE23  34                     DB      52      ;6
EE24  64                     DB      100     ;7
EE25  C4                     DB      196     ;8
EE26  C4                     DB      196     ;9
EE27  C4                     DB      196     ;10
EE28  C4                     DB      196     ;11
EE29  1C                     DB      28      ;12
EE2A  10                     DB      16      ;13
EE2B  64                     DB      100     ;14
EE2C  C4                     DB      196     ;15


                     **      TDLE - Table of Display List Entry Counts
                     *
                     *       Each entry is 2 bytes.


EE2D  1717           TDLE    DB      23,23   ;0
EE2F  0B17                   DB      11,23   ;1
EE31  2F2F                   DB      47,47   ;2
EE33  5F5F                   DB      95,95   ;3
EE35  6161                   DB      97,97   ;4
EE37  6161                   DB      97,97   ;5
EE39  170B                   DB      23,11   ;6
EE3B  BF61                   DB      191,97  ;7
EE3D  1313                   DB      19,19   ;8
EE3F  0913                   DB      9,19    ;9
EE41  2727                   DB      39,39   ;10
EE43  4F4F                   DB      79,79   ;11
EE45  4141                   DB      65,65   ;12
EE47  4141                   DB      65,65   ;13
EE49  1309                   DB      19,9    ;14
EE4B  9F41                   DB      159,65  ;15
```

```
                        **      TAGM - Table of ANTIC Graphics Modes
                        *
                        *       Entry n is the ANTIC graphics mode corresponding to:
                        *       graphics mode n.

EE4D  02                TAGM    DB      $02     ;internal 0 - 40x2x8 characters
EE4E  06                        DB      $06     ;internal 1 - 20x5x8 characters
EE4F  07                        DB      $07     ;internal 2 - 20x5x16 characters
EE50  08                        DB      $08     ;internal 3 - 40x4x8 graphics
EE51  09                        DB      $09     ;internal 4 - 80x2x4 graphics
EE52  0A                        DB      $0A     ;internal 5 - 80x4x4 graphics
EE53  0B                        DB      $0B     ;internal 6 - 160x2x2 graphics
EE54  0D                        DB      $0D     ;internal 7 - 160x4x2 graphics
EE55  0F                        DB      $0F     ;internal 8 - 320x2x1 graphics
EE56  0F                        DB      $0F     ;internal 9 - 320x2x1 GTIA "lum" mo:
EE57  0F                        DB      $0F     ;internal 10 - 320x2x1 GTIA "color/:
EE58  0F                        DB      $0F     ;internal 11 - 320x2x1 GTIA "color":
EE59  04                        DB      $04     ;internal 12 - 40x5x8 characters
EE5A  05                        DB      $05     ;internal 13 - 40x5x16 characters
EE5B  0C                        DB      $0C     ;internal 14 - 160x2x1 graphics
EE5C  0E                        DB      $0E     ;internal 15 - 160x4x1 graphics


                        **      TDLV - Table of Display List Vulnerability
                        *
                        *       Entry n is non-zero if the display list for mode n :
                        *       cross a page boundary.

EE5D  00                TDLV    DB      0       ;0
EE5E  00                        DB      0       ;1
EE5F  00                        DB      0       ;2
EE60  00                        DB      0       ;3
EE61  00                        DB      0       ;4
EE62  00                        DB      0       ;5
EE63  00                        DB      0       ;6
EE64  01                        DB      1       ;7
EE65  01                        DB      1       ;8
EE66  01                        DB      1       ;9
EE67  01                        DB      1       ;10
EE68  01                        DB      1       ;11
EE69  00                        DB      0       ;12
EE6A  00                        DB      0       ;13
EE6B  01                        DB      1       ;14
EE6C  01                        DB      1       ;15
```

```
                        **      TLSC - Table of Left Shift Counts
                        *
                        *       Entry n is the NUMBER OF LEFT SHIFTS NEEDED TO MULT:
                        *       COLCRS BY # BYTES/ROW ((ROWCRS*5)/(2**TLSC)) for mo:


EE6D  03                TLSC    DB      3       ;0
EE6E  02                        DB      2       ;1
EE6F  02                        DB      2       ;2
EE70  01                        DB      1       ;3
EE71  01                        DB      1       ;4
EE72  02                        DB      2       ;5
EE73  02                        DB      2       ;6
EE74  03                        DB      3       ;7
EE75  03                        DB      3       ;8
EE76  03                        DB      3       ;9
EE77  03                        DB      3       ;10
EE78  03                        DB      3       ;11
EE79  03                        DB      3       ;12
EE7A  03                        DB      3       ;13
EE7B  02                        DB      2       ;14
EE7C  03                        DB      3       ;15


                        **      TMCC - Table of Mode Column Counts
                        *
                        *       Entry n is the low column count for mode n.


EE7D  28                TMCC    DB      low 40          ;0
EE7E  14                        DB      low 20          ;1
EE7F  14                        DB      low 20          ;2
EE80  28                        DB      low 40          ;3
EE81  50                        DB      low 80          ;4
EE82  50                        DB      low 80          ;5
EE83  A0                        DB      low 160         ;6
EE84  A0                        DB      low 160         ;7
EE85  40                        DB      low 320         ;8
EE86  50                        DB      low 80          ;9
EE87  50                        DB      low 80          ;10
EE88  50                        DB      low 80          ;11
EE89  28                        DB      low 40          ;12
EE8A  28                        DB      low 40          ;13
EE8B  A0                        DB      low 160         ;14
EE8C  A0                        DB      low 160         ;15
```

```
                        **      TMRC - Table of Mode Row Counts
                        *
                        *       Entry n is the row count for mode n.


EE8D  18                TMRC    DB      24      ;0
EE8E  18                        DB      24      ;1
EE8F  0C                        DB      12      ;2
EE90  18                        DB      24      ;3
EE91  30                        DB      48      ;4
EE92  30                        DB      48      ;5
EE93  60                        DB      96      ;6
EE94  60                        DB      96      ;7
EE95  C0                        DB      192     ;8
EE96  C0                        DB      192     ;9
EE97  C0                        DB      192     ;10
EE98  C0                        DB      192     ;11
EE99  18                        DB      24      ;12
EE9A  0C                        DB      12      ;13
EE9B  C0                        DB      192     ;14
EE9C  C0                        DB      192     ;15



                        **      TRSC - Table of Right Shift Counts
                        *
                        *       Entry n is HOW MANY RIGHT SHIFTS FOR HCRSR FOR PART;
                        *       BYTE MODES for mode n.


EE9D  00                TRSC    DB      0       ;0
EE9E  00                        DB      0       ;1
EE9F  00                        DB      0       ;2
EEA0  02                        DB      2       ;3
EEA1  03                        DB      3       ;4
EEA2  02                        DB      2       ;5
EEA3  03                        DB      3       ;6
EEA4  02                        DB      2       ;7
EEA5  03                        DB      3       ;8
EEA6  01                        DB      1       ;9
EEA7  01                        DB      1       ;10
EEA8  01                        DB      1       ;11
EEA9  00                        DB      0       ;12
EEAA  00                        DB      0       ;13
EEAB  03                        DB      3       ;14
EEAC  02                        DB      2       ;15
```

```
                        **      TDSM - Table of Display Masks
                        *
                        *       NOTES
                        *               Includes TBTM - Table of Bit Masks.


EEAD  FF                TDSM    DB      $FF     ;1
EEAE  F0                        DB      $F0     ;2
EEAF  0F                        DB      $0F     ;3
EEB0  C0                        DB      $C0     ;4
EEB1  30                        DB      $30     ;5
EEB2  0C                        DB      $0C     ;6
EEB3  03                        DB      $03     ;7

EEB4  80                TBTM    DB      $80     ;8 (0)
EEB5  40                        DB      $40     ;9 (1)
EEB6  20                        DB      $20     ;10 (2)
EEB7  10                        DB      $10     ;11 (3)
EEB8  08                        DB      $08     ;12 (4)
EEB9  04                        DB      $04     ;13 (5)
EEBA  02                        DB      $02     ;14 (6)
EEBB  01                        DB      $01     ;15 (7)
```

```
EEBC                 **      PHE - Perform Peripheral Handler Entry
                     *
                     *       PHE attempts to enter a peripheral handler in the h:
                     *
                     *       ENTRY   JSR     PHE
                     *               X = device code
                     *               A = high linkage table address
                     *               Y = low linkage table address
                     *
                     *       EXIT
                     *               Success:
                     *               C clear
                     *               Handler table entry made
                     *
                     *               Failure due to entry previously made:
                     *               C set
                     *               N clear
                     *               X = offset to second byte of duplicate entr:
                     *               A, Y unchanged
                     *
                     *               Failure due to handler table full:
                     *               C set
                     *               N set
                     *
                     *       CHANGES
                     *               A X Y
                     *
                     *       CALLS
                     *               -none-
                     *
                     *       MODS
                     *               R. S. Scheiman  04/01/82
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = EEBC         PHE     =       *       ;entry

                     ;       Initialize.

EEBC  48                     PHA             ;save high linkage table address
EEBD  98                     TYA
EEBE  48                     PHA             ;save low linkage table address

                     ;       Search for device code in handler table.

EEBF  8A                     TXA                     ;device code
EEC0  A200                   LDX     #0              ;offset to first entry of t:

EEC2  DD1A03         PHE1    CMP     HATABS,X        ;device code from table
EEC5  F01E ^EEE5             BEQ     PHE3            ;if device code found

EEC7  E8                     INX
EEC8  E8                     INX
EEC9  E8                     INX
EECA  E022                   CPX     #MAXDEV+1       ;offset+1 of last possible :
EECC  30F4 ^EEC2             BMI     PHE1            ;if not done
```

```
                          ;       Search for empty entry in handler table.

EECE  A200                        LDX     #0                  ;offset to first entry of t:
EED0  A8                          TAY                         ;save device code
EED1  A900                        LDA     #0

EED3  DD1A03        PHE2          CMP     HATABS,X            ;device code from table
EED6  F013 ^EEEB                  BEQ     PHE4                ;if empty entry found

EED8  E8                          INX
EED9  E8                          INX
EEDA  E8                          INX
EEDB  E022                        CPX     #MAXDEV+1           ;offset+1 of last possible :
EEDD  30F4 ^EED3                  BMI     PHE2                ;if not done

                          ;       Return table full condition.

EEDF  68                          PLA             ;clean stack
EEE0  68                          PLA
EEE1  A0FF                        LDY     #$FF    ;indicate table full (set N)
EEE3  38                          SEC             ;indicate failure
EEE4  60                          RTS             ;return

                          ;       Return device code found condition.

EEE5  68            PHE3          PLA             ;saved Y
EEE6  A8                          TAY             ;restore Y
EEE7  68                          PLA             ;restore A
EEE8  E8                          INX             ;indicate device code found (clear :
EEE9  38                          SEC             ;indicate failure
EEEA  60                          RTS             ;return

                          ;       Enter handler in table.

EEEB  98            PHE4          TYA                         ;device code
EEEC  9D1A03                      STA     HATABS,X            ;enter device code
EEEF  68                          PLA                         ;saved low linkage table ad:
EEF0  9D1B03                      STA     HATABS+1,X          ;low address
EEF3  68                          PLA                         ;saved high linkage table a:
EEF4  9D1C03                      STA     HATABS+2,X          ;high address

                          ;       Return success condition.

EEF7  18                          CLC             ;indicate success
EEF8  60                          RTS             ;return
```

```
                         **      PHO - Perform Peripheral Handler Poll at OPEN
                         *
                         *       Subroutine to perform Type 4 Poll at OPEN time, and
                         *       "provisionally" open IOCB if peripheral answers.
                         *
                         *       Input parameters:
                         *       ICIDNO identifies calling IOCB;
                         *       From zero-page IOCB:
                         *               ICBALZ,ICBAHZ (buffer pointer)
                         *               ICDNOZ (device number from caller's filespe:
                         *       From caller's buffer: device name (in filespec.).
                         *
                         *       Output parameters:
                         *       "No device" error returned if Poll not answered.
                         *       If poll is answered, the calling IOCB is "Provision:
                         *               opened (and successful status is returned)-:
                         *               ICHIDZ set to mark provisional open
                         *               ICPTLZ,ICPTHZ points to PTL (special PUT-BY:
                         *               ICSPR in calling IOCB set to device name (f:
                         *               ICSPR+1 in calling IOCB set to device seria:
                         *
                         *       Modified:
                         *       Registers not saved.
                         *
                         *       Subroutines called:
                         *       PHP performs poll.
                         *
                         *       ENTRY   JSR     PHO
                         *
                         *       NOTES
                         *               Problem: in the CRASS65 version, ICIDNO was:
                         *               zero-page.
                         *
                         *       MODS
                         *               R. S. Scheiman  04/01/82
                         *               1. Bring closer to Coding Standard (object :
                         *                  R. K. Nordin 11/01/83


       = EEF9    PHO     =       *               ;entry
EEF9   A000              LDY     #0              ;Call for Type 4 Poll with
EEFB   B124              LDA     (ICBALZ),Y      ;device name from user
EEFD   A421              LDY     ICDNOZ          ;OPEN
EEFF   208EE7            JSR     PHP
EF02   1003 ^EF07        BPL     PHO1            ;if poll answered

EF04   A082              LDY     #NONDEV         ;Return "no device" error
EF06   60                RTS                     ;return

EF07   A97F      PHO1    LDA     #$7F            ;"Provisionally" OPEN the I:
EF09   8520              STA     ICHIDZ          ;(Mark "provisional")
EF0B   A925              LDA     #low [PTL-1]
EF0D   8526              STA     ICPTLZ          ;(Special put byte routine :
EF0F   A9EF              LDA     #high [PTL-1]
EF11   8527              STA     ICPTHZ
EF13   ADEC02            LDA     DVSTAT+2        ;(Peripheral address for lo:
                 ;       LDX     ICIDNO
```

```
EF16  AE2E00           VFD     8\$AE,8\low ICIDNO,8\high ICIDNO
EF19  9D4D03           STA     ICSPR+1,X
EF1C  A000             LDY     #0
EF1E  B124             LDA     (ICBALZ),Y      ;(Device name from user)
EF20  9D4C03           STA     ICSPR,X
EF23  A001             LDY     #SUCCES         ;indicate success
EF25  60               RTS                     ;return


                **      PTL - Perform PUT-BYTE for Provisionally Open IOCB
                *
                *       Put byte entry for provisionally opened IOCB's.
                *       This routine performs load, relocation, initializat:
                *       and finishes OPEN, then calls handler's put byte en:
                *
                *       Input parameters:
                *       A       Byte to output;
                *       X       IOCB index (IOCB number times 16);
                *       Y       "Function not supported" error code $92.
                *       AUX1 and AUX2 in zero-page IOCB are copied from the:
                *               IOCB prior to the call to PTL.
                *
                *       Output parameters:
                *       Various errors may be returned if loading fails (ei:
                *               did not allow loading by setting HNDLOD fla:
                *               was a loading error or calling error);
                *       If no loading error, this routine returns nothing--:
                *               returned is returned by the loaded PUT-BYTE:
                *               is called by this routine after the handler:
                *               initialized, and opened.
                *
                *       Modified:
                *       ICIDNO (a CIO variable);
                *       all of the zero-page IOCB is copied from the callin:
                *       normal CIO open-operation variables are affected;
                *       after opening, the zero-page IOCB is copied to the :
                *       Registers not saved if error return;if handler is l:
                *               and opened properly, the caller's A and X r:
                *               passed to the loaded handler's PUT-BYTE rou:
                *               Y is passed to that routine as $92)--then r:
                *               on return is up to handler PUT-BYTE since i:
                *               directly to caller.
                *
                *       Subroutines called:
                *       PHL (does loading, initializing and opening--calls :
                *       loaded handler's INIT, OPEN, and PUT-BYTE entries a:
                *       The PUT-BYTE entry returns directly to the PTL call:
                *
                *       ENTRY   JSR     PTL
                *
                *       NOTES
                *               Problem: in the CRASS65 version, ICIDNO was:
                *               zero-page.
                *
                *       MODS
```

```
                        *               R. S. Scheiman  04/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = EF26            PTL     =       *               ;entry
EF26  48                        PHA                     ;save byte to output
EF27  8A                        TXA                     ;IOCB index
EF28  48                        PHA                     ;save IOCB index
EF29  290F                      AND     #$0F            ;IOCB index modulo 16
EF2B  D010 ^EF3D                BNE     PTL2            ;if IOCB not divisble by 16, error

EF2D  E080                      CPX     #MAXIOC
EF2F  100C ^EF3D                BPL     PTL2            ;if IOCB index invalid

EF31  ADE902                    LDA     HNDLOD
EF34  D008 ^EF41                BNE     PTL3            ;if user wants loading

EF36  A082                      LDY     #NONDEV ;indicate nonexistent device error

                        ;       Return error.

EF38  68                PTL1    PLA                     ;clean stack
EF39  68                        PLA
EF3A  C000                      CPY     #0              ;indicate failure (set N)
EF3C  60                        RTS                     ;return

EF3D  A086              PTL2    LDY     #BADIOC ;indicate bad IOCB number error
EF3F  30F7 ^EF38                BMI     PTL1            ;return error

                        ;       Simulate beginning of CIO, since CIO bypassed.

EF41                    PTL3
                        ;       STX     ICIDNO  ;IOCB index
EF41  8E2E00                    VFD     8\$8E,8\low ICIDNO,8\high ICIDNO
EF44  A000                      LDY     #0              ;offset to first byte of page zero :

                        ;       Copy IOCB to page zero IOCB.

EF46  BD4003            PTL4    LDA     IOCB,X  ;byte of IOCB
EF49  992000                    STA     ZIOCB,Y ;byte of page zero IOCB
EF4C  E8                        INX
EF4D  C8                        INY
EF4E  C00C                      CPY     #12
EF50  30F4 ^EF46                BMI     PTL4            ;if not done

EF52  2029CA                    JSR     PHL             ;load and initialize peripheral han:
EF55  30E1 ^EF38                BMI     PTL1            ;if error

EF57  68                        PLA                     ;Re-do the put byte call,
EF58  AA                        TAX                     ;this time calling real handler...
EF59  68                        PLA
EF5A  A8                        TAY
EF5B  A527                      LDA     ICPTHZ
EF5D  48                        PHA                     ;(Put byte entry address minus one)
EF5E  A526                      LDA     ICPTLZ
EF60  48                        PHA
```

```
EF61  98                    TYA
EF62  A092                  LDY     #FNCNOT
EF64  60                    RTS             ;invoke handler (address on stack)
```

```
EF6B                    FIX     $EF6B


                  **    EF6B - $EF6B Patch
                  *
                  *     For compatibility with OS Revision B, initiate cass:

EF6B  4C05FD            JMP     ICR     ;initiate cassette READ, return
```

```
EF6E                **      SIN - Initialize Screen
                    *
                    *       ENTRY   JSR     SIN
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = EF6E        SIN     =       *               ;entry

EF6E  A9FF                  LDA     #$FF            ;clear code indicator
EF70  8DFC02                STA     CH              ;key code

EF73  ADE402                LDA     RAMSIZ          ;size of RAM
EF76  856A                  STA     RAMTOP          ;RAM size

EF78  A940                  LDA     #$40            ;CAPS lock indicator
EF7A  8DBE02                STA     SHFLOK          ;shift/control lock flags

EF7D  A951                  LDA     #low TCKD       ;table of character key def:
EF7F  8579                  STA     KEYDEF          ;key definition table addre:
EF81  A9FB                  LDA     #high TCKD
EF83  857A                  STA     KEYDEF+1

EF85  A911                  LDA     #low TFKD       ;table of function key defi:
EF87  8560                  STA     FKDEF           ;function key definition ta:
EF89  A9FC                  LDA     #high TFKD
EF8B  8561                  STA     FKDEF+1

EF8D  60                    RTS                     ;return


                    **      SOP - Perform Screen OPEN
                    *
                    *       ENTRY   JSR     SOP
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = EF8E        SOP     =       *       ;entry

                    ;       Check mode.

EF8E  A52B                  LDA     ICAX2Z
EF90  290F                  AND     #$0F
EF92  D008 ^EF9C            BNE     COC     ;if not mode 0, complete OPEN comma:

                    ;       Process mode 0.

                    ;       JMP     EOP     ;perform editor OPEN, return
```

```
                         **     EOP - Perform Editor OPEN
                         *
                         *      ENTRY   JSR     EOP
                         *
                         *      MODS
                         *              Original Author Unknown
                         *              1. Bring closer to Coding Standard (object :
                         *                 R. K. Nordin 11/01/83


      = EF94       EOP          =       *       ;entry
EF94  A52A                      LDA     ICAX1Z
EF96  290F                      AND     #$0F
EF98  852A                      STA     ICAX1Z
EF9A  A900                      LDA     #0
                         ;      JMP     COC     ;complete OPEN command, return



                         **     COC - Complete OPEN Command
                         *
                         *      ENTRY   JSR     COC
                         *              A = mode
                         *
                         *      MODS
                         *              Original Author Unknown
                         *              1. Bring closer to Coding Standard (object :
                         *                 R. K. Nordin 11/01/83


      = EF9C       COC          =       *       ;entry

                         ;      Check mode.

EF9C  8557                      STA     DINDEX  ;save mode
EF9E  C910                      CMP     #16
EFA0  9005 ^EFA7                BCC     COC1    ;if mode within range

                         ;      Process invalid mode.

EFA2  A991                      LDA     #BADMOD
EFA4  4C54F1                    JMP     COC17

                         ;      Initialize for OPEN.

EFA7  A9E0         COC1         LDA     #high DCSORG    ;high domestic character se:
EFA9  8DF402                    STA     CHBAS           ;character set base
EFAC  A9CC                      LDA     #high ICSORG    ;high international charact:
EFAE  8D6B02                    STA     CHSALT          ;alternate character set ba:
EFB1  A902                      LDA     #2
EFB3  8DF302                    STA     CHACT
EFB6  8D2F02                    STA     SDMCTL          ;turn off DMA
EFB9  A901                      LDA     #SUCCES
EFBB  854C                      STA     DSTAT           ;clear status
EFBD  A9C0                      LDA     #$C0            ;enable IRQ
EFBF  0510                      ORA     POKMSK
```

```
EFC1  8510                 STA     POKMSK
EFC3  8D0ED2               STA     IRQEN

                   ;       Set DLI status.

EFC6  A940                 LDA     #$40              ;disable DLI
EFC8  8D0ED4               STA     NMIEN
EFCB  2C6E02               BIT     FINE
EFCE  100C ^EFDC           BPL     COC2              ;if not fine scrolling (VBL:

EFD0  A9C4                 LDA     #low FDL
EFD2  8D0002               STA     VDSLST            ;DLI vector
EFD5  A9FC                 LDA     #high FDL
EFD7  8D0102               STA     VDSLST+1
EFDA  A9C0                 LDA     #$C0

EFDC  8D0ED4       COC2    STA     NMIEN

                   ;       Clear control.

EFDF  A900                 LDA     #0
EFE1  8D9302               STA     TINDEX            ;clear text index (must alw:
EFE4  8564                 STA     ADRESS
EFE6  857B                 STA     SWPFLG
EFE8  8DF002               STA     CRSINH

                   ;       Set initial tab stops.

EFEB  A00E                 LDY     #14               ;offset to last byte of bit:
EFED  A901                 LDA     #$01              ;tab stop every 8 character:

EFEF  994302       COC3    STA     TABMAP,Y          ;set tab stop
EFF2  88                   DEY
EFF3  10FA ^EFEF           BPL     COC3              ;if not done

                   ;       Load initialize color register shadows.

EFF5  A204                 LDX     #4                ;offset to last color regis:

EFF7  BD08F8       COC4    LDA     TDSC,X            ;default screen color
EFFA  9DC402               STA     COLOR0,X          ;set color register shadow
EFFD  CA                   DEX
EFFE  10F7 ^EFF7           BPL     COC4              ;if not done

                   ;       Set up.

F000  A46A                 LDY     RAMTOP            ;(high) RAM size
F002  88                   DEY                       ;decrement (high) RAM size
F003  8C9502               STY     TXTMSC+1
F006  A960                 LDA     #low [$0000-160]          ;low RAM size - 160
F008  8D9402               STA     TXTMSC
F00B  A657                 LDX     DINDEX            ;mode
F00D  BD4DEE               LDA     TAGM,X            ;convert to ANTIC code
F010  8551                 STA     HOLD1             ;ANTIC code
F012  A56A                 LDA     RAMTOP            ;(high) RAM size
F014  8565                 STA     ADRESS+1
```

```
                         ;        Allocate memory.

F016  BC1DEE                      LDY     TSMA,X         ;number of 40-byte blocks t:

F019  A928           COC5         LDA     #40            ;40 bytes
F01B  207AF5                      JSR     DBS            ;perform double byte subtra:
F01E  88                          DEY
F01F  D0F8 ^F019                  BNE     COC5           ;if not done

                         ;        Clear GTIA modes.

F021  AD6F02                      LDA     GPRIOR
F024  293F                        AND     #$3F           ;clear GTIA modes
F026  8567                        STA     OPNTMP+1
F028  A8                          TAY

                         ;        Determine mode.

F029  E008                        CPX     #8
F02B  901F ^F04C                  BCC     COC7           ;if mode < 8

F02D  E00F                        CPX     #15
F02F  F00D ^F03E                  BEQ     COC6           ;if mode 15

F031  E00C                        CPX     #12
F033  B017 ^F04C                  BCS     COC7           ;if mode >= 12

                         ;        Process modes 9, 10 and 11.

F035  8A                          TXA                    ;mode
F036  6A                          ROR     A
F037  6A                          ROR     A
F038  6A                          ROR     A
F039  29C0                        AND     #$C0           ;extract 2 low bits (in 2 h:
F03B  0567                        ORA     OPNTMP+1
F03D  A8                          TAY

                         ;        Establish line boundary at X000.

F03E  A910           COC6         LDA     #16            ;subtract 16 for page bound:
F040  207AF5                      JSR     DBS            ;perform double byte subtra:

                         ;        Check for mode 11.

F043  E00B                        CPX     #11
F045  D005 ^F04C                  BNE     COC7           ;if mode 11

                         ;        Set GTIA luminance.

F047  A906                        LDA     #6             ;GTIA luminance value
F049  8DC802                      STA     COLOR4         ;background color

                         ;        Set new priority.

F04C  8C6F02         COC7         STY     GPRIOR         ;new priority

                         ;        Set memory scan counter.
```

```
F04F  A564                     LDA     ADRESS          ;memory scan counter
F051  8558                     STA     SAVMSC          ;save memory scan counter
F053  A565                     LDA     ADRESS+1
F055  8559                     STA     SAVMSC+1

                       ;       Wait for VBLANK.

F057  AD0BD4           COC8    LDA     VCOUNT
F05A  C97A                     CMP     #$7A
F05C  D0F9 ^F057               BNE     COC8            ;if VBLANK has not occured

                       ;       Put display list under RAM.

F05E  2078F5                   JSR     DSD             ;perform double byte single:
F061  BD50EE                   LDA     TDLV,X          ;display list vulnerability
F064  F006 ^F06C               BEQ     COC9            ;if not vulnerable

F066  A9FF                     LDA     #$FF
F068  8564                     STA     ADRESS
F06A  C665                     DEC     ADRESS+1        ;drop down 1 page

F06C  2065F5           COC9    JSR     DDD             ;perform double byte double:
F06F  A564                     LDA     ADRESS          ;end of-display list
F071  8568                     STA     SAVADR          ;save address
F073  A565                     LDA     ADRESS+1
F075  8569                     STA     SAVADR+1

                       ;       Set up.

F077  A941                     LDA     #$41            ;ANTIC wait for VBLANK and :
F079  2070F5                   JSR     SDI             ;store data indirect
F07C  8666                     STX     OPNTMP
F07E  A918                     LDA     #24
F080  8DBF02                   STA     BOTSCR

                       ;       Check for modes 9 ,10 and 11.

F083  A557                     LDA     DINDEX          ;mode
F085  C90C                     CMP     #12
F087  B004 ^F08D               BCS     COC10           ;if mode >= 12, mixed mode :

F089  C909                     CMP     #9
F08B  B039 ^F0C6               BCS     COC12           ;if mode >= 9, mixed mode n:

                       ;       Check for mixed mode.

F08D  A52A             COC10   LDA     ICAX1Z
F08F  2910                     AND     #MXDMOD
F091  F033 ^F0C6               BEQ     COC12           ;if not mixed mode

                       ;       Process mixed mode.

F093  A904                     LDA     #4
F095  8DBF02                   STA     BOTSCR
F098  A202                     LDX     #2
F09A  AD6E02                   LDA     FINE
```

```
F09D  F003 ^F0A2           BEQ     COC11           ;if not fine scrolling

F09F  20A0F5               JSR     SSE             ;set scrolling display list:

F0A2  A902        COC11    LDA     #$02
F0A4  2069F5               JSR     SDF             ;store data indirect for fi:
F0A7  CA                   DEX
F0A8  10F8 ^F0A2           BPL     COC11           ;if not done

                  ;        Reload MSC for text.

F0AA  A46A                 LDY     RAMTOP          ;(high) RAM size
F0AC  88                   DEY                     ;decrement (high) RAM size
F0AD  98                   TYA
F0AE  2070F5               JSR     SDI             ;store data indirect
F0B1  A960                 LDA     #low [$0000-160]        ;low RAM size - 160
F0B3  2070F5               JSR     SDI             ;store data indirect
F0B6  A942                 LDA     #$42            ;fine scrolling
F0B8  2069F5               JSR     SDF             ;store data indirect
F0BB  18                   CLC
F0BC  A910                 LDA     #MXDMOD
F0BE  6566                 ADC     OPNTMP
F0C0  A8                   TAY
F0C1  BE2DEE               LDX     TDLE,Y
F0C4  D015 ^F0DB           BNE     COC13

                  ;        Check mode.

F0C6  A466        COC12    LDY     OPNTMP
F0C8  BE2DEE               LDX     TDLE,Y          ;number of display list ent:
F0CB  A557                 LDA     DINDEX          ;mode
F0CD  D00C ^F0DB           BNE     COC13           ;if not mode 0

                  ;        Check for fine scrolling.

F0CF  AD6E02               LDA     FINE            ;fine scrolling flag
F0D2  F007 ^F0DB           BEQ     COC13           ;if not fine scrolling

                  ;        Process fine scrolling.

F0D4  20A0F5               JSR     SSE             ;set scrolling display list:
F0D7  A922                 LDA     #$22
F0D9  8551                 STA     HOLD1

                  ;        Continue.

F0DB  A551        COC13    LDA     HOLD1
F0DD  2070F5               JSR     SDI             ;store data indirect
F0E0  CA                   DEX
F0E1  D0F8 ^F0DB           BNE     COC13           ;if not done

                  ;        Determine mode.

F0E3  A557                 LDA     DINDEX          ;mode
F0E5  C908                 CMP     #8
F0E7  9026 ^F10F           BCC     COC16           ;if mode < 8
```

```
F0E9  C90F                   CMP     #15
F0EB  F004 ^F0F1             BEQ     COC14           ;if mode 15

F0ED  C90C                   CMP     #12
F0EF  B01E ^F10F             BCS     COC16           ;if mode >= 12

                     ;       Process modes 8, 9, 10, 11 and 15.

F0F1  A25D           COC14   LDX     #93             ;remaining number of DLE's
F0F3  A56A                   LDA     RAMTOP          ;(high) RAM size
F0F5  38                     SEC
F0F6  E910                   SBC     #high $1000     ;subtract 4K
F0F8  2070F5                 JSR     SDI             ;store data indirect
F0FB  A900                   LDA     #low $0000
F0FD  2070F5                 JSR     SDI             ;store data indirect
F100  A551                   LDA     HOLD1           ;ANTIC MSC code
F102  0940                   ORA     #$40
F104  2070F5                 JSR     SDI             ;store data indirect

F107  A551           COC15   LDA     HOLD1           ;remaining DLE's
F109  2070F5                 JSR     SDI             ;store data indirect
F10C  CA                     DEX
F10D  D0F8 ^F107             BNE     COC15           ;if DLE's remain

                     ;       Complete display list with LMS.

F10F  A559           COC16   LDA     SAVMSC+1        ;high saved memory scan cou
F111  2070F5                 JSR     SDI             ;store data indirect
F114  A558                   LDA     SAVMSC          ;low saved memory scan coun
F116  2070F5                 JSR     SDI             ;store data indirect
F119  A551                   LDA     HOLD1
F11B  0940                   ORA     #$40
F11D  2070F5                 JSR     SDI             ;store data indirect
F120  A970                   LDA     #$70            ;8 blank lines
F122  2070F5                 JSR     SDI             ;store data indirect
F125  A970                   LDA     #$70            ;8 blank lines
F127  2070F5                 JSR     SDI             ;store data indirect
F12A  A564                   LDA     ADRESS          ;display list address
F12C  8D3002                 STA     SDLSTL          ;save display list address
F12F  A565                   LDA     ADRESS+1
F131  8D3102                 STA     SDLSTL+1
F134  A970                   LDA     #$70            ;8 blank lines
F136  2070F5                 JSR     SDI             ;store data indirect
F139  A564                   LDA     ADRESS          ;display list address
F13B  8DE502                 STA     MEMTOP          ;update top of memory
F13E  A565                   LDA     ADRESS+1
F140  8DE602                 STA     MEMTOP+1
F143  A001                   LDY     #1              ;offset
F145  AD3002                 LDA     SDLSTL          ;saved display list address
F148  9168                   STA     (SAVADR),Y
F14A  C8                     INY
F14B  AD3102                 LDA     SDLSTL+1
F14E  9168                   STA     (SAVADR),Y

                     ;       Check status.

F150  A54C                   LDA     DSTAT           ;status
```

```
F152  1010 ^F164            BPL     COC18            ;if no error

                    ;       Process error.

F154  8DEC03        COC17   STA     DERRF            ;screen OPEN error flag
F157  2094EF                JSR     EOP              ;perform editor OPEN
F15A  ADEC03                LDA     DERRF            ;restore status
F15D  A000                  LDY     #0               ;no screen OPEN error indic:
F15F  8CEC03                STY     DERRF            ;screen OPEN error flag
F162  A8                    TAY                      ;status
F163  60                    RTS                      ;return

                    ;       Check clear inhibit.

F164  A52A          COC18   LDA     ICAX1Z
F166  2920                  AND     #$20             ;extract clear inhibit bit
F168  D00B ^F175            BNE     COC19            ;if clear inhibited

                    ;       Clear screen.

F16A  2020F4                JSR     CSC              ;clear screen
F16D  8D9002                STA     TXTROW           ;set cursor at top row
F170  A552                  LDA     LMARGN           ;left margin
F172  8D9102                STA     TXTCOL           ;set cursor at left margin

                    ;       Exit.

F175  A922          COC19   LDA     #$22             ;turn on DMA control
F177  0D2F02                ORA     SDMCTL
F17A  8D2F02                STA     SDMCTL
F17D  4C08F2                JMP     SEC              ;set exit conditions, retur:


                    **      SGB - Perform Screen GET-BYTE
                    *
                    *       ENTRY   JSR     SGB
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F180        SGB     =       *       ;entry
F180  20CAF6                JSR     CCR     ;check cursor range
F183  208FF1                JSR     GDC     ;get data under cursor
F186  206AF7                JSR     CIA     ;convert internal character to ATAS:
F189  200AF6                JSR     SZA     ;set zero data and advance cursor
F18C  4C1EF2                JMP     SST     ;perform screen STATUS, return
```

```
                        **      GDC - Get Data under Cursor
                        *
                        *       ENTRY   JSR     GDC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F18F          GDC     =       *       ;entry
F18F  20ACF5                    JSR     CCA     ;convert cursor row/column to addre:
F192  B164                      LDA     (ADRESS),Y
F194  2DA002                    AND     DMASK

F197  466F              GDC1    LSR     SHFAMT  ;shift data down to low bits
F199  B003 ^F19E                BCS     GDC2    ;if done

F19B  4A                        LSR     A
F19C  10F9 ^F197                BPL     GDC1    ;continue shifting

F19E  8DFA02            GDC2    STA     CHAR
F1A1  C900                      CMP     #0      ;retore flags
F1A3  60                        RTS             ;return



                        **      SPB - Perform Screen PUT-BYTE
                        *
                        *       ENTRY   JSR     SPB
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F1A4          SPB     =       *       ;entry
F1A4  8DFB02                    STA     ATACHR

                        ;       JSR     ROD     ;restore old data under cursor

F1A7  C97D                      CMP     #CLS
F1A9  D006 ^F1B1                BNE     SPB1    ;if not clear screen

F1AB  2020F4                    JSR     CSC     ;clear screen
F1AE  4C06F2                    JMP     SEC     ;set exit conditions, return

F1B1  20CAF6            SPB1    JSR     CCR     ;check cursor range
                        ;       JMP     CEL     ;check EOL, return
                        **      CEL - Check End of Line
                        *
                        *       ENTRY   JSR     CEL
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F1B4    CEL     =       *       ;entry
F1B4  ADF602            LDA     ATACHR
F1B7  C99B              CMP     #EOL
F1B9  D006 ^F1C1        BNE     CEL1    ;if not EOL

F1BB  2061F6            JSR     RWS     ;return with scrolling
F1BE  4C0BF2            JMP     SEC     ;set exit conditions, return

F1C1  20CAF1    CEL1    JSR     PLO     ;plot point
F1C4  200EF6            JSR     SEA     ;set EOL data and advance cursor
F1C7  4C0BF2            JMP     SEC     ;set exit conditions, return


                        **      PLO - Plot Point
                        *
                        *       ENTRY   JSR     PLO
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F1CA    PLO     =       *               ;entry

                ;       Wait for start/stop flag clear.

F1CA  ADFF02    PLO0    LDA     SSFLAG          ;start/stop flag
F1CD  D0FB ^F1CA        BNE     PLO0            ;if start/stop flag non-zer:

                ;       Save cursor row/column.

F1CF  A202              LDX     #2              ;offset to last byte

F1D1  B554      PLO1    LDA     ROWCRS,X        ;byte of cursor row/column
F1D3  955A              STA     OLDROW,X        ;save byte of cursor row/co:
F1D5  CA                DEX
F1D6  10F9 ^F1D1        BPL     PLO1            ;if not done

                ;       Convert ATASCII character to internal.

F1D8  ADFB02            LDA     ATACHR          ;character
F1DB  A8                TAY                     ;character
F1DC  2A                ROL     A
F1DD  2A                ROL     A
```

```
                        **      SEC - Set Exit Conditions
                        *
                        *       ENTRY   JSR     SEC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F20B          SEC     =       *       ;entry
F20B    208FF1                  JSR     GDC     ;get data under cursor
F20E    855D                    STA     OLDCHR
F210    A657                    LDX     DINDEX  ;mode
F212    D00A ^F21E              BNE     SST     ;if graphics, no cursor

F214    AEF002                  LDX     CRSINH  ;cursor inhibit flag
F217    D005 ^F21E              BNE     SST     ;if cursor inhibited

F219    4980                    EOR     #$80    ;complement most significant bit
F21B    20E9F1                  JSR     SPQ     ;display
                        ;       JMP     SST     ;perform screen status, return


                        **      SST - Perform Screen STATUS
                        *
                        *       ENTRY   JSR     SST
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F21E          SST     =       *       ;entry
F21E    A44C                    LDY     DSTAT   ;status
F220    4C26F2                  JMP     SST1    ;continue
```

```
F1DE  2A                    ROL     A
F1DF  2A                    ROL     A
F1E0  2903                  AND     #3
F1E2  AA                    TAX                     ;index into TAIC
F1E3  98                    TYA                     ;character
F1E4  299F                  AND     #$9F            ;strip off column address
F1E6  1D49FB                ORA     TAIC,X          ;or in new column address
                  ;         JMP     SPQ             ;display, return



                  **        SPQ - Display
                  *
                  *         ENTRY   JSR     SPQ
                  *
                  *         MODS
                  *                 Original Author Unknown
                  *                 1. Bring closer to Coding Standard (object :
                  *                    R. K. Nordin 11/01/83

      = F1E9      SPQ       =       *               ;entry

                  ;         Set CHAR.

F1E9  8DFA02                STA     CHAR            ;character

                  ;         Convert cursor row/column to address.

F1EC  20ACF5                JSR     CCA             ;convert cursor row/column :

                  ;         Shift up to proper position.

F1EF  ADFA02                LDA     CHAR            ;character

F1F2  466F        SPQ1      LSR     SHFAMT
F1F4  B004 ^F1FA            BCS     SPQ2            ;if done

F1F6  0A                    ASL     A
F1F7  4CF2F1                JMP     SPQ1            ;continue shifting

                  ;         Update data.

F1FA  2DA002      SPQ2      AND     DMASK
F1FD  8550                  STA     TMPCHR          ;save shifted data
F1FF  ADA002                LDA     DMASK           ;display mask
F202  49FF                  EOR     #$FF            ;complement display mask
F204  3164                  AND     (ADRESS),Y      ;mask off old data
F206  0550                  ORA     TMPCHR          ;or in new data
F208  9164                  STA     (ADRESS),Y      ;update data
F20A  60                    RTS                     ;return
```

```
F223                          FIX     $F223



                        **      F223 - $F223 Patch
                        *
                        *       For compatibility with OS Revision B, perform power:

     = F223         PPD     =       *       ;entry
F223  4CFCC8                JMP     SES     ;select and execute self-test
```

```
F226                  ;       Continue.

F226  A901      SST1    LDA     #SUCCES ;indicate success
F228  854C              STA     DSTAT   ;status
F22A  ADF502            LDA     ATACHR  ;data
                ;       JMP     ESP     ;return



                **      ESP - Perform Editor SPECIAL
                *
                *       ESP does nothing.
                *
                *       ENTRY   JSR     ESP
                *
                *       MODS
                *               Original Author Unknown
                *               1. Bring closer to Coding Standard (object :
                *                  R. K. Nordin 11/01/83


     = F22D     ESP     =       *       ;entry
F22D  60                RTS             ;return



                **      ECL - Perform Editor CLOSE
                *
                *       ENTRY   JSR     ECL
                *
                *       MODS
                *               Original Author Unknown
                *               1. Bring closer to Coding Standard (object :
                *                  R. K. Nordin 11/01/83


     = F22E     ECL     =       *       ;entry

                ;       Check for fine scrolling.

F22E  2C6E02            BIT     FINE    ;fine scrolling flag
F231  10EB ^F21E        BPL     SST     ;if not fine scrolling, perform STA:

                ;       Process fine scrolling.

F233  A940              LDA     #$40
F235  8D0ED4            STA     NMIEN           ;disable DLI
F238  A900              LDA     #0              ;clear fine scrolling flag
F23A  8D6E02            STA     FINE
F23D  A9CE              LDA     #low RIR        ;return from interrupt rout:
F23F  8D0002            STA     VDSLST          ;restore initial DLI vector:
F242  A9C0              LDA     #high RIR
F244  8D0102            STA     VDSLST+1
F247  4C94EF            JMP     EOP             ;perform editor OPEN, retur:
```

```
                      **      EGB - Perform Editor GET-BYTE
                      *
                      *       ENTRY   JSR     EGB
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


       = F24A         EGB     =        *      ;entry

                      ;       Initialize.

F24A  2062F9                  JSR     SWA     ;swap
F24D  20BCF6                  JSR     CRE     ;check cursor range for editor
F250  A56B                    LDA     BUFCNT  ;buffer count
F252  D034 ^F288              BNE     EGB4    ;if something in the buffer

                      ;       Get line.

F254  A554                    LDA     ROWCRS          ;cursor row
F256  856C                    STA     BUFSTR          ;buffer start pointer
F258  A555                    LDA     COLCRS          ;low cursor column
F25A  856D                    STA     BUFSTR+1        ;high buffer start pointer

F25C  20FDF2          EGB1    JSR     KGB     ;perform keyboard GET-BYTE
F25F  844C                    STY     DSTAT   ;status
F261  ADFB02                  LDA     ATACHR  ;ATASCII character
F264  C99B                    CMP     #EOL
F266  F012 ^F27A              BEQ     EGB3    ;if EOL

F268  208EF2                  JSR     PCH     ;process character
F26B  2062F9                  JSR     SWA     ;swap
F26E  A563                    LDA     LOGCOL  ;logical column
F270  C971                    CMP     #113    ;column near column 120
F272  D003 ^F277              BNE     EGB2    ;if not near column 120, no beep

F274  2056F5                  JSR     BEL     ;beep

F277  4C5CF2          EGB2    JMP     EGB1    ;process next character

                      ;       Process EOL.

F27A  2018F7          EGB3    JSR     ROD             ;restore old data under cur:
F27D  2081F8                  JSR     CBC             ;compute buffer count
F280  A56C                    LDA     BUFSTR          ;buffer start pointer
F282  8554                    STA     ROWCRS          ;cursor row
F284  A56D                    LDA     BUFSTR+1        ;high buffer start pointer
F286  8555                    STA     COLCRS          ;low cursor column

                      ;       Check buffer count.

F288  A56B            EGB4    LDA     BUFCNT  ;buffer count
F28A  F011 ^F29D              BEQ     EGB6    ;if buffer count zero

                      ;       Decrement and check buffer count.
```

```
F28C  C66B          EGB5    DEC     BUFCNT  ;decrement buffer count
F28E  F00D ^F29D            BEQ     EGB6    ;if buffer count zero

                    ;       Check status.

F290  A54C                  LDA     DSTAT   ;status
F292  30F8 ^F28C            BMI     EGB5    ;if error, continue decrementing.

                    ;       Perfrom GET-BYTE.

F294  2080F1                JSR     SGB     ;perform screen GET-BYTE
F297  8DFB02                STA     ATACHR  ;ATASCII character
F29A  4C62F9                JMP     SWA     ;swap, return

                    ;       Exit.

F29D  2061F6        EGB6    JSR     RWS     ;return with scrolling
F2A0  A99B                  LDA     #EOL
F2A2  8DFB02                STA     ATACHR  ;ATASCII character
F2A5  2006F2                JSR     SEC     ;set exit conditions
F2A8  844C                  STY     DSTAT   ;status
F2AA  4C62F9                JMP     SWA     ;swap, return



                    **      IRA - Invoke Routine Pointed to by ADRESS
                    *
                    *       ENTRY   JSR     IRA
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = F2AD         IRA     =       *               ;entry
F2AD  6C6400                JMP     (ADRESS)        ;execute, return



                    **      EPB - Perform Editor PUT-BYTE
                    *
                    *       ENTRY   JSR     EBP
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = F2B0         EPB     =       *       ;entry
F2B0  8DFB02                STA     ATACHR  ;ATASCII character
F2B3  2062F9                JSR     SWA     ;swap
F2B6  20BCF6                JSR     CRE     ;check cursor range for editor
```

```
F2B9  A900                  LDA     #0
F2BB  8DE803                STA     SUPERF  ;clear super function flag
                   ;        JMP     PCH     ;process character, return



                   **       PCH - Process Character
                   *
                   *        PCH displays the character or processes control cha:
                   *        super functions (shifted function keys).
                   *
                   *        ENTRY   JSR     PCH
                   *
                   *        MODS
                   *                Original Author Unknown
                   *                1. Bring closer to Coding Standard (object :
                   *                   R. K. Nordin 11/01/83

      = F2BE       PCH      =       *       ;entry
F2BE  2018F7                JSR     ROD     ;restore old data under cursor
F2C1  203CF9                JSR     CCC     ;check for control character
F2C4  F009 ^F2CF            BEQ     PCH2    ;if control chanacter

                   ;        Display character.

F2C6  0EA202       PCH1     ASL     ESCFLG  ;escape flag
F2C9  20B4F1                JSR     CEL     ;check EOL
F2CC  4C62F9                JMP     SWA     ;swap, return

                   ;        Process control character.

F2CF  ADFE02       PCH2     LDA     DSPFLG  ;display flag
F2D2  0DA202                ORA     ESCFLG  ;escape flag
F2D5  D0EF ^F2C6            BNE     PCH1    ;if dislay or escape, display chara:

                   ;        Continue.

F2D7  0EA202                ASL     ESCFLG
F2DA  E8                    INX

                   ;        Check for super function.

F2DB  ADE803                LDA     SUPERF
F2DE  F005 ^F2E5            BEQ     PCH3            ;if not super function

                   ;        Adjust for super function.

F2E0  8A                    TXA
F2E1  18                    CLC
F2E2  6920                  ADC     #TSFR-TCCR-3
F2E4  AA                    TAX                     ;adjusted offset

                   ;        Process control character or super function.

F2E5  BD00FB       PCH3     LDA     TCCR,X          ;low routine address
```

```
F2E8  8564                      STA    ADRESS
F2EA  BD0EF3                    LDA    TCCR+1,X        ;high routine address
F2ED  8565                      STA    ADRESS+1
F2EF  20A0F2                    JSR    IRA             ;invoke routine pointed to :
F2F2  2008F2                    JSR    SEC             ;set exit conditions
F2F5  4C62F9                    JMP    SWA             ;swap, return



                        **      IGN - Ignore Character and Perform Keyboard GET-BYT:
                        *
                        *       ENTRY   JSR     IGN
                        *
                        *       EXIT
                        *               CH = $FF
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F2F8            IGN     =       *       ;entry
F2F8  A9FF                      LDA     #$FF    ;clear code indicator
F2FA  8DFC02                    STA     CH      ;key code
                        ;       JMP     KGB     ;perform keyborad GET-BYTE, return



                        **      KGB - Perform Keyboard GET-BYTE
                        *
                        *       ENTRY   JSR     KGB
                        *
                        *       NOTES
                        *               Problem: byte wasted by unnecessary TAX nea:
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F2FD            KGB     =       *       ;entry

                        ;       Initialize.

F2FD  A900              KGB1    LDA     #0
F2FF  8DE803                    STA     SUPERF  ;clear super function flag

                        ;       Check for special edit read mode.

F302  A52A                      LDA     ICAX1Z
F304  4A                        LSR     A
F305  B06F ^F376                BCS     KGB11   ;if special edit read mode
```

```
                         ;       Check for BREAK abort.

F307  A980                       LDA     #BRKABT ;assume BREAK abort
F309  A611                       LDX     BRKKEY  ;BREAK key flag
F30B  F065 ^F372                 BEQ     KGB10   ;if BREAK abort

                         ;       Check for character.

F30D  ADFC02                     LDA     CH      ;key code
F310  C9FF                       CMP     #$FF    ;clear code indicator
F312  F0E9 ^F2FD                 BEQ     KGB1    ;if no character

                         ;       Process character.

F314  857C                       STA     HOLDCH  ;save character
F316  A2FF                       LDX     #$FF    ;clear code indicator
F318  8EFC02                     STX     CH      ;key code

                         ;       Sound key click if desired.

F31B  AED802                     LDX     NOCLIK  ;click inhibit flag
F31E  D003 ^F323                 BNE     KGB2    ;if click inhibited

F320  2083F9                     JSR     SKC     ;sound key click

                         ;       Set offset to key definition.

F323  A8                 KGB2    TAY             ;save character

                         ;       Check for CTRL and SHIFT together.

F324  C0C0                       CPY     #$C0
F326  B0D0 ^F2F8                 BCS     IGN     ;if CTRL and SHIFT together, ignore

                         ;       Convert to ATASCII character.

F328  B179                       LDA     (KEYDEF),Y      ;ATASCII character

                         ;       Set ATASCII character.

F32A  8DFB02             KGB3    STA     ATACHR  ;ATASCII character
F32D  AA                         TAX
F32E  3003 ^F333                 BMI     KGB4    ;if special key

F330  4CB4F3                     JMP     KGB17   ;process shift/control lock

                         ;       Check for null character.

F333  C980               KGB4    CMP     #$80
F335  F0C1 ^F2F8                 BEQ     IGN     ;if null, ignore

                         ;       Check for inverse video key.

F337  C981                       CMP     #$81
F339  D00A ^F345                 BNE     KGB5    ;if not inverse video key

                         ;       Process inverse video key.
```

```
F33B  AD8602                LDA     INVFLG
F33E  4980                  EOR     #$80
F340  8D8602                STA     INVFLG
F343  B0B3 ^F2F8            BCS     IGN      ;ignore

                    ;       Check for CAPS key.

F345  C982          KGB5    CMP     #$82
F347  D00C ^F355            BNE     KGB6     ;if not CAPS key

                    ;       Process CAPS key.

F349  ADBE02                LDA     SHFLOK   ;shift/control lock flags
F34C  F00B ^F359            BEQ     KGB7     ;if no lock, process CAPS lock

F34E  A900                  LDA     #$00     ;no lock indicator
F350  8DBE02                STA     SHFLOK   ;shift/control lock flags
F353  F0A3 ^F2F8            BEQ     IGN      ;ignore

                    ;       Check for SHIFT-CAPS key.

F355  C983          KGB6    CMP     #$83
F357  D007 ^F360            BNE     KGB8     ;if not SHIFT-CAPS

                    ;       Process SHIFT-CAPS key.

F359  A940          KGB7    LDA     #$40     ;CAPS lock indicator
F35B  8DBE02                STA     SHFLOK   ;shift/control lock flags
F35E  D098 ^F2F8            BNE     IGN      ;ignore

                    ;       Check for CTRL-CAPS key.

F360  C984          KGB8    CMP     #$84
F362  D008 ^F36C            BNE     KGB9     ;if not CTRL-CAPS

                    ;       Process CTRL-CAPS key.

F364  A980                  LDA     #$80     ;control lock indicator
F366  8DBE02                STA     SHFLOK   ;shift/control lock flags
F369  4CF8F2                JMP     IGN      ;ignore

                    ;       Check for CTRL-3 key.

F36C  C985          KGB9    CMP     #$85
F36E  D00B ^F37B            BNE     KGB12    ;if not CTRL-3 key.

                    ;       Process CTRL-3 key.

F370  A988                  LDA     #EOFERR

                    ;       Set status and BREAK key flag.

F372  854C          KGB10   STA     DSTAT    ;status
F374  8511                  STA     BRKKEY   ;BREAK key flag

                    ;       Set EOL character.
```

```
F376  A99B        KGB11   LDA    #EOL
F378  4CDAF3              JMP    KGB19   ;set ATASCII character

                  ;       Check for CTRL-F3 key.

F37B  C989        KGB12   CMP    #$89
F37D  D010 ^F38F          BNE    KGB14   ;if not CTRL-F3 key

                  ;       Process CTRL-F3 key.

F37F  ADDB02              LDA    NOCLIK  ;toggle keyclick status
F382  49FF                EOR    #$FF
F384  8DDB02              STA    NOCLIK
F387  D003 ^F38C          BNE    KGB13   ;if click inhibited

F389  2083F9              JSR    SKC     ;sound key click

F38C  4CF8F2      KGB13   JMP    IGN     ;ignore

                  ;       Check for function key.

F38F  C98E        KGB14   CMP    #$8E
F391  B012 ^F3A5          BCS    KGB16   ;if code >= $8E, not a function key

F393  C98A                CMP    #$8A
F395  90F5 ^F38C          BCC    KGB13   ;if code < $8A, not a function key,:

                  ;       Process function key.

F397  E98A                SBC    #$8A             ;convert $8A - $8D TO 0 - 3
F399  067C                ASL    HOLDCH           ;saved character
F39B  1002 ^F39F          BPL    KGB15            ;if no SHIFT

F39D  0904                ORA    #$04             ;convert 0 - 3 to 4 - 7

F39F  A8          KGB15   TAY                     ;offset to function key def:
F3A0  B160                LDA    (FKDEF),Y        ;function key
F3A2  4C2AF3              JMP    KGB3             ;set ATASCII character

                  ;       Check for super function.

F3A5  C992        KGB16   CMP    #$92
F3A7  B00B ^F3B4          BCS    KGB17   ;if code >= $92, process shift/cont:

F3A9  C98E                CMP    #$8E
F3AB  90DF ^F38C          BCC    KGB13   ;if code < $8E, not super function,:

                  ;       Process super function.

F3AD  E972                SBC    #$8E-$1C         ;convert $8E - $91 TO $1C -:
F3AF  EEE803              INC    SUPERF           ;set super function flag
F3B2  D026 ^F3DA          BNE    KGB19            ;set ATASCII character

                  ;       Process shift/control lock.

F3B4  A57C        KGB17   LDA    HOLDCH  ;saved character
```

```
F3B6  C940                CMP    #$40
F3B8  B015 ^F3CF          BCS    KGB18    ;if not lower case

F3BA  ADFB02              LDA    ATACHR   ;ATASCII character
F3BD  C961                CMP    #'a'
F3BF  900E ^F3CF          BCC    KGB18    ;if < "a", do not process

F3C1  C97B                CMP    #'z'+1
F3C3  B00A ^F3CF          BCS    KGB18    ;if > "z", do not process

F3C5  ADBE02              LDA    SHFLOK   ;shift/control lock flags
F3C8  F005 ^F3CF          BEQ    KGB18    ;if no lock

F3CA  057C                ORA    HOLDCH   ;modify character
F3CC  4C23F3              JMP    KGB2     ;reprocess character

                  ;       Invert character, if necessary.

F3CF  203CF9      KGB18   JSR    CCC      ;check for control character
F3D2  F009 ^F3DD          BEQ    KGB20    ;if control character, do not inver

F3D4  ADFB02              LDA    ATACHR   ;ATASCII character
F3D7  4DB602              EOR    INVFLG   ;invert character

                  ;       Set ATASCII character.

F3DA  8DFB02      KGB19   STA    ATACHR   ;ATASCII character

                  ;       Exit.

F3DD  4C1EF2      KGB20   JMP    SST      ;perform screen status, return



                  **      ESC - Escape
                  *
                  *       ENTRY   JSR     ESC
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


      = F3E0      ESC     =      *        ;entry
F3E0  A980                LDA    #$80     ;indicate escape detected
F3E2  8DA202              STA    ESCFLG   ;escape flag
F3E5  60                  RTS             ;return
```

```
                        **      CUP - Move Cursor Up
                        *
                        *       ENTRY   JSR     CUP
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F3E6            CUP     =       *       ;entry
F3E6  C654                      DEC     ROWCRS  ;decrement cursor row
F3E8  1006 ^F3F0                BPL     CUP2    ;if row positive

F3EA  AEBF02                    LDX     BOTSCR  ;screen bottom
F3ED  CA                        DEX             ;screen bottom - 1

F3EE  8654              CUP1    STX     ROWCRS  ;update cursor row

F3F0  4C0CF9            CUP2    JMP     SBS     ;set buffer start and logical colum:



                        **      CDN - Move Cursor Down
                        *
                        *       ENTRY   JSR     CDN
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F3F3            CDN     =       *       ;entry
F3F3  E654                      INC     ROWCRS  ;increment cursor row
F3F5  A554                      LDA     ROWCRS  ;cursor row
F3F7  CDBF02                    CMP     BOTSCR  ;screen bottom
F3FA  90F4 ^F3F0                BCC     CUP2    ;if at bottom, set buffer start, re:

F3FC  A200                      LDX     #0
F3FE  F0EE ^F3EE                BEQ     CUP1    ;update cursor row, return
```

```
                        **      CLF - Move Cursor Left
                        *
                        *       ENTRY   JSR     CLF
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F400    CLF     =       *       ;entry
F400  C655              DEC     COLCRS  ;decrement low cursor column
F402  A555              LDA     COLCRS  ;low cursor column
F404  3004 ^F40A        BMI     CRM     ;if negative, move cursor to margin:

F406  C552              CMP     LMARGN  ;left margin
F408  B004 ^F40E        BCS     SCC1    ;if at left margin, set logical col:

                        ;       JMP     CRM     ;move cursor to right margin, retur:



                        **      CRM - Move Cursor to Right Margin
                        *
                        *       ENTRY   JSR     CRM
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F40A    CRM     =       *       ;entry
F40A  A553              LDA     RMARGN  ;right margin
                        ;       JMP     SCC     ;set cursor column, return



                        **      SCC - Set Cursor Column
                        *
                        *       ENTRY   JSR     SCC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F40C    SCC     =       *       ;entry
F40C  8555              STA     COLCRS  ;set low cursor column

F40E  4C8EF6    SCC1    JMP     SLC     ;set logical column, return
```

```
                    **      CRT - Move Cursor Right
                    *
                    *       ENTRY   JSR     CRT
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = F411         CRT     =       *       ;entry
F411  E655                  INC     COLCRS  ;increment low cursor column
F413  A555                  LDA     COLCRS  ;low cursor column
F415  C553                  CMP     RMARGN  ;right margin
F417  90F5 ^F40E            BCC     SCC1    ;if before right margin, process, r:

F419  F0F3 ^F40E            BEQ     SCC1    ;if at right margin

                    ;       JMP     CLM     ;move cursor to left margin, return



                    **      CLM - Move Cursor to Left Margin
                    *
                    *       ENTRY   JSR     CLM
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


     = F41B         CLM     =       *       ;entry
F41B  A552                  LDA     LMARGN  ;left margin
F41D  4C0CF4                JMP     SCC     ;set cursor column, return



                    **      CSC - Clear Screen
                    *
                    *       ENTRY   JSR     CSC
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

     = F420         CSC     =       *               ;entry

                    ;       Set memory scan counter address.

F420  20A6F9                JSR     SMS             ;set memory scan counter ad:

                    ;       Clear address.
```

```
F423  A464                LDY     ADRESS
F425  A900                LDA     #0
F427  8564                STA     ADRESS

F429  9164        CSC1    STA     (ADRESS),Y
F42B  C8                  INY
F42C  D0FB ^F429          BNE     CSC1            ;if not done with page

F42E  E665                INC     ADRESS+1
F430  A665                LDX     ADRESS+1
F432  E46A                CPX     RAMTOP          ;(high) RAM size
F434  90F3 ^F429          BCC     CSC1            ;if not done

                  ;       Clean up logical line bit map.

                  ;       LDY     #0              ;offset to first byte of bi:
F436  A9FF                LDA     #$FF

F438  998202      CSC2    STA     LOGMAP,Y        ;byte of logical line bit m:
F43B  C8                  INY
F43C  C004                CPY     #4              ;4 bytes
F43E  90F8 ^F438          BCC     CSC2            ;if not done

                  ;       Exit.

                  ;       JMP     CHM             ;move cursor home, return



                  **      CHM - Move Cursor Home
                  *
                  *       ENTRY   JSR     CHM
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


      = F440      CHM     =       *               ;entry
F440  2097F9              JSR     SCL             ;set cursor at left edge
F443  8563                STA     LOGCOL          ;logical column
F445  856D                STA     BUFSTR+1        ;high buffer start
F447  A900                LDA     #0
F449  8554                STA     ROWCRS          ;cursor row
F44B  8556                STA     COLCRS+1        ;high cursor column
F44D  856C                STA     BUFSTR          ;low buffer start pointer
F44F  60                  RTS                     ;return
```

```
                          **       BSP - Backspace
                          *
                          *        ENTRY    JSR      BSP
                          *
                          *        MODS
                          *                 Original Author Unknown
                          *                 1. Bring closer to Coding Standard (object :
                          *                    R. K. Nordin 11/01/83


        = F450    BSP     =        *        ;entry
F450  A563                LDA      LOGCOL   ;logical column
F452  C552                CMP      LMARGN   ;left margin
F454  F021 ^F477          BEQ      BSP3     ;if at left margin

F456  A555                LDA      COLCRS   ;low cursor column
F458  C552                CMP      LMARGN   ;left margin
F45A  D003 ^F45F          BNE      BSP1     ;if not at left margin

F45C  2023F9              JSR      DWQ      ;see if line should be deleted

F45F  2000F4      BSP1    JSR      CLF      ;move cursor left
F462  A555                LDA      COLCRS   ;low cursor column
F464  C553                CMP      RMARGN   ;right margin
F466  D007 ^F46F          BNE      BSP2     ;if not at right margin

F468  A554                LDA      ROWCRS   ;cursor row  } this can't happen!   because top li
F46A  F003 ^F46F          BEQ      BSP2     ;if row zero }                        is start of

F46C  20E6F3              JSR      CUP      ;move cursor up

F46F  A920        BSP2    LDA      #' '
F471  8DFB02              STA      ATACHR   ;ATASCII character
F474  20CAF1              JSR      PLO      ;plot point

F477  4C8EF8      BSP3    JMP      SLC      ;set logical column, return



                          **       TAB - Tab
                          *
                          *        ENTRY    JSR      TAB
                          *
                          *        MODS
                          *                 Original Author Unknown
                          *                 1. Bring closer to Coding Standard (object :
                          *                    R. K. Nordin 11/01/83

        = F47A    TAB     =        *        ;entry

F47A  2011F4      TAB1    JSR      CRT      ;move cursor right
F47D  A555                LDA      COLCRS   ;low cursor column
F47F  C552                CMP      LMARGN   ;left margin
F481  D008 ^F48B          BNE      TAB2     ;if not at left margin
```

```
F483  2065F6                JSR     RET     ;return
F486  2058F7                JSR     BLG     ;get bit from logical line bit map
F489  B007 ^F492            BCS     TAB3    ;if end of logical line

                    ;       Check for tab stop.

F48B  A563          TAB2    LDA     LOGCOL  ;logical column
F48D  205DF7                JSR     BMG     ;get bit from bit map
F490  90E8 ^F47A            BCC     TAB1    ;if not tab stop, keep looking

                    ;       Set logical column.

F492  4C8EF8        TAB3    JMP     SLC     ;set logical column, return



                    **      STB - Set Tab
                    *
                    *       ENTRY   JSR     STB
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F495        STB     =       *       ;entry
F495  A563                  LDA     LOGCOL  ;logical column
F497  4C3EF7                JMP     BMS     ;set bit in bit map, return



                    **      CTB - Clear Tab
                    *
                    *       ENTRY   JSR     CTB
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F49A        CTB     =       *       ;entry
F49A  A563                  LDA     LOGCOL  ;logical column
F49C  4C4AF7                JMP     BMC     ;clear bit in bit map, return
```

```
                          **     ICH - Insert Character
                          *
                          *      ENTRY   JSR     ICH
                          *
                          *      MODS
                          *              Original Author Unknown
                          *              1. Bring closer to Coding Standard (object :
                          *                 R. K. Nordin 11/01/83


         = F49F           ICH    =       *       ;entry
F49F    204CF9                   JSR     SRC     ;save row and column
F4A2    208FF1                   JSR     GDC     ;get data under cursor
F4A5    857D                     STA     INSDAT
F4A7    A900                     LDA     #0
F4A9    8DBB02                   STA     SCRFLG

F4AC    20E9F1          ICH1     JSR     SPQ     ;store data
F4AF    A563                     LDA     LOGCOL  ;logical column
F4B1    48                       PHA             ;save logical column
F4B2    2012F6                   JSR     ACC     ;advance cursor column
F4B5    68                       PLA             ;saved logical column
F4B6    C563                     CMP     LOGCOL  ;logical column
F4B8    B00C ^F4C6               BCS     ICH2    ;if saved logical column >= logical:

F4BA    A57D                     LDA     INSDAT
F4BC    48                       PHA
F4BD    208FF1                   JSR     GDC     ;get data under cursor
F4C0    857D                     STA     INSDAT
F4C2    68                       PLA
F4C3    4CACF4                   JMP     ICH1    ;continue

                          ;      Exit.

F4C6    2057F9          ICH2     JSR     RRC     ;restore row and column

F4C9    CEBB02          ICH3     DEC     SCRFLG
F4CC    3004 ^F4D2               BMI     ICH4    ;if scroll occurred

F4CE    C654                     DEC     ROWCRS  ;decrement cursor row
F4D0    D0F7 ^F4C9               BNE     ICH3    ;continue

F4D2    4C8EF6          ICH4     JMP     SLC     ;set logical column, return
```

```
                        **      DCH - Delete Character
                        *
                        *       ENTRY   JSR     DCH
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F4D5          DCH     =       *               ;entry

                        ;       Save row and column.

F4D5  204CF9                    JSR     SRC             ;save row and column

                        ;       Get data to the right of cursor.

F4D8  20ACF5            DCH1    JSR     CCA     ;convert cursor row/column to addre:
F4DB  A564                      LDA     ADRESS
F4DD  8568                      STA     SAVADR  ;save address
F4DF  A565                      LDA     ADRESS+1
F4E1  8569                      STA     SAVADR+1
F4E3  A563                      LDA     LOGCOL  ;logical column.
F4E5  48                        PHA             ;save logical column
F4E6  200AF6                    JSR     SZA     ;set zero data and advance cursor
F4E9  68                        PLA             ;saved logical column
F4EA  C563                      CMP     LOGCOL  ;logical column
F4EC  B010 ^F4FE                BCS     DCH2    ;if saved logical column >= logical:

F4EE  A554                      LDA     ROWCRS          ;cursor row
F4F0  CDBF02                    CMP     BOTSCR          ;screen bottom
F4F3  B009 ^F4FE                BCS     DCH2            ;if row off screen, exit

F4F5  208FF1                    JSR     GDC             ;get data under cursor
F4F8  A000                      LDY     #0
F4FA  9168                      STA     (SAVADR),Y      ;put data in previous posit:
F4FC  F0DA ^F4D8                BEQ     DCH1            ;continue

F4FE  A000              DCH2    LDY     #0
F500  98                        TYA
F501  9168                      STA     (SAVADR),Y      ;clear last position
F503  2018F9                    JSR     DQQ             ;try to delete a line
F506  2057F9                    JSR     RRC             ;restore row and column
F509  4C8EF8                    JMP     SLC             ;set logical column, return
```

```
                        **      ILN - Insert Line
                        *
                        *       ENTRY   JSR     ILN
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = F50C          ILN     =       *       ;entry
F50C    38                      SEC
                        ;       JMP     ILN1



                        **      ILN1 - Insert Line
                        *
                        *       ENTRY   JSR     ILN1
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = F50D          ILN1    =       *       ;entry
F50D    20C2F7                  JSR     ELL     ;extend logical line
F510    A552                    LDA     LMARGN  ;left margin
F512    8555                    STA     COLCRS  ;low cursor column
F514    20ACF5                  JSR     CCA     ;convert cursor row/column to addre:
F517    208EF7                  JSR     MLN     ;move line
F51A    20E2F7                  JSR     CLN     ;clear current line
F51D    4C8EF8                  JMP     SLC     ;set logical column, return



                        **      DLN - Delete Line
                        *
                        *       ENTRY   JSR     DLN
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

        = F520          DLN     =       *       ;entry
F520    208EF8                  JSR     SLC     ;set logical column
F523    A451                    LDY     HOLD1
F525    8454                    STY     ROWCRS  ;cursor row
                        ;       JMP     DLN1
                        **      DLN1 - Delete Line
                        *
                        *       ENTRY   JSR     DLN1
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F527          DLN1    =       *               ;entry

F527    A454            DLN0    LDY     ROWCRS          ;cursor row

F529    98              DLN2    TYA
F52A    38                      SEC
F52B    205BF7                  JSR     BLG2            ;get next bit
F52E    08                      PHP
F52F    98                      TYA
F530    18                      CLC
F531    6978                    ADC     #8*[LOGMAP-TABMAP]      ;add offset for log:
F533    28                      PLP
F534    203CF7                  JSR     BMP             ;put bit in bit map
F537    C8                      INY
F538    C018                    CPY     #24
F53A    D0ED ^F529              BNE     DLN2            ;if not done

F53C    ADB402                  LDA     LOGMAP+2
F53F    0901                    ORA     #1              ;set least significant bit
F541    8DB402                  STA     LOGMAP+2        ;update logical line bit ma:
F544    A900                    LDA     #0              ;delete line of data
F546    8555                    STA     COLCRS          ;low cursor column
F548    20ACF5                  JSR     CCA             ;convert cursor row/column :
F54B    202AF8                  JSR     SSD             ;scroll screen for delete

                        ;       Check for new logical line.

F54E    2058F7                  JSR     BLG             ;get bit from logical line :
F551    90D4 ^F527              BCC     DLN0            ;if not new logical line

                        ;       Move cursor to left margin.

F553    4C1BF4                  JMP     CLM             ;move cursor to left margin:
```

```
                    **      BEL - Sound Bell
                    *
                    *       ENTRY   JSR     BEL
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F556        BEL     =       *       ;entry
F556  A020                  LDY     #$20

F558  2083F9        BEL1    JSR     SKC     ;sound key click
F55B  88                    DEY
F55C  10FA ^F558            BPL     BEL1    ;if not done

F55E  60                    RTS             ;return



                    **      CBT - Move Cursor to Bottom
                    *
                    *       ENTRY   JSR     CBT
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F55F        CBT     =       *       ;entry
F55F  2040F4                JSR     CHM     ;move cursor home
F562  4CE6F3                JMP     CUP     ;move cursor up, return



                    **      DDD - Perform Double Byte Double Decrement
                    *
                    *       ENTRY   JSR     DDD
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F565        DDD     =       *       ;entry
F565  A902                  LDA     #2      ;indicate subtracting 2
F567  D011 ^F57A            BNE     DBS     ;perform double byte subtract, retu:
```

```
                   **      SDF - Store Data Indirect for Fine Scrolling
                   *
                   *       ENTRY   JSR     SDF
                   *
                   *       MODS
                   *               Original Author Unknown
                   *               1. Bring closer to Coding Standard (object :
                   *                  R. K. Nordin 11/01/83


     = F569        SDF     =       *       ;entry
F569  AC6E02               LDY     FINE
F56C  F002 ^F570           BEQ     SDI     ;if not fine scrolling

F56E  0920                 ORA     #$20    ;enable vertical scroll
                   ;       JMP     SDI     ;store data indirect, return



                   **      SDI - Store Data Indirect
                   *
                   *       ENTRY   JSR     SDI
                   *
                   *       MODS
                   *               Original Author Unknown
                   *               1. Bring closer to Coding Standard (object :
                   *                  R. K. Nordin 11/01/83


     = F570        SDI     =       *       ;entry

                   ;       Check current status.

F570  A44C                 LDY     DSTAT   ;status
F572  3028 ^F59F           BMI     DBS3    ;if error, return

                   ;       Store data.

F574  A000                 LDY     #0
F576  9164                 STA     (ADRESS),Y

                   ;       Decrement.

                   ;       JMP     DSD     ;perform double byte single decreme:
                         **     DSD - Perform Double Byte Single Decrement
                         *
                         *      ENTRY    JSR     DSD
                         *
                         *      MODS
                         *               Original Author Unknown
                         *               1. Bring closer to Coding Standard (object :
                         *                  R. K. Nordin 11/01/83


      = F578             DSD    =        *       ;entry
F578  A901                      LDA      #1      ;indicate subtracting 1
                         ;      JMP      DBS     ;perform double byte subtract, retu:



                         **     DBS - Perform Double Byte Subtract
                         *
                         *      ENTRY    JSR     DBS
                         *
                         *      MODS
                         *               Original Author Unknown
                         *               1. Bring closer to Coding Standard (object :
                         *                  R. K. Nordin 11/01/83


      = F57A             DBS    =        *       ;entry

                         ;      Initialize.

F57A  8D9E02                    STA      SUBTMP

                         ;      Check current status.

F57D  A54C                      LDA      DSTAT           ;status
F57F  301E ^F59F                BMI      DBS3            ;if error

                         ;      Subtract.

F581  A564                      LDA      ADRESS
F583  38                        SEC
F584  ED9E02                    SBC      SUBTMP
F587  8564                      STA      ADRESS
F589  B002 ^F58D                BCS      DBS1            ;if no borrow

F58B  C665                      DEC      ADRESS+1        ;adjust high byte

                         ;      Check for overwriting APPMHI.

F58D  A50F               DBS1   LDA      APPMHI+1
F58F  C565                      CMP      ADRESS+1
F591  900C ^F59F                BCC      DBS3            ;if not overwriting APPMHI

F593  D006 ^F59B                BNE      DBS2            ;if overwriting APPMHI, err:

F595  A50E                      LDA      APPMHI
```

```
F597  C564                  CMP     ADRESS
F599  9004 ^F59F            BCC     DBS3            ;if not overwriting APPMHI

                    ;       Process error.

F59B  A993          DBS2    LDA     #SCRMEM         ;indicate insufficient memo:
F59D  854C                  STA     DSTAT           ;status

                    ;       Exit.

F59F  60            DBS3    RTS             ;return



                    **      SSE - Set Scrolling Display List Entry
                    *
                    *       Store extra line in display list for fine scrolling:
                    *
                    *       ENTRY   JSR     SSE
                    *
                    *       MODS
                    *               H. Stewart      06/01/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F5A0        SSE     =       *       ;entry
F5A0  A902                  LDA     #$02
F5A2  2070F5                JSR     SDI     ;store data indirect
F5A5  A9A2                  LDA     #$A2    ;DLI on last visible line
F5A7  2070F5                JSR     SDI     ;store data indirect
F5AA  CA                    DEX
F5AB  60                    RTS             ;return



                    **      CCA - Convert Cursor Row/Column to Address
                    *
                    *       ENTRY   JSR     CCA
                    *
                    *       MODS
                    *               L. Winner       06/01/82
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F5AC        CCA     =       *               ;entry
F5AC  A201                  LDX     #1
F5AE  8666                  STX     MLTTMP          ;initialize
F5B0  CA                    DEX
F5B1  8665                  STX     ADRESS+1        ;clear high address
F5B3  A554                  LDA     ROWCRS          ;cursor row position
F5B5  0A                    ASL     A               ;2 times row position
F5B6  2665                  ROL     ADRESS+1
F5B8  0A                    ASL     A               ;4 time row position
```

```
F5B9  2665                 ROL    ADRESS+1
F5BB  6554                 ADC    ROWCRS         ;add to get 5 times row pos:
F5BD  8564                 STA    ADRESS
F5BF  9002 ^F5C3           BCC    CCA1

F5C1  E665                 INC    ADRESS+1

F5C3  A457         CCA1    LDY    DINDEX         ;mode
F5C5  BE6DEE               LDX    TLSC,Y         ;left shift count

F5C8  0664         CCA2    ASL    ADRESS         ;ADRESS = ADRESS*X
F5CA  2665                 ROL    ADRESS+1       ;divide
F5CC  CA                   DEX
F5CD  D0F9 ^F5C8           BNE    CCA2

F5CF  A556                 LDA    COLCRS+1       ;high cursor column
F5D1  4A                   LSR    A              ;save least significant bit
F5D2  A555                 LDA    COLCRS         ;low cursor column
F5D4  BE9DEE               LDX    TRSC,Y         ;right shift count
F5D7  F006 ^F5DF           BEQ    CCA4           ;if no shift

F5D9  6A           CCA3    ROR    A              ;roll in carry
F5DA  0666                 ASL    MLTTMP         ;shift index
F5DC  CA                   DEX
F5DD  D0FA ^F5D9           BNE    CCA3

F5DF  6564         CCA4    ADC    ADRESS         ;add address
F5E1  9002 ^F5E5           BCC    CCA5           ;if no carry

F5E3  E665                 INC    ADRESS+1       ;adjust high address

F5E5  18           CCA5    CLC
F5E6  6558                 ADC    SAVMSC         ;add saved memory scan count
F5E8  8564                 STA    ADRESS         ;update address
F5EA  855E                 STA    OLDADR         ;save address
F5EC  A565                 LDA    ADRESS+1
F5EE  6559                 ADC    SAVMSC+1
F5F0  8565                 STA    ADRESS+1
F5F2  855F                 STA    OLDADR+1

F5F4  BE9DEE               LDX    TRSC,Y
F5F7  BD04FB               LDA    TMSK,X
F5FA  2555                 AND    COLCRS         ;and in low cursor column
F5FC  6566                 ADC    MLTTMP
F5FE  A8                   TAY
F5FF  B9ACEE               LDA    TDSM-1,Y       ;display mask
F602  8DA002               STA    DMASK          ;display mask
F605  856F                 STA    SHFAMT
F607  A000                 LDY    #0

F609  60           CCA6    RTS                   ;return
```

```
                    **      SZA - Set Zero Data and Advance Cursor Column
                    *
                    *       ENTRY   JSR     SZA
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F60A        SZA     =       *       ;entry
F60A  A900                  LDA     #0
F60C  F002 ^F610            BEQ     SDA     ;set data and advance cursor



                    **      SEA - Set EOL Data and Advance Cursor Column
                    *
                    *       ENTRY   JSR     SEA
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F60E        SEA     =       *       ;entry
F60E  A99B                  LDA     #EOL    ;special case eliminator
                    ;       JMP     SDA     ;set data and advance cursor, retur:



                    **      SDA - Set Data and Advance Cusor Column
                    *
                    *       ENTRY   JSR     SDA
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83


      = F610        SDA     =       *       ;entry
F610  857D                  STA     INSDAT  ;set data
                    ;       JMP     ACC     ;advance cursor column, return
```

```
                        **      ACC - Advance Cursor Column
                        *
                        *       ENTRY   JSR     ACC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


     = F612     ACC     =       *                       ;entry
F612  E663              INC     LOGCOL                  ;increment logical column
F614  E655              INC     COLCRS                  ;increment low cursor colum:
F616  D002 ^F61A        BNE     ACC1                    ;if no carry

F618  E656              INC     COLCRS+1                ;adjust high cursor column

F61A  A555      ACC1    LDA     COLCRS                  ;low cursor column
F61C  A657              LDX     DINDEX                  ;mode
F61E  DD7DEE            CMP     TMCC,X
F621  F00A ^F62D        BEQ     ACC2                    ;if equal, process EOL

F623  E000              CPX     #0
F625  D0E2 ^F609        BNE     CCA6                    ;if not mode 0, exit

F627  C553              CMP     RMARGN                  ;right margin
F629  F0DE ^F609        BEQ     CCA6                    ;if at right margin, exit

F62B  90DC ^F609        BCC     CCA6                    ;if before right margin, ex:

F62D  E008      ACC2    CPX     #8
F62F  D004 ^F635        BNE     ACC3                    ;if not mode 8

F631  A556              LDA     COLCRS+1                ;high cursor column
F633  F0D4 ^F609        BEQ     CCA6                    ;if only at 64

F635  A557      ACC3    LDA     DINDEX                  ;mode
F637  D02C ^F665        BNE     RET                     ;if mode 0, exit

F639  A563              LDA     LOGCOL                  ;logical column
F63B  C951              CMP     #81
F63D  900A ^F649        BCC     ACC4                    ;if < 81, definitely not li:

F63F  A57D              LDA     INSDAT
F641  F022 ^F665        BEQ     RET                     ;if non-zero, do not do log:

F643  2061F6            JSR     RWS                     ;return with scrolling
F646  4CABF6            JMP     RETS                    ;return

F649  2065F6    ACC4    JSR     RET                     ;return
F64C  A554              LDA     ROWCRS                  ;curosr row
F64E  18                CLC
F64F  6978              ADC     #8*[LOGMAP-TABMAP]      ;add offset for log:
F651  205DF7            JSR     BMG                     ;get bit from bit map
F654  9008 ^F65E        BCC     ACC5

F656  A57D              LDA     INSDAT
```

```
F658  F004 ^F65E         BEQ     ACC5            ;if zero, do not extend

F65A  18                 CLC
F65B  2000F5             JSR     ILN1            ;insert line

F65E  4C8EF8     ACC5    JMP     SLC             ;set logical column, return



                 **      RWS - Return with Scrolling
                 *
                 *       ENTRY   JSR     RWS
                 *
                 *       MODS
                 *               Original Author Unknown
                 *               1. Bring closer to Coding Standard (object :
                 *                  R. K. Nordin 11/01/83


      = F661     RWS     =       *       ;entry
F661  A99B               LDA     #EOL    ;select scrolling
F663  857D               STA     INSDAT
                 ;       JMP     RET     ;return, return .



                 **      RET - Return
                 *
                 *       ENTRY   JSR     RET
                 *
                 *       MODS
                 *               Original Author Unknown
                 *               1. Bring closer to Coding Standard (object :
                 *                  R. K. Nordin 11/01/83


      = F665     RET     =       *               ;entry
F665  2097F9             JSR     SCL             ;set cursor at left edge
F668  A900               LDA     #0
F66A  8556               STA     COLCRS+1        ;high cursor column
F66C  E654               INC     ROWCRS          ;increment cursor row
F66E  A657               LDX     DINDEX
F670  A018               LDY     #24             ;assume 24 lines
F672  247B               BIT     SWPFLG
F674  1005 ^F67B         BPL     RET1            ;if normal

F676  A004               LDY     #4              ;substitute 4 lines
F678  98                 TYA
F679  D003 ^F67E         BNE     RET2

F67B  BD8DEE     RET1    LDA     TMRC,X  ;mode row count

F67E  C554       RET2    CMP     ROWCRS  ;cursor row
F680  D029 ^F6AB         BNE     RET5
```

```
F682  8C9D02                STY     HOLD3
F685  8A                    TXA             ;mode
F686  D023 ^F6AB            BNE     RETS    ;if mode not 0, do not scroll

F688  A57D                  LDA     INSDAT
F68A  F01F ^F6AB            BEQ     RETS    ;if zero, do not scroll

                   ;        If EOL, roll in a 0.

F68C  C99B                  CMP     #EOL    ;to extend bottom logical line
F68E  F001 ^F691            BEQ     RET3    ;if EOL

F690  18                    CLC

F691  20F7F7       RET3     JSR     SCR
F694  EE8B02                INC     SCRFLG
F697  C66C                  DEC     BUFSTR
F699  1002 ^F69D            BPL     RET4

F69B  E66C                  INC     BUFSTR

F69D  CE9D02       RET4     DEC     HOLD3
F6A0  ADB202                LDA     LOGMAP
F6A3  38                    SEC             ;indicate for partial line
F6A4  10EB ^F691            BPL     RET3    ;if partial logical line

F6A6  AD9D02                LDA     HOLD3
F6A9  8554                  STA     ROWCRS  ;cursor row

F6AB  4C8EF3       RETS     JMP     SLC     ;set logical column, return



                   **       SEP - Subtract End Point
                   *
                   *        ENTRY   JSR     SEP
                   *                X = 0, if row or 2, if column
                   *
                   *        MODS
                   *                Original Author Unknown
                   *                1. Bring closer to Coding Standard (object :
                   *                   R. K. Nordin 11/01/83


      = F6AE       SEP      =       *               ;entry
F6AE  38                    SEC
F6AF  B570                  LDA     ROWAC,X         ;low value from which to su:
F6B1  E574                  SBC     ENDPT
F6B3  9570                  STA     ROWAC,X         ;new low value
F6B5  B571                  LDA     ROWAC+1,X       ;high value from which to s:
F6B7  E575                  SBC     ENDPT+1
F6B9  9571                  STA     ROWAC+1,X       ;new high value
F6BB  60                    RTS                     ;return
```

```
                  **      CRE - Check Cursor Range for Editor
                  *
                  *       ENTRY   JSR     CRE
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


      = F6BC      CRE     =       *       ;entry

                  ;       Check for mixed mode.

F6BC  AD8F02              LDA     BOTSCR
F6BF  C904                CMP     #4      ;mixed mode indicator
F6C1  F007 ^F6CA          BEQ     CCR     ;if mixed mode, check cursor range,:

                  ;       Check for mode 0.

F6C3  A557                LDA     DINDEX  ;mode
F6C5  F003 ^F6CA          BEQ     CCR     ;if mode 0, check cursor range

                  ;       Open editor.

F6C7  2094EF              JSR     EOP     ;perform editor OPEN
                  ;       JMP     CCR     ;check cursor range, return



                  **      CCR - Check Cursor Range
                  *
                  *       ENTRY   JSR     CCR
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83


      = F6CA      CCR     =       *               ;entry
F6CA  A927                LDA     #39
F6CC  C553                CMP     RMARGN          ;right margin
F6CE  B002 ^F6D2          BCS     CCR1            ;if 39 >= right margin

F6D0  8553                STA     RMARGN          ;set right margin

F6D2  A657        CCR1    LDX     DINDEX
F6D4  BD8DEE              LDA     TMRC,X          ;mode row count
F6D7  C554                CMP     ROWCRS          ;cursor row
F6D9  902A ^F705          BCC     CCR5            ;if count > row position, e:

F6DB  F028 ^F705          BEQ     CCR5            ;if count = row position, e:

F6DD  E008                CPX     #8
F6DF  D00A ^F6EB          BNE     CCR2            ;if not mode 8
```

```
F6E1  A556                   LDA    COLCRS+1        ;high cursor column
F6E3  F013 ^F6F8             BEQ    CCR4            ;if high cursor column zero

F6E5  C901                   CMP    #1
F6E7  D01C ^F705             BNE    CCR5            ;if >1, bad

F6E9  F004 ^F6EF             BEQ    CCR3            ;if 1, check low

F6EB  A556          CCR2     LDA    COLCRS+1        ;high cursor column
F6ED  D016 ^F705             BNE    CCR5            ;if high cursor column non-:

F6EF  BD7DEE        CCR3     LDA    TMCC,X          ;mode column count
F6F2  C555                   CMP    COLCRS          ;low cursor column
F6F4  900F ^F705             BCC    CCR5            ;if count > column position:

F6F6  F00D ^F705             BEQ    CCR5            ;if count = column position:

F6F8  A901          CCR4     LDA    #SUCCES         ;success indicator
F6FA  854C                   STA    DSTAT           ;indicate success
F6FC  A980                   LDA    #BRKABT         ;assume BREAK abort
F6FE  A611                   LDX    BRKKEY          ;BREAK key status
F700  8511                   STA    BRKKEY          ;clear BREAK key status
F702  F006 ^F70A             BEQ    CCR6            ;if BREAK

F704  60                     RTS                    ;return

                    ;        Process range error.

F705  2040F4        CCR5     JSR    CHM             ;move cursor home
F708  A98D                   LDA    #CRSROR         ;indicate cursor overrange

                    ;        Exit.

F70A  854C          CCR6     STA    DSTAT           ;status
F70C  68                     PLA                    ;clean stack for return to :
F70D  68                     PLA
F70E  A578                   LDA    SWPFLG
F710  1003 ^F715             BPL    CCR7            ;if not swapped

F712  4C62F9                 JMP    SWA             ;swap, return

F715  4C1EF2        CCR7     JMP    SST             ;return (to CIO)
```

```
                        **     ROD - Restore Old Data under Cursor
                        *
                        *      ENTRY   JSR     ROD
                        *
                        *      MODS
                        *              Original Author Unknown
                        *              1. Bring closer to Coding Standard (object :
                        *                 R. K. Nordin 11/01/83


      = F718    ROD     =       *               ;entry
F718  A000              LDY     #0
F71A  A55F              LDA     OLDADR+1
F71C  F004 ^F722        BEQ     ROD1            ;if page zero

F71E  A55D              LDA     OLDCHR          ;old data
F720  915E              STA     (OLDADR),Y

F722  60        ROD1    RTS                     ;return



                        **     BMI - Initialize for Bit Map Operation
                        *
                        *      BMI sets the bit mask in BITMSK and byte offset in :
                        *
                        *      ENTRY   JSR     BMI
                        *
                        *      MODS
                        *              Original Author Unknown
                        *              1. Bring closer to Coding Standard (object :
                        *                 R. K. Nordin 11/01/83


      = F723    BMI     =       *       ;entry
F723  48                PHA             ;save logical column
F724  2907              AND     #7      ;logical column modulo 8
F726  AA                TAX             ;offset to bit mask
F727  BDB4EE            LDA     TBTM,X  ;bit mask
F72A  856E              STA     BITMSK  ;set bit mask
F72C  68                PLA             ;logical column
F72D  4A                LSR     A
F72E  4A                LSR     A
F72F  4A                LSR     A       ;logical column divided by 8
F730  AA                TAX             ;offset
F731  60                RTS             ;return
```

```
                            **      BLR - Rotate Logical Line Bit Map Left
                            *
                            *       BLR rotates the logical line bit map left, scrollin:
                            *       logical lines up.
                            *
                            *       ENTRY   JSR     BLR
                            *
                            *       MODS
                            *               Original Author Unknown
                            *               1. Bring closer to Coding Standard (object :
                            *                  R. K. Nordin 11/01/83


      = F732        BLR             =       *               ;entry
F732  2EB402                        ROL     LOGMAP+2
F735  2EB302                        ROL     LOGMAP+1
F738  2EB202                        ROL     LOGMAP
F73B  60                            RTS                     ;return




                            **      BMP - Put Bit in Bit Map
                            *
                            *       PUT CARRY INTO BITMAP
                            *
                            *       ENTRY   JSR     BMP
                            *
                            *       MODS
                            *               Original Author Unknown
                            *               1. Bring closer to Coding Standard (object :
                            *                  R. K. Nordin 11/01/83


      = F73C        BMP             =       *       ;entry
F73C  900C ^F74A                    BCC     BMC     ;if C clear, clear bit in bit map, :

                            ;       JMP     BMS     ;set bit in bit map, return




                            **      BMS - Set Bit in Bit Map
                            *
                            *       ENTRY   JSR     BMS
                            *
                            *       MODS
                            *               Original Author Unknown
                            *               1. Bring closer to Coding Standard (object :
                            *                  R. K. Nordin 11/01/83


      = F73E        BMS             =       *               ;entry
F73E  2023F7                        JSR     BMI             ;initialize for bit mask op:
F741  BDA302                        LDA     TABMAP,X
F744  056E                          ORA     BITMSK          ;set bit
F746  9DA302                        STA     TABMAP,X        ;update bit map
```

```
F749  60                    RTS                     ;return



                    **      BMC - Clear Bit in Bit Map
                    *
                    *       ENTRY   JSR     BMC
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = F74A        BMC     =       *                 ;entry
F74A  2023F7                JSR     BMI               ;initialize for bit mask op:
F74D  A56E                  LDA     BITMSK
F74F  49FF                  EOR     #$FF
F751  3DA302                AND     TABMAP,X          ;clear bit
F754  9DA302                STA     TABMAP,X          ;update bit map
F757  60                    RTS                       ;return



                    **      BLG - Get Bit from Logical Line Bit Map
                    *
                    *       ENTRY   JSR     BLG
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = F758        BLG     =       *       ;entry
F758  A554                  LDA     ROWCRS  ;cursor row
                    ;       JMP     BLG1



                    **      BLG1 - Get Bit from Logical Line Bit Map
                    *
                    *       ENTRY   JSR     BLG1
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = F75A        BLG1    =       *       ;entry
F75A  18                    CLC
                    ;       JMP     BLG2
```

```
                          **      BLG2 - Get Bit from Logical Line Bit Map
                          *
                          *       ENTRY   JSR     BLG2
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


        = F75B    BLG2    =       *       ;entry
F75B    6978              ADC     #8*(LOGMAP-TABMAP)      ;add offset for log:
                          ;       JMP     BMG     ;get bit from bit map, return



                          **      BMG - Get Bit from Bit Map
                          *
                          *       ENTRY   JSR     BMG
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


        = F75D    BMG     =       *       ;entry
F75D    2023F7            JSR     BMI     ;initialize for bit mask operation
F760    18                CLC
F761    BDA302            LDA     TABMAP,X
F764    256E              AND     BITMSK
F766    F001 ^F769        BEQ     BMG1

F768    38                SEC

F769    60        BMG1    RTS             ;return



                          **      CIA - Convert Internal Character to ATASCII
                          *
                          *       ENTRY   JSR     CIA
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


        = F76A    CIA     =       *       ;entry

                          ;       Initialize.

F76A    ADFA02            LDA     CHAR
```

```
                        ;       Check mode.

F76D  A457                      LDY     DINDEX  ;mode
F76F  C00E                      CPY     #14
F771  B017 ^F78A                BCS     CIA2    ;if mode >= 14

F773  C00C                      CPY     #12
F775  B004 ^F77B                BCS     CIA1    ;if mode 12 or 13

F777  C003                      CPY     #3
F779  B00F ^F78A                BCS     CIA2    ;if mode >= 3

                        ;       Convert internal character to ATASCII.

F77B  2A                CIA1    ROL     A
F77C  2A                        ROL     A
F77D  2A                        ROL     A
F77E  2A                        ROL     A
F77F  2903                      AND     #3
F781  AA                        TAX
F782  ADFA02                    LDA     CHAR    ;character
F785  299F                      AND     #$9F    ;strip off cloumn address
F787  1D4DFB                    ORA     TIAC,X  ;or in new column address

                        ;       Exit.

F78A  8DFB02            CIA2    STA     ATACHR  ;ATASCII character

F78D  60                CIA3    RTS             ;return



                        **      MLN - Move Line
                        *
                        *       ENTRY   JSR     MLN
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F78E            MLN     =       *               ;entry

                        ;       Initialize.

F78E  A66A                      LDX     RAMTOP          ;(high) RAM size
F790  CA                        DEX                     ;decrement (high) RAM size
F791  8669                      STX     FRMADR+1        ;high source address
F793  8667                      STX     TOADR+1         ;high destination address
F795  A9B0                      LDA     #low [$0000-80] ;low RAM size - 80
F797  8568                      STA     FRMADR          ;low source address
F799  A9D8                      LDA     #low [$0000-40] ;low RAM size - 40
F79B  8566                      STA     TOADR           ;low destination address
```

```
F79D  A654                   LDX     ROWCRS            ;cursor row

                      ;      Check for completion.

F79F  E8            MLN1     INX
F7A0  ECBF02                 CPX     BOTSCR            ;screen bottom
F7A3  F0E8 ^F78D             BEQ     CIA3              ;if done, return

                      ;      Move line.

F7A5  A027                   LDY     #39               ;offset to last byte

F7A7  B168          MLN2     LDA     (FRMADR),Y        ;byte of source
F7A9  9166                   STA     (TOADR),Y         ;byte of destination
F7AB  88                     DEY
F7AC  10F9 ^F7A7             BPL     MLN2              ;if not done

                      ;      Adjust source and destination addresses.

F7AE  38                     SEC
F7AF  A568                   LDA     FRMADR            ;source address
F7B1  8566                   STA     TOADR             ;update destination address
F7B3  E928                   SBC     #low 40           ;subtract 40
F7B5  8568                   STA     FRMADR            ;update.source address
F7B7  A569                   LDA     FRMADR+1
F7B9  8567                   STA     TOADR+1
F7BB  E900                   SBC     #high 40
F7BD  8569                   STA     FRMADR+1

                      ;      Continue.

F7BF  4C9FF7                 JMP     MLN1              ;continue



                      **     ELL - Extend Logical Line
                      *
                      *      ENTRY   JSR     ELL
                      *
                      *      MODS
                      *              Original Author Unknown
                      *              1. Bring closer to Coding Standard (object :
                      *                 R. K. Nordin 11/01/83


      = F7C2        ELL      =       *         ;entry
F7C2  08                     PHP               ;save bit
F7C3  A016                   LDY     #22

F7C5  98            ELL1     TYA
F7C6  205AF7                 JSR     BLG1
F7C9  08                     PHP
F7CA  98                     TYA
F7CB  18                     CLC
F7CC  6979                   ADC     #8*(LOGMAP-TABMAP)+1     ;add offset for log:
F7CE  28                     PLP
```

```
F7CF  203CF7                 JSR     BMP     ;put bit in bit map
F7D2  88                     DEY
F7D3  3004 ^F7D9             BMI     ELL2

F7D5  C454                   CPY     ROWCRS  ;cursor row
F7D7  B0EC ^F7C5             BCS     ELL1

F7D9  A554          ELL2     LDA     ROWCRS  ;cursor row
F7DB  18                     CLC
F7DC  6978                   ADC     #8*(LOGMAP-TABMAP)      ;add offset for log:
F7DE  28                     PLP
F7DF  4C3CF7                 JMP     BMP     ;put bit in bit map, return



                    **       CLN - Clear Line
                    *
                    *        ENTRY   JSR     CLN
                    *
                    *        MODS
                    *                Original Author Unknown
                    *                1. Bring closer to Coding Standard (object :
                    *                   R. K. Nordin 11/01/83


      = F7E2        CLN      =       *       ;entry
F7E2  A552                   LDA     LMARGN  ;left margin
F7E4  8555                   STA     COLCRS  ;low cursor column
F7E6  20ACF5                 JSR     CCA     ;convert cursor row/column to addre:
F7E9  38                     SEC
F7EA  A553                   LDA     RMARGN  ;right margin
F7EC  E552                   SBC     LMARGN  ;subtract left margin
F7EE  A8                     TAY             ;screen width
F7EF  A900                   LDA     #0

F7F1  9164          CLN1     STA     (ADRESS),Y
F7F3  88                     DEY
F7F4  10FB ^F7F1             BPL     CLN1    ;if not done

F7F6  60                     RTS             ;return



                    **       SCR - Scroll
                    *
                    *        ENTRY   JSR     SCR
                    *
                    *        MODS
                    *                Original Author Unknown
                    *                1. Bring closer to Coding Standard (object :
                    *                   R. K. Nordin 11/01/83


      = F7F7        SCR      =       *       ;entry
```

```
                     ;         Initialize.

F7F7  2032F7                   JSR      BLR      ;rotate logical line bit map left

                     ;         Check for fine scrolling.

F7FA  AD6E02                   LDA      FINE
F7FD  F028 ^F827               BEQ      SCR5     ;if not fine scrolling

F7FF  AD6C02         SCR1      LDA      VSFLAG   ;vertical scroll count
F802  D0FB ^F7FF               BNE      SCR1     ;if prior scroll not yet done

F804  A908                     LDA      #8
F806  8D6C02                   STA      VSFLAG   ;vertical scroll count

                     ;         Wait for scroll to complete.

F809  AD6C02         SCR2      LDA      VSFLAG   ;vertical scroll count
F80C  C901                     CMP      #1       ;start of last scan
F80E  D0F9 ^F809               BNE      SCR2     ;if not done waiting

F810  AD0BD4         SCR3      LDA      VCOUNT
F813  C940                     CMP      #$40
F815  B0F9 ^F810               BCS      SCR3     ;if not done waiting for safe place

F817  A20D                     LDX      #$0D
F819  ADBF02                   LDA      BOTSCR
F81C  C904                     CMP      #4
F81E  D002 ^F822               BNE      SCR4     ;if not split screen

F820  A270                     LDX      #$70

F822  EC0BD4         SCR4      CPX      VCOUNT
F825  B0FB ^F822               BCS      SCR4     ;if not done waiting

                     ;         Exit.

F827  20A6F9         SCR5      JSR      SMS      ;set memory scan counter address
                     ;         JMP      SSD      ;scroll screen for delete, return



                     **        SSD - Scroll Screen for Delete
                     *
                     *         ENTRY    JSR      SSD
                     *
                     *         MODS
                     *                  Original Author Unknown
                     *                  1. Bring closer to Coding Standard (object :
                     *                     R. K. Nordin 11/01/83


       = F82A        SSD       =        *                  ;entry

                     ;         Initialize.
```

```
F82A  A564                 LDA     ADRESS           ;address
F82C  A665                 LDX     ADRESS+1

                    ;      Calculate number of bytes to move.

F82E  E8            SSD1   INX
F82F  E46A                 CPX     RAMTOP
F831  F006 ^F839           BEQ     SSD2             ;if at RAMTOP

F833  38                   SEC
F834  E910                 SBC     #$10
F836  4C2EF8               JMP     SSD1             ;continue

F839  6927          SSD2   ADC     #39              ;(CLC and ADC #40)
F83B  D00A ^F847           BNE     SSD3             ;if byte count non-zero

F83D  A665                 LDX     ADRESS+1
F83F  E8                   INX
F840  E46A                 CPX     RAMTOP
F842  F038 ^F87C           BEQ     SSD6             ;if at RAMTOP

F844  18                   CLC
F845  6910                 ADC     #$10

                    ;      Adjust address.

F847  A8            SSD3   TAY                      ;number of bytes
F848  857E                 STA     COUNTR
F84A  38                   SEC
F84B  A564                 LDA     ADRESS
F84D  E57E                 SBC     COUNTR           ;subtract
F84F  8564                 STA     ADRESS           ;update low address
F851  B002 ^F855           BCS     SSD4             ;if no borrow

F853  C665                 DEC     ADRESS+1         ;adjust high address

                    ;      Move data down.

F855  A564          SSD4   LDA     ADRESS
F857  18                   CLC
F858  6928                 ADC     #40
F85A  857E                 STA     COUNTR           ;address + 40
F85C  A565                 LDA     ADRESS+1
F85E  6900                 ADC     #0
F860  857F                 STA     COUNTR+1

F862  B17E          SSD5   LDA     (COUNTR),Y       ;byte to move
F864  9164                 STA     (ADRESS),Y       ;move byte
F866  C8                   INY
F867  D0F9 ^F862           BNE     SSD5             ;if not done (256-16 times)

F869  A010                 LDY     #256-240
F86B  A564                 LDA     ADRESS
F86D  C9D8                 CMP     #-40
F86F  F00B ^F87C           BEQ     SSD6             ;if all done

F871  18                   CLC
```

```
F872  69F0                      ADC     #240            ;update low address
F874  8564                      STA     ADRESS          ;if no carry
F876  900D ^F855                BCC     SSD4

F878  E665                      INC     ADRESS+1        ;adjust high address
F87A  D0D9 ^F855                BNE     SSD4            ;continue

                        ;       Clear last line.

F87C  A66A              SSD6    LDX     RAMTOP
F87E  CA                        DEX
F87F  867F                      STX     COUNTR+1
F881  A2D8                      LDX     #-40
F883  867E                      STX     COUNTR
F885  A900                      LDA     #0
F887  A027                      LDY     #39

F889  917E              SSD7    STA     (COUNTR),Y      ;clear byte of last line
F88B  88                        DEY
F88C  10FB ^F889                BPL     SSD7            ;if not done

                        ;       JMP     SLC             ;set logical column, return



                        **      SLC - Set Logical Column
                        *
                        *       ENTRY   JSR     SLC
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = F88E            SLC     =       *       ;entry

                        ;       Initialize.

F88E  A900                      LDA     #0
F890  8563                      STA     LOGCOL  ;initialize logical column
F892  A554                      LDA     ROWCRS  ;cursor row
F894  8551                      STA     HOLD1   ;working row

                        ;       Search for beginning of line.

F896  A551              SLC1    LDA     HOLD1   ;add in row component
F898  205AF7                    JSR     BLG1
F89B  B00C ^F8A9                BCS     SLC2    ;if beginning of line found

F89D  A563                      LDA     LOGCOL  ;logical column
F89F  18                        CLC
F8A0  6928                      ADC     #40     ;add number of characters per line
F8A2  8563                      STA     LOGCOL  ;update logical column
F8A4  C651                      DEC     HOLD1   ;decrement working row
F8A6  4C96F8                    JMP     SLC1    ;continue
```

```
                    ;       Add in cursor column.

F8A9  18            SLC2    CLC
F8AA  A563                  LDA     LOGCOL  ;logical column
F8AC  6555                  ADC     COLCRS  ;add low cursor cloumn
F8AE  8563                  STA     LOGCOL  ;update logical column
F8B0  60                    RTS             ;return



                    **      CBC - Compute Buffer Count
                    *
                    *       CBC computes the buffer count as the number of byte:
                    *       buffer start to the end of the logical line (with t:
                    *       spaces removed).
                    *
                    *       ENTRY   JSR     CBC
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83.

      = F8B1        CBC     =       *       ;entry

                    ;       Initialize.

F8B1  204CF9                JSR     SRC             ;save row and column
F8B4  A563                  LDA     LOGCOL          ;logical column
F8B6  48                    PHA                     ;save logical column
F8B7  A56C                  LDA     BUFSTR          ;start of buffer
F8B9  8554                  STA     ROWCRS          ;cursor row
F8BB  A56D                  LDA     BUFSTR+1
F8BD  8555                  STA     COLCRS          ;low cursor column
F8BF  A901                  LDA     #1
F8C1  856B                  STA     BUFCNT          ;initialize buffer count

                    ;       Determine last line on screen.

F8C3  A217          CBC1    LDX     #23             ;normal last line on screen
F8C5  A57B                  LDA     SWPFLG
F8C7  1002 ^F8CB            BPL     CBC2            ;if not swapped

F8C9  A203                  LDX     #3              ;last line on screen

                    ;       Check for cursor on last line of screen.

F8CB  E454          CBC2    CPX     ROWCRS  ;cursor row
F8CD  D00B ^F8DA            BNE     CBC3    ;if cursor on last line

F8CF  A555                  LDA     COLCRS  ;low cursor column
F8D1  C553                  CMP     RMARGN  ;right margin
F8D3  D005 ^F8DA            BNE     CBC3    ;if not at right margin
```

```
F8D5  E668                   INC    BUFCNT  ;fake SEA to avoid scrolling
F8D7  4CEAF8                 JMP    CBC4

F8DA  200AF6        CBC3     JSR    SZA     ;set zero data and advance cursor } advance cursor col
F8DD  E668                   INC    BUFCNT
F8DF  A563                   LDA    LOGCOL  ;logical column
F8E1  C552                   CMP    LMARGN  ;left margin
F8E3  D0DE ^F8C3             BNE    CBC1    ;if not yet at left margin

F8E5  C654                   DEC    ROWCRS  ;decrement cursor row
F8E7  2000F4                 JSR    CLF     ;move cursor left

F8EA  208FF1        CBC4     JSR    GDC     ;get data under cursor
F8ED  D017 ^F906             BNE    CBC6    ;if non-zero, quit

F8EF  C668                   DEC    BUFCNT  ;DECREMENT COUNTER
F8F1  A563                   LDA    LOGCOL  ;logical column
F8F3  C552                   CMP    LMARGN  ;left margin
F8F5  F00F ^F906             BEQ    CBC6    ;if beginning of logical line, exit

F8F7  2000F4                 JSR    CLF     ;move cursor left
F8FA  A555                   LDA    COLCRS  ;low cursor column
F8FC  C553                   CMP    RMARGN  ;right margin
F8FE  D002 ^F902             BNE    CBC5    ;if cursor column not right margin

F900  C654                   DEC    ROWCRS  ;decrement cursor row

F902  A568          CBC5     LDA    BUFCNT
F904  D0E4 ^F8EA             BNE    CBC4    ;if BUFCNT non-zero, continue

F906  68            CBC6     PLA            ;saved logical column
F907  8563                   STA    LOGCOL  ;restore logical column
F909  4C57F9                 JMP    RRC     ;restore row and column, return


                    **       SBS - Set Buffer Start and Logical Column
                    *
                    *        ENTRY   JSR      SBS
                    *
                    *        MODS
                    *                Original Author Unknown
                    *                1. Bring closer to Coding Standard (object :
                    *                   R. K. Nordin 11/01/83


      = F90C        SBS      =      *                 ;entry
F90C  208EF8                 JSR    SLC               ;set logical column
F90F  A551                   LDA    HOLD1
F911  856C                   STA    BUFSTR
F913  A552                   LDA    LMARGN            ;left margin
F915  856D                   STA    BUFSTR+1

F917  60            SBS1     RTS                      ;return
```

```
                        **      DQQ - Delete Line
                        *
                        *       ENTRY   JSR     DQQ
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F918          DQQ     =       *       ;entry
F918    A563                    LDA     LOGCOL  ;logical column
F91A    C552                    CMP     LMARGN  ;left margin
F91C    D002 ^F920              BNE     DQQ1    ;if not at left margin

F91E    C654                    DEC     ROWCRS  ;decrement cursor row

F920    208EF8          DQQ1    JSR     SLC     ;set logical column
                        ;       JMP     DWQ



                        **      DWQ - Delete Line
                        *
                        *       ENTRY   JSR     DWQ
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F923          DWQ     =       *       ;entry

                        ;       Check for left margin.

F923    A563                    LDA     LOGCOL  ;logical column
F925    C552                    CMP     LMARGN  ;left margin
F927    F0EE ^F917              BEQ     SBS1    ;if at left margin, return

F929    20ACF5                  JSR     CCA     ;convert cursor row/column to addre:
F92C    A553                    LDA     RMARGN  ;right margin
F92E    38                      SEC
F92F    E552                    SBC     LMARGN  ;subtract left margin
F931    A8                      TAY             ;offset to last byte

F932    B164            DWQ1    LDA     (ADRESS),Y
F934    D0E1 ^F917              BNE     SBS1

F936    88                      DEY
F937    10F9 ^F932              BPL     DWQ1    ;if not done

F939    4C27F5                  JMP     DLN1    ;delete line, return
```

```
                           **      CCC - Check for Control Character
                           *
                           *       ENTRY   JSR     CCC
                           *
                           *       MODS
                           *               Original Author Unknown
                           *               1. Bring closer to Coding Standard (object :
                           *                  R. K. Nordin 11/01/83


          = F93C          CCC      =       *         ;entry

F93C  A220                         LDX     #TCCRL-3          ;offset to last entry

F93E  BD00FB              CCC1     LDA     TCCR,X            ;control character
F941  CDFB02                       CMP     ATACHR            ;ATASCII character
F944  F005 ^F94B                   BEQ     CCC2              ;if character found, exit

F946  CA                           DEX
F947  CA                           DEX
F948  CA                           DEX
F949  10F3 ^F93E                   BPL     CCC1              ;if not done, continue sear:

F94B  60                  CCC2     RTS                       ;return



                           **      SRC - Save Row and Column
                           *
                           *       ENTRY   JSR     SRC
                           *
                           *       MODS
                           *               Original Author Unknown
                           *               1. Bring closer to Coding Standard (object :
                           *                  R. K. Nordin 11/01/83


          = F94C          SRC      =       *                 ;entry
F94C  A202                         LDX     #2                ;offset to last byte

F94E  B554                SRC1     LDA     ROWCRS,X          ;byte of cursor row/column
F950  9DB802                       STA     TMPROW,X          ;save byte of cursor row/co:
F953  CA                           DEX
F954  10F8 ^F94E                   BPL     SRC1              ;if not done

F956  60                           RTS                       ;return
```

```
                     **      RRC - Restore Row and Column
                     *
                     *       ENTRY   JSR     RRC
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


       = F957        RRC     =       *                   ;entry
F957  A202                   LDX     #2                  ;offset to last byte

F959  BDB802         RRC1    LDA     TMPROW,X            ;byte of saved cursor row/c:
F95C  9554                   STA     ROWCRS,X            ;byte of cursor row/column
F95E  CA                     DEX
F95F  10F8 ^F959             BPL     RRC1                ;if not done

F961  60                     RTS                         ;return



                     **      SWA - Swap Cursor Position with Regular Cursor Posi:
                     *
                     *       ENTRY   JSR     SWA
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


       = F962        SWA     =       *                   ;entry

                     ;       Check for split screen.

F962  ADBF02                 LDA     BOTSCR              ;screen bottom
F965  C918                   CMP     #24                 ;normal indicator
F967  F017 ^F980             BEQ     SWA2                ;if not split screen

                     ;       Swap cursor parameters.

F969  A20B                   LDX     #11                 ;offset to last byte

F96B  B554           SWA1    LDA     ROWCRS,X            ;destination cursor paramet:
F96D  48                     PHA                         ;save cursor parameter
F96E  BD9002                 LDA     TXTROW,X            ;source cursor parameter
F971  9554                   STA     ROWCRS,X            ;update destination cursor :
F973  68                     PLA                         ;saved cursor paramater
F974  9D9002                 STA     TXTROW,X            ;update source cursor param:
F977  CA                     DEX
F978  10F1 ^F96B             BPL     SWA1                ;if not done

                     ;       Complement swap flag.

F97A  A57B                   LDA     SWPFLG              ;swap flag
```

```
F97C  49FF                EOR     #$FF           ;complement swap flag
F97E  857B                STA     SWPFLG         ;update swap flag

                  ;       Exit.

F980  4C1EF2      SWA2    JMP     SST      ;perform screen STATUS, return



                  **      SKC - Sound Key Click
                  *
                  *       ENTRY   JSR     SKC
                  *
                  *       MODS
                  *               Original Author Unknown
                  *               1. Bring closer to Coding Standard (object :
                  *                  R. K. Nordin 11/01/83

      = F983      SKC     =       *        ;entry

                  ;       Initialize.

F983  A27E                LDX     #2*63    ;2 times trip count
F985  48                  PHA              ;save A

                  ;       Turn loudspeaker on.

F986  8E1FD0      SKC1    STX     CONSOL   ;turn loudspeaker on

                  ;       Wait for VBLANK (loudspeaker off).

F989  AD0BD4              LDA     VCOUNT   ;vertical line counter

F98C  CD0BD4      SKC2    CMP     VCOUNT   ;current vertical line counter
F98F  F0FB ^F98C          BEQ     SKC2     ;if vertical line not changed

                  ;       Decrement and check trip count.

F991  CA                  DEX
F992  CA                  DEX
F993  10F1 ^F986          BPL     SKC1     ;if not done

                  ;       Exit.

F995  68                  PLA              ;restore A
F996  60                  RTS              ;return
```

```
                        **      SCL - Set Cursor at Left Edge
                        *
                        *       ENTRY   JSR     SCL
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = F997            SCL     =       *       ;entry

F997  A900                      LDA     #0      ;assume 0

F999  A67B                      LDX     SWPFLG  ;swap flag
F99B  D004 ^F9A1                BNE     SCL1    ;if not swapped

F99D  A657                      LDX     DINDEX  ;mode
F99F  D002 ^F9A3                BNE     SCL2    ;if not mode 0

F9A1  A552              SCL1    LDA     LMARGN  ;use left margin instead of 0

F9A3  8555              SCL2    STA     COLCRS  ;set low cursor column
F9A5  60                        RTS             ;return



                        **      SMS - Set Memory Scan Counter Address
                        *
                        *       ENTRY   JSR     SMS
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = F9A6            SMS     =       *               ;entry
F9A6  A558                      LDA     SAVMSC          ;saved low memory scan coun:
F9A8  8564                      STA     ADRESS          ;set low address
F9AA  A559                      LDA     SAVMSC+1        ;saved high memory scan cou:
F9AC  8565                      STA     ADRESS+1        ;set high address
F9AE  60                        RTS                     ;return
```

```
                        **      SSP - Perform Screen SPECIAL
                        *
                        *       SSP draws a line from OLDROW/OLDCOL to NEWROW/NEWCO:
                        *
                        *       ENTRY   JSR     SSP
                        *
                        *       MODS
                        *               A. Miller
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


        = F9AF          SSP     =       *               ;entry

                        ;       Determine command.

F9AF  A200                      LDX     #0              ;assume no fill
F9B1  A522                      LDA     ICCOMZ          ;command
F9B3  C911                      CMP     #$11            ;DRAW command
F9B5  F008 ^F9BF                BEQ     SSP2            ;if DRAW command

F9B7  C912                      CMP     #$12            ;FILL command
F9B9  F003 ^F9BE                BEQ     SSP1            ;if FILL command

F9BB  A084                      LDY     #NVALID         ;invalid command error
F9BD  60                        RTS                     ;return

F9BE  E8                SSP1    INX                     ;indicate fill

F9BF  8EB702            SSP2    STX     FILFLG          ;right fill flag

                        ;       Set destination row/coulmn.

F9C2  A554                      LDA     ROWCRS          ;cursor row
F9C4  8DF502                    STA     NEWROW
F9C7  A555                      LDA     COLCRS          ;cursor column
F9C9  8DF602                    STA     NEWCOL
F9CC  A556                      LDA     COLCRS+1
F9CE  8DF702                    STA     NEWCOL+1

                        ;       Compute row increment and difference.

F9D1  A901                      LDA     #1              ;assume increment +1
F9D3  8DF802                    STA     ROWINC          ;row increment
F9D6  8DF902                    STA     COLINC          ;column increment
F9D9  38                        SEC
F9DA  ADF502                    LDA     NEWROW          ;destination row
F9DD  E55A                      SBC     OLDROW          ;subtract source row
F9DF  8576                      STA     DELTAR          ;row difference
F9E1  B00E ^F9F1                BCS     SSP3            ;if difference positive

                        ;       Set row increment to -1 and complement row differen:

F9E3  A9FF                      LDA     #$FF            ;increment -1
F9E5  8DF802                    STA     ROWINC          ;update row increment
F9E8  A576                      LDA     DELTAR          ;row difference
F9EA  49FF                      EOR     #$FF
```

```
F9EC  18                  CLC
F9ED  6901                ADC      #1           ;add 1 for 2's complement
F9EF  8576                STA      DELTAR       ;update row difference

                   ;      Compute column increment and difference.

F9F1  38           SSP3   SEC
F9F2  ADF602              LDA      NEWCOL       ;destination column
F9F5  E558                SBC      OLDCOL       ;source column
F9F7  8577                STA      DELTAC       ;column difference
F9F9  ADF702              LDA      NEWCOL+1
F9FC  E55C                SBC      OLDCOL+1
F9FE  8578                STA      DELTAC+1
FA00  B017 ^FA19          BCS      SSP4         ;if difference positive

                   ;      Set column increment to -1 and complement column di:

FA02  A9FF                LDA      #$FF         ;increment -1
FA04  8DF902              STA      COLINC       ;update column increment
FA07  A577                LDA      DELTAC       ;column difference
FA09  49FF                EOR      #$FF         ;absolute value of column d:
FA0B  8577                STA      DELTAC       ;update column difference
FA0D  A578                LDA      DELTAC+1
FA0F  49FF                EOR      #$FF
FA11  8578                STA      DELTAC+1
FA13  E677                INC      DELTAC       ;add 1 for 2's complement
FA15  D002 ^FA19          BNE      SSP4         ;if no carry

FA17  E678                INC      DELTAC+1     ;adjust for 2's complement

                   ;      Set up working row/column and cursor row/column.

FA19  A202         SSP4   LDX      #2           ;offset to last byte
FA1B  A000                LDY      #0
FA1D  8473                STY      COLAC+1      ;zero high working column

FA1F  98           SSP5   TYA
FA20  9570                STA      ROWAC,X      ;zero byte of working row/c:
FA22  B55A                LDA      OLDROW,X     ;byte of source row/column
FA24  9554                STA      ROWCRS,X     ;byte of cursor row/column
FA26  CA                  DEX
FA27  10F6 ^FA1F          BPL      SSP5         ;if not done

                   ;      Determine difference.

FA29  A577                LDA      DELTAC       ;low column difference
FA2B  E8                  INX                   ;offset to working row
FA2C  A8                  TAY                   ;low column difference
FA2D  A578                LDA      DELTAC+1     ;high column difference
FA2F  857F                STA      COUNTR+1     ;initialize high iteration :
FA31  8575                STA      ENDPT+1      ;initialize high end point
FA33  D00B ^FA40          BNE      SSP6         ;if high column difference :

FA35  A577                LDA      DELTAC       ;low column difference
FA37  C576                CMP      DELTAR       ;row difference
FA39  B005 ^FA40          BCS      SSP6         ;if column difference > row:
```

```
FA3B  A576                LDA    DELTAR          ;row difference
FA3D  A202                LDX    #2              ;offset to working column
FA3F  A8                  TAY                    ;row difference

FA40  98          SSP6    TYA                    ;low maximum difference
FA41  857E                STA    COUNTR          ;low iteration counter
FA43  8574                STA    ENDPT           ;low end point
FA45  48                  PHA                    ;save low end point
FA46  A575                LDA    ENDPT+1         ;high end point
FA48  4A                  LSR    A               ;C = LSB of high end point
FA49  68                  PLA                    ;saved low end point
FA4A  6A                  ROR    A
FA4B  9570                STA    ROWAC,X         ;low working row or column

                  ;       Check for iteration counter zero.

FA4D  A57E        SSP7    LDA    COUNTR          ;low iteration counter
FA4F  057F                ORA    COUNTR+1        ;or in high iteration count:
FA51  D003 ^FA56          BNE    SSP8            ;if iteration counter is no:

FA53  4C01F8              JMP    SSP19           ;exit

                  ;       Update working row.

FA56  18          SSP8    CLC
FA57  A570                LDA    ROWAC           ;working row
FA59  6576                ADC    DELTAR          ;row difference
FA5B  8570                STA    ROWAC           ;update working row
FA5D  9002 ^FA61          BCC    SSP9            ;if no carry

FA5F  E671                INC    ROWAC+1         ;adjust high working row

FA61  A571        SSP9    LDA    ROWAC+1         ;high working row
FA63  C575                CMP    ENDPT+1         ;high end point
FA65  9015 ^FA7C          BCC    SSP11           ;if high working row < high:

FA67  D006 ^FA6F          BNE    SSP10           ;if high working row > high:

FA69  A570                LDA    ROWAC           ;low working row
FA6B  C574                CMP    ENDPT           ;low end point
FA6D  900D ^FA7C          BCC    SSP11           ;if low working row < low e:

FA6F  18          SSP10   CLC
FA70  A554                LDA    ROWCRS          ;cursor row
FA72  6DF802              ADC    ROWINC          ;add row increment
FA75  8554                STA    ROWCRS          ;update cursor row
FA77  A200                LDX    #0              ;indicate subtract from wor:
FA79  20AEF6              JSR    SEP             ;subtract end pointer

FA7C  18          SSP11   CLC
FA7D  A572                LDA    COLAC           ;low working column
FA7F  6577                ADC    DELTAC          ;add column difference
FA81  8572                STA    COLAC           ;update working column
FA83  A573                LDA    COLAC+1
FA85  6578                ADC    DELTAC+1
FA87  8573                STA    COLAC+1
FA89  C575                CMP    ENDPT+1         ;high end point
```

```
FA8B  9026 ^FAB5            BCC     SSP15          ;if high working column < h:

FA8D  D006 ^FA95            BNE     SSP12          ;if high working column > h:

FA8F  A572                  LDA     COLAC          ;low working column
FA91  C574                  CMP     ENDPT          ;low end point
FA93  9020 ^FAB5            BCC     SSP15          ;if low working column < lo:

FA95  2CF902       SSP12    BIT     COLINC         ;column increment
FA98  1010 ^FAAA            BPL     SSP13          ;if column increment positi:

FA9A  C655                  DEC     COLCRS         ;decrement low cursor colum:
FA9C  A555                  LDA     COLCRS         ;low cursor column
FA9E  C9FF                  CMP     #$FF
FAA0  D00E ^FAB0            BNE     SSP14

FAA2  A556                  LDA     COLCRS+1       ;high cursor column
FAA4  F00A ^FAB0            BEQ     SSP14          ;if zero, do not decrement

FAA6  C656                  DEC     COLCRS+1       ;decrement high cursor colu:
FAA8  1006 ^FAB0            BPL     SSP14

FAAA  E655         SSP13    INC     COLCRS         ;increment low cursor colum:
FAAC  D002 ^FAB0            BNE     SSP14          ;if no carry

FAAE  E656                  INC     COLCRS+1       ;adjust high cursor column

FAB0  A202         SSP14    LDX     #2             ;indicate subtract from wor:
FAB2  20AEF6                JSR     SEP            ;subtract end pointer

                   ;        Plot point.

FAB5  20CAF6       SSP15    JSR     CCR            ;check cursor range
FAB8  20CAF1                JSR     PLO            ;plot point

                   ;        Check for right fill.

FABB  AD8702                LDA     FILFLG         ;right fill flag
FABE  F02F ^FAEF            BEQ     SSP18          ;if no right fill

                   ;        Process right fill.

FAC0  204CF9                JSR     SRC            ;save row and column
FAC3  ADFB02                LDA     ATACHR         ;plot point
FAC6  8DBC02                STA     HOLD4          ;save plot point

FAC9  A554         SSP16    LDA     ROWCRS         ;cursor row
FACB  48                    PHA                    ;save cursor row
FACC  2012F6                JSR     ACC            ;advance cursor column
FACF  68                    PLA                    ;saved cursor row
FAD0  8554                  STA     ROWCRS         ;restore cursor row
FAD2  20CAF6                JSR     CCR            ;check cursor range
FAD5  208FF1                JSR     GDC            ;get data under cursor
FAD8  D00C ^FAE6            BNE     SSP17          ;if non-zero data encounter:

FADA  ADFD02                LDA     FILDAT         ;fill data
FADD  8DFB02                STA     ATACHR         ;plot point
```

```
FAE0  20CAF1                 JSR      PLO                ;plot point
FAE3  4CC9FA                 JMP      SSP16              ;continue

FAE6  AD8C02       SSP17     LDA      HOLD4              ;saved plot point
FAE9  8DFB02                 STA      ATACHR             ;restore plot point
FAEC  2057F9                 JSR      RRC                ;restore row and column

                   ;         Subtract 1 from iteration counter.

FAEF  38           SSP18     SEC
FAF0  A57E                   LDA      COUNTR             ;iteration counter
FAF2  E901                   SBC      #1                 ;subtract 1
FAF4  857E                   STA      COUNTR             ;update iteration counter
FAF6  A57F                   LDA      COUNTR+1
FAF8  E900                   SBC      #0
FAFA  857F                   STA      COUNTR+1

                   ;         Check for completion.

FAFC  3003 ^FB01             BMI      SSP19     ;if iteration counter negative, exit

FAFE  4C4DFA                 JMP      SSP7      ;continue

                   ;         Exit.

FB01  4C1EF2       SSP19     JMP      SST       ;perform screen STATUS, return



                   **        TMSK - Table of Bit Masks


FB04  00           TMSK      DB       $00       ;0 - mask for no bits
FB05  01                     DB       $01       ;1 - mask for lower 1 bit
FB06  03                     DB       $03       ;2 - mask for lower 2 bits
FB07  07                     DB       $07       ;3 - mask for lower 3 bits



                   **        TDSC - Table of Default Screen Colors


FB08  28           TDSC      DB       $28       ;default playfield 0 color
FB09  CA                     DB       $CA       ;default playfield 1 color
FB0A  94                     DB       $94       ;default playfield 2 color
FB0B  46                     DB       $46       ;default playfield 3 color
FB0C  00                     DB       $00       ;default background color
```

```
                        **      TCCR - Table of Control Character Routines
                        *
                        *       Each entry is 3 bytes.  The first byte is the contr
                        *       character; the second and third bytes are the addre
                        *       the routine which processes the control character.

FB0D  1B          TCCR    DB      $1B
FB0E  E0F3                DW      ESC       ;escape

FB10  1C                  DB      $1C
FB11  E6F3                DW      CUP       ;move cursor up

FB13  1D                  DB      $1D
FB14  F3F3                DW      CDN       ;move cursor down

FB16  1E                  DB      $1E
FB17  00F4                DW      CLF       ;move cursor left

FB19  1F                  DB      $1F
FB1A  11F4                DW      CRT       ;move  cursor right

FB1C  7D                  DB      $7D
FB1D  20F4                DW      CSC       ;clear screen .

FB1F  7E                  DB      $7E
FB20  50F4                DW      BSP       ;backspace

FB22  7F                  DB      $7F
FB23  7AF4                DW      TAB       ;tab

FB25  9B                  DB      $9B
FB26  61F6                DW      RWS       ;return with scrolling

FB28  9C                  DB      $9C
FB29  20F5                DW      DLN       ;delete line

FB2B  9D                  DB      $9D
FB2C  0CF5                DW      ILN       ;insert line

FB2E  9E                  DB      $9E
FB2F  9AF4                DW      CTB       ;clear tab

FB31  9F                  DB      $9F
FB32  95F4                DW      STB       ;set tab

FB34  FD                  DB      $FD
FB35  56F5                DW      BEL       ;sound bell

FB37  FE                  DB      $FE
FB38  D5F4                DW      DCH       ;delete character

FB3A  FF                  DB      $FF
FB3B  9FF4                DW      ICH       ;insert character

      = 0030      TCCRL   =       *-TCCR    ;length
```

```
                        **      TSFR - Table of Super Function (Shifted Function Ke
                        *
                        *       Each entry is 3 bytes.  The first byte is the super
                        *       character; the second and third bytes are the addre
                        *       routine which processes the super function.


FB3D  1C                TSFR    DB      $1C
FB3E  40F4                      DW      CHM      ;move cursor home

FB40  1D                        DB      $1D
FB41  5FF5                      DW      CBT      ;move cursor to bottom

FB43  1E                        DB      $1E
FB44  1BF4                      DW      CLM      ;move cursor to left margin

FB46  1F                        DB      $1F
FB47  0AF4                      DW      CRM      ;move cursor to right margin



                        **      TAIC - Table of ATASCII to Internal Conversion Cons


FB49  40                TAIC    DB      $40      ;0
FB4A  00                        DB      $00      ;1
FB4B  20                        DB      $20      ;2
FB4C  60                        DB      $60      ;3



                        **      TIAC - Table of Internal to ATASCII Conversion Cons


FB4D  20                TIAC    DB      $20      ;0
FB4E  40                        DB      $40      ;1
FB4F  00                        DB      $00      ;2
FB50  60                        DB      $60      ;3



                        **      TCKD - Table of Character Key Definitions
                        *
                        *       Entry n is the ATASCII equivalent of key code n.


      = FB51            TCKD    =       *

                        ;       Lower Case Characters

FB51  6C                        DB      $6C      ;$00 - l
FB52  6A                        DB      $6A      ;$01 - j
FB53  3B                        DB      $3B      ;$02 - semicolon
FB54  8A                        DB      $8A      ;$03 - F1
```

```
FB55  8B            DB       $8B        ;$04 - F2
FB56  6B            DB       $6B        ;$05 - k
FB57  2B            DB       $2B        ;$06 - +
FB58  2A            DB       $2A        ;$07 - *
FB59  6F            DB       $6F        ;$08 - o
FB5A  80            DB       $80        ;$09 - (invalid)
FB5B  70            DB       $70        ;$0A - p
FB5C  75            DB       $75        ;$0B - u
FB5D  9B            DB       $9B        ;$0C - return
FB5E  69            DB       $69        ;$0D - i
FB5F  2D            DB       $2D        ;$0E - -
FB60  3D            DB       $3D        ;$0F - =

FB61  76            DB       $76        ;$10 - v
FB62  80            DB       $80        ;$11 - (invalid)
FB63  63            DB       $63        ;$12 - c
FB64  8C            DB       $8C        ;$13 - F3
FB65  8D            DB       $8D        ;$14 - F4
FB66  62            DB       $62        ;$15 - b
FB67  78            DB       $78        ;$16 - x
FB68  7A            DB       $7A        ;$17 - z
FB69  34            DB       $34        ;$18 - 4
FB6A  80            DB       $80        ;$19 - (invalid)
FB6B  33            DB       $33        ;$1A - 3
FB6C  36            DB       $36        ;$1B - 6
FB6D  1B            DB       $1B        ;$1C - escape
FB6E  35            DB       $35        ;$1D - 5
FB6F  32            DB       $32        ;$1E - 2
FB70  31            DB       $31        ;$1F - 1

FB71  2C            DB       $2C        ;$20 - comma
FB72  20            DB       $20        ;$21 - space
FB73  2E            DB       $2E        ;$22 - period
FB74  6E            DB       $6E        ;$23 - n
FB75  80            DB       $80        ;$24 - (invalid)
FB76  6D            DB       $6D        ;$25 - m
FB77  2F            DB       $2F        ;$26 - /
FB78  81            DB       $81        ;$27 - inverse
FB79  72            DB       $72        ;$28 - r
FB7A  80            DB       $80        ;$29 - (invalid)
FB7B  65            DB       $65        ;$2A - e
FB7C  79            DB       $79        ;$2B - y
FB7D  7F            DB       $7F        ;$2C - tab
FB7E  74            DB       $74        ;$2D - t
FB7F  77            DB       $77        ;$2E - w
FB80  71            DB       $71        ;$2F - q

FB81  39            DB       $39        ;$30 - 9
FB82  80            DB       $80        ;$31 - (invalid)
FB83  30            DB       $30        ;$32 - 0
FB84  37            DB       $37        ;$33 - 7
FB85  7E            DB       $7E        ;$34 - backspace
FB86  38            DB       $38        ;$35 - 8
FB87  3C            DB       $3C        ;$36 - <
FB88  3E            DB       $3E        ;$37 - >
FB89  66            DB       $66        ;$38 - f
FB8A  68            DB       $68        ;$39 - h
```

```
FB8B  64                DB      $64     ;$3A - d
FB8C  80                DB      $80     ;$3B - (invalid)
FB8D  82                DB      $82     ;$3C - CAPS
FB8E  67                DB      $67     ;$3D - g
FB8F  73                DB      $73     ;$3E - s
FB90  61                DB      $61     ;$3F - a

              ;         Upper Case Characters

FB91  4C                DB      $4C     ;$40 - L
FB92  4A                DB      $4A     ;$41 - J
FB93  3A                DB      $3A     ;$42 - colon
FB94  8A                DB      $8A     ;$43 - SHIFT-F1
FB95  8B                DB      $8B     ;$44 - SHIFT-F2
FB96  4B                DB      $4B     ;$45 - K
FB97  5C                DB      $5C     ;$46 - \
FB98  5E                DB      $5E     ;$47 - ^
FB99  4F                DB      $4F     ;$48 - O
FB9A  80                DB      $80     ;$49 - (invalid)
FB9B  50                DB      $50     ;$4A - P
FB9C  55                DB      $55     ;$4B - U
FB9D  9B                DB      $9B     ;$4C - SHIFT-return
FB9E  49                DB      $49     ;$4D - I
FB9F  5F                DB      $5F     ;$4E - _
FBA0  7C                DB      $7C     ;$4F - |

FBA1  56                DB      $56     ;$50 - V
FBA2  80                DB      $80     ;$51 - (invalid)
FBA3  43                DB      $43     ;$52 - C
FBA4  8C                DB      $8C     ;$53 - SHIFT-F3
FBA5  8D                DB      $8D     ;$54 - SHIFT-F4
FBA6  42                DB      $42     ;$55 - B
FBA7  58                DB      $58     ;$56 - X
FBA8  5A                DB      $5A     ;$57 - Z
FBA9  24                DB      $24     ;$58 - $
FBAA  80                DB      $80     ;$59 - (invalid)
FBAB  23                DB      $23     ;$5A - #
FBAC  26                DB      $26     ;$5B - &
FBAD  1B                DB      $1B     ;$5C - SHIFT-escape
FBAE  25                DB      $25     ;$5D - %
FBAF  22                DB      $22     ;$5E - "
FBB0  21                DB      $21     ;$5F - !

FBB1  5B                DB      $5B     ;$60 - [
FBB2  20                DB      $20     ;$61 - SHIFT-space
FBB3  5D                DB      $5D     ;$62 - ]
FBB4  4E                DB      $4E     ;$63 - N
FBB5  80                DB      $80     ;$64 - (invalid)
FBB6  4D                DB      $4D     ;$65 - M
FBB7  3F                DB      $3F     ;$66 - ?
FBB8  81                DB      $81     ;$67 - SHIFT-inverse
FBB9  52                DB      $52     ;$68 - R
FBBA  80                DB      $80     ;$69 - (invalid)
FBBB  45                DB      $45     ;$6A - E
FBBC  59                DB      $59     ;$6B - Y
FBBD  9F                DB      $9F     ;$6C - SHIFT-tab
FBBE  54                DB      $54     ;$6D - T
```

```
FBBF  57                    DB      $57      ;$6E - W
FBC0  51                    DB      $51      ;$6F - Q

FBC1  28                    DB      $28      ;$70 - (
FBC2  80                    DB      $80      ;$71 - (invalid)
FBC3  29                    DB      $29      ;$72 - )
FBC4  27                    DB      $27      ;$73 - '
FBC5  9C                    DB      $9C      ;$74 - SHIFT-delete
FBC6  40                    DB      $40      ;$75 - @
FBC7  7D                    DB      $7D      ;$76 - SHIFT-clear
FBC8  9D                    DB      $9D      ;$77 - SHIFT-insert
FBC9  46                    DB      $46      ;$78 - F
FBCA  48                    DB      $48      ;$79 - H
FBCB  44                    DB      $44      ;$7A - D
FBCC  80                    DB      $80      ;$7B - (invalid)
FBCD  83                    DB      $83      ;$7C - SHIFT-CAPS
FBCE  47                    DB      $47      ;$7D - G
FBCF  53                    DB      $53      ;$7E - S
FBD0  41                    DB      $41      ;$7F - A

                  ;         Control Characters

FBD1  0C                    DB      $0C      ;$80 - CTRL-L
FBD2  0A                    DB      $0A      ;$81 - CTRL-J  .
FBD3  7B                    DB      $7B      ;$82 - CTRL-semicolon
FBD4  80                    DB      $80      ;$83 - (invalid)
FBD5  80                    DB      $80      ;$84 - (invalid)
FBD6  0B                    DB      $0B      ;$85 - CTRL-K
FBD7  1E                    DB      $1E      ;$86 - CTRL-left arrow
FBD8  1F                    DB      $1F      ;$87 - CTRL-right arrow
FBD9  0F                    DB      $0F      ;$88 - CTRL-O
FBDA  80                    DB      $80      ;$89 - (invalid)
FBDB  10                    DB      $10      ;$8A - CTRL-P
FBDC  15                    DB      $15      ;$8B - CTRL-U
FBDD  9B                    DB      $9B      ;$8C - CTRL-return
FBDE  09                    DB      $09      ;$8D - CTRL-I
FBDF  1C                    DB      $1C      ;$8E - CTRL-up arrow
FBE0  1D                    DB      $1D      ;$8F - CTRL-down arrow

FBE1  16                    DB      $16      ;$90 - CTRL-V
FBE2  80                    DB      $80      ;$91 - (invalid)
FBE3  03                    DB      $03      ;$92 - CTRL-C
FBE4  89                    DB      $89      ;$93 - CTRL-F3
FBE5  80                    DB      $80      ;$94 - (invalid)
FBE6  02                    DB      $02      ;$95 - CTRL-B
FBE7  18                    DB      $18      ;$96 - CTRL-X
FBE8  1A                    DB      $1A      ;$97 - CTRL-Z
FBE9  80                    DB      $80      ;$98 - (invalid)
FBEA  80                    DB      $80      ;$99 - (invalid)
FBEB  85                    DB      $85      ;$9A - CTRL-3
FBEC  80                    DB      $80      ;$9B - (invalid)
FBED  1B                    DB      $1B      ;$9C - CTRL-escape
FBEE  80                    DB      $80      ;$9D - (invalid)
FBEF  FD                    DB      $FD      ;$9E - CTRL-2
FBF0  80                    DB      $80      ;$9F - (invalid)

FBF1  00                    DB      $00      ;$A0 - CTRL-comma
```

```
FBF2  20              DB      $20         ;$A1 - CTRL-space
FBF3  60              DB      $60         ;$A2 - CTRL-period
FBF4  0E              DB      $0E         ;$A3 - CTRL-N
FBF5  80              DB      $80         ;$A4 - (invalid)
FBF6  0D              DB      $0D         ;$A5 - CTRL-M
FBF7  80              DB      $80         ;$A6 - (invalid)
FBF8  81              DB      $81         ;$A7 - CTRL-inverse
FBF9  12              DB      $12         ;$A8 - CTRL-R
FBFA  80              DB      $80         ;$A9 - (invalid)
FBFB  05              DB      $05         ;$AA - CTRL-E
FBFC  19              DB      $19         ;$AB - CTRL-Y
FBFD  9E              DB      $9E         ;$AC - CTRL-tab
FBFE  14              DB      $14         ;$AD - CTRL-T
FBFF  17              DB      $17         ;$AE - CTRL-W
FC00  11              DB      $11         ;$AF - CTRL-Q

FC01  80              DB      $80         ;$B0 - (invalid)
FC02  80              DB      $80         ;$B1 - (invalid)
FC03  80              DB      $80         ;$B2 - (invalid)
FC04  80              DB      $80         ;$B3 - (invalid)
FC05  FE              DB      $FE         ;$B4 - CTRL-delete
FC06  80              DB      $80         ;$B5 - (invalid)
FC07  7D              DB      $7D         ;$B6 - CTRL-clear
FC08  FF              DB      $FF         ;$B7 - CTRL-insert
FC09  06              DB      $06         ;$B8 - CTRL-F
FC0A  08              DB      $08         ;$B9 - CTRL-H
FC0B  04              DB      $04         ;$BA - CTRL-D
FC0C  80              DB      $80         ;$BB - (invalid)
FC0D  84              DB      $84         ;$BC - CTRL-CAPS
FC0E  07              DB      $07         ;$BD - CTRL-G
FC0F  13              DB      $13         ;$BE - CTRL-S
FC10  01              DB      $01         ;$BF - CTRL-A


                **    TFKD - Table of Function Key Definitions
                *
                *     Entry n is the ATASCII equivalent of adjusted funct:
                *     code n.


FC11  1C        TFKD  DB      $1C         ;0 - F1 key
FC12  1D              DB      $1D         ;1 - F2 key
FC13  1E              DB      $1E         ;2 - F3 key
FC14  1F              DB      $1F         ;3 - F4 key

FC15  8E              DB      $8E         ;4 - SHIFT-F1 key
FC16  8F              DB      $8F         ;5 - SHIFT-F2 key
FC17  90              DB      $90         ;6 - SHIFT-F3 key
FC18  91              DB      $91         ;7 - SHIFT-F4 key
```

```
                        **      KIR - Process Keyboard IRQ
                        *
                        *       ENTRY   JMP     KIR
                        *
                        *       EXIT
                        *               Exits via RTI
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = FC19            KIR     =       *       ;entry

                        ;       Initialize.

FC19  8A                        TXA
FC1A  48                        PHA             ;save X
FC1B  98                        TYA
FC1C  48                        PHA             ;save Y
FC1D  AC01D3                    LDY     PORTB   ;port B memory control
FC20  AD09D2                    LDA     KBCODE  ;keyboard code
FC23  CDF202                    CMP     CH1     ;last key code
FC26  D005 ^FC2D                BNE     KIR1    ;if not last key code

FC28  AEF102                    LDX     KEYDEL  ;keyboard debounce delay
FC2B  D049 ^FC76                BNE     KIR8    ;if delay not expired, treat as bou:

                        ;       Check for CTRL-F1.

FC2D  AE6D02            KIR1    LDX     KEYDIS  ;save keyboard disable flag
FC30  C983                      CMP     #CNTLF1
FC32  D013 ^FC47                BNE     KIR4    ;if not CTRL-F1

                        ;       Process CTRL-F1.

FC34  8A                        TXA             ;keyboard disable flag
FC35  49FF                      EOR     #$FF    ;complement keyboard disable flag
FC37  8D6D02                    STA     KEYDIS  ;update keyboard disable flag
FC3A  D005 ^FC41                BNE     KIR2    ;if keyboard disabled

FC3C  98                        TYA             ;port B memory control
FC3D  0904                      ORA     #$04    ;turn off LED 1
FC3F  D003 ^FC44                BNE     KIR3    ;update port B memory control

FC41  98                KIR2    TYA             ;port B memory control
FC42  29FB                      AND     #$FB    ;turn on LED 1

FC44  A8                KIR3    TAY             ;updated port B memory control
FC45  B026 ^FC6D                BCS     KIR7    ;reset keyboard controls

                        ;       Check keyboard disable.

FC47  8A                KIR4    TXA             ;keyboard disable flag
FC48  D03D ^FC87                BNE     KIR9    ;if keyboard disabled, exit
```

```
                         ;       Get character.

FC4A  AD0902                     LDA     KBCODE  ;keyboard code
FC4D  AA                         TAX             ;character

                         ;       Check for CTRL-1.

FC4E  C99F                       CMP     #CNTL1
FC50  D00A ^FC5C                 BNE     KIR5    ;if not CTRL-1

                         ;       Process CTRL-1.

FC52  ADFF02                     LDA     SSFLAG  ;start/stop flag
FC55  49FF                       EOR     #$FF    ;complement start/stop flag
FC57  8DFF02                     STA     SSFLAG  ;update start/stop flag
FC5A  B011 ^FC6D                 BCS     KIR7    ;make CTRL-1 invisible

                         ;       Check character.

FC5C  293F               KIR5    AND     #$3F    ;mask off shift and control bits
FC5E  C911                       CMP     #HELP
FC60  D02E ^FC90                 BNE     KIR10   ;if not HELP key

                         ;       Process HELP.

FC62  8EDC02                     STX     HELPFG  ;indicate HELP key pressed
FC65  F006 ^FC6D                 BEQ     KIR7    ;reset keyboard controls

                         ;       Process character.

FC67  8EFC02             KIR6    STX     CH      ;key code
FC6A  8EF202                     STX     CH1     ;reset previous key code

                         ;       Reset keyboard controls.

FC6D  A903               KIR7    LDA     #3
FC6F  8DF102                     STA     KEYDEL  ;re-initialize for debounce
FC72  A900                       LDA     #0
FC74  854D                       STA     ATRACT  ;clear attract-mode timer/flag

                         ;       Prepare to exit.

FC76  ADD902             KIR8    LDA     KRPDEL  ;auto-repeat delay
FC79  8D2B02                     STA     SRTIMR  ;reset software key repeat timer
FC7C  AD2F02                     LDA     SDMCTL  ;DMA control
FC7F  D006 ^FC87                 BNE     KIR9    ;if DMA not disabled, exit

FC81  ADDD02                     LDA     DMASAV  ;saved DMA control
FC84  8D2F02                     STA     SDMCTL  ;DMA control

                         ;       Exit.

FC87  8C01D3             KIR9    STY      PORTB  ;update port B memory control
FC8A  68                         PLA             ;saved Y
FC8B  A8                         TAY             ;restore Y
FC8C  68                         PLA             ;saved X
FC8D  AA                         TAX             ;restore X
```

```
FC8E  68                       PLA                ;restore A
FC8F  40                       RTI                ;return

                  ;        Check for CTRL-F2 or CTRL-F4.

FC90  E084          KIR10      CPX      #CNTLF2
FC92  F021 ^FCB5               BEQ      KIR12    ;if CTRL-F2

FC94  E094                     CPX      #CNTLF4
FC96  D0CF ^FC67               BNE      KIR6     ;if not CTRL-F4

                  ;        Process CTRL-F4.

FC98  ADF402                   LDA      CHBAS    ;character set base
FC9B  AE6B02                   LDX      CHSALT   ;character set alternate
FC9E  8D6B02                   STA      CHSALT   ;update character set alternate
FCA1  8EF402                   STX      CHBAS    ;update character set base

FCA4  E0CC                     CPX      #high ICSORG     ;high international charact:
FCA6  F006 ^FCAE               BEQ      KIR11            ;if international character:

FCA8  98                       TYA                ;port B memory control
FCA9  0908                     ORA      #$08     ;turn off LED 2
FCAB  A8                       TAY                ;updated port B memory control
FCAC  D0BF ^FC6D               BNE      KIR7     ;reset keyboard controls

FCAE  98            KIR11      TYA                ;port B memory control
FCAF  29F7                     AND      #$F7     ;turn on LED 2
FCB1  A8                       TAY                ;updated port B memory control
FCB2  4C6DFC                   JMP      KIR7     ;reset keyboard controls

                  ;        Process CTRL-F2.

FCB5  AD2F02        KIR12      LDA      SDMCTL   ;DMA control
FCB8  F0CD ^FC87               BEQ      KIR9     ;if disabled, exit

FCBA  8DDD02                   STA      DMASAV   ;save DMA state
FCBD  A900                     LDA      #0       ;disable DMA
FCBF  8D2F02                   STA      SDMCTL   ;DMA control
FCC2  F0C3 ^FC87               BEQ      KIR9     ;exit



                  **       FDL - Process Display List Interrupt for Fine Scrol:
                  *
                  *        ENTRY    JMP      FDL
                  *
                  *        EXIT
                  *                 Exits via RTI
                  *
                  *        NOTES
                  *                 Problem: in the CRASS65 version, COLRSH was:
                  *                 zero-page.
                  *                 Problem: in the CRASS65 version, DRKMSK was:
                  *                 zero-page.
                  *
```

```
                        *       MODS
                        *               H. Stewart      06/01/82
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = FCC4    FDL     =       *               ;entry
FCC4  48                PHA                     ;save A
FCC5  ADC602            LDA     COLOR2          ;playfield 2 color
                ;       EOR     COLRSH          ;modify with attract-mode c:
FCC8  4D4F00            VFD     8\$4D,8\low COLRSH,8\high COLRSH
                ;       AND     DRKMSK          ;modify with attract-mode l:
FCCB  2D4E00            VFD     8\$2D,8\low DRKMSK,8\high COLRSH
FCCE  8D0AD4            STA     WSYNC           ;wait for HBLANK synchroniz:
FCD1  8D17D0            STA     COLPF1          ;playfield 1 color/luminanc:
FCD4  68                PLA                     ;restore A
FCD5  40                RTI                     ;return
```

```
FCD6                          FIX     $FCD8



                        **    FCD8 - $FCD8 Patch
                        *
                        *     For compatibility with OS Revision B, sound key cli

FCD8  4C83F9                  JMP     SKC      ;sound key click, return
```

```
FCDB                 **      CIN - Initialize Cassette
                     *
                     *       ENTRY   JSR     CIN
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = FCDB         CIN     =       *               ;entry
 FCDB  A9CC                  LDA     #low B00600     ;indicate 600 baud
 FCDD  8DEE02                STA     CBAUDL          ;cassette baud rate
 FCE0  A905                  LDA     #high B00600
 FCE2  8DEF02                STA     CBAUDH
                     ;       JMP     CSP             ;return



                     **      CSP - Perform Cassette SPECIAL
                     *
                     *       CSP does nothing.
                     *
                     *       ENTRY   JSR     CSP
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = FCE5         CSP     =       *       ;entry
 FCE5  60                    RTS             ;return



                     **      COP - Perform Cassette OPEN
                     *
                     *       ENTRY   JSR     COP
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = FCE6         COP     =       *       ;entry

                     ;       Set cassette IRG type.

 FCE6  A52B                  LDA     ICAX2Z  ;second auxiliary information
 FCE8  853E                  STA     FTYPE   ;cassette IRG type

                     ;       Check OPEN mode.
```

```
FCEA  A52A                 LDA     ICAX1Z  ;OPEN mode
FCEC  290C                 AND     #$0C    ;open for input and output bits
FCEE  C904                 CMP     #$04    ;open for input bit
FCF0  F005 ^FCF7           BEQ     OCI     ;if open for input, process, return

FCF2  C908                 CMP     #$08    ;open for output bit
FCF4  F03E ^FD34           BEQ     OCO     ;if open for output, process, retur:

                     ;     Exit.

FCF6  60                   RTS             ;return



                     **    OCI - Open Cassette for Input
                     *
                     *     ENTRY   JSR     OCI
                     *
                     *     MODS
                     *             Original Author Unknown
                     *             1. Bring closer to Coding Standard (object :
                     *                R. K. Nordin 11/01/83


      = FCF7         OCI   =       *       ;entry

                     ;     Process open for input.

FCF7  A900                 LDA     #0      ;indicate reading
FCF9  8D8902               STA     WMODE   ;WRITE mode
FCFC  853F                 STA     FEOF    ;indicate no EOF yet
FCFE  A901                 LDA     #TONE2  ;tone for pressing PLAY
FD00  20FCFD               JSR     AUB     ;alert user with beep
FD03  3029 ^FD2E           BMI     PBC1    ;if error

                     ;     Initiate cassette READ.

                     ;     JMP     ICR     ;initiate cassette READ, return



                     **    ICR - Initiate Cassette READ
                     *
                     *     ENTRY   JSR     ICR
                     *
                     *     MODS
                     *             Original Author Unknown
                     *             1. Bring closer to Coding Standard (object :
                     *                R. K. Nordin 11/01/83


      = FD05         ICR   =       *               ;entry

                     ;     Initialize.
```

```
FD05  A934                LDA     #MOTRGO         ;motor on
FD07  8D02D3              STA     PACTL           ;port A control

                 ;        Wait for leader read.

FD0A  A662                LDX     PALNTS
FD0C  BC93FE              LDY     RLEADL,X        ;low READ leader
FD0F  BD91FE              LDA     RLEADH,X        ;high READ leader
FD12  AA                  TAX
FD13  A903                LDA     #3
FD15  8D2A02              STA     CDTMF3
FD18  205CE4              JSR     SETVBV          ;set up VBLANK timer

FD1B  AD2A02      ICR1    LDA     CDTMF3
FD1E  D0FB ^FD1B          BNE     ICR1            ;if not done waiting

                 ;        Initialize.

FD20  A980                LDA     #128    ;buffer size
FD22  853D                STA     BPTR    ;initialize buffer pointer
FD24  8D8A02              STA     BLIM    ;initialize buffer limit
FD27  4C77FD              JMP     OC02    ;exit


                 **       PBC - Process BREAK for Cassette Operation
                 *
                 *        ENTRY   JSR     PBC
                 *
                 *        MODS
                 *                Original Author Unknown
                 *                1. Bring closer to Coding Standard (object :
                 *                   R. K. Nordin 11/01/83


      = FD2A      PBC     =       *       ;entry
FD2A  A080                LDY     #BRKABT ;BREAK abort error
FD2C  C611                DEC     BRKKEY  ;reset BREAK key flag

FD2E  A900        PBC1    LDA     #0      ;indicate reading
FD30  8D8902              STA     WMODE   ;WRITE mode
FD33  60                  RTS             ;return
```

```
                          **      OCO - Open Cassette for Output
                          *
                          *       ENTRY   JSR     OCO
                          *
                          *       MODS
                          *               Original Author Unknown
                          *               1. Bring closer to Coding Standard (object :
                          *                  R. K. Nordin 11/01/83


     = FD34               OCO     =       *       ;entry

                          ;       Initialize.

FD34  A980                        LDA     #$80    ;indicate writing
FD36  8D8902                      STA     WMODE   ;WRITE mode
FD39  A902                        LDA     #TONE1
FD3B  20FCFD                      JSR     AUB     ;alert user with beep
FD3E  30EE ^FD2E                  BMI     PBC1    ;if error

                          ;       Set baud rate to 600.

FD40  A9CC                        LDA     #low B00600     ;600 baud
FD42  8D04D2                      STA     AUDF3
FD45  A905                        LDA     #high B00600
FD47  8D06D2                      STA     AUDF4

                          ;       Write marks.

FD4A  A960                        LDA     #$60
FD4C  8D0003                      STA     DDEVIC
FD4F  2068E4                      JSR     SENDEV
FD52  A934                        LDA     #MOTRGO         ;write 5 second blank tape
FD54  8D02D3                      STA     PACTL

                          ;       Wait for leader written.

FD57  A662                        LDX     PALNTS
FD59  BC8FFE                      LDY     WLEADL,X
FD5C  BD8DFE                      LDA     WLEADH,X
FD5F  AA                          TAX
FD60  A903                        LDA     #3
FD62  205CE4                      JSR     SETVBV          ;set VBLANK parmaeters
FD65  A9FF                        LDA     #$FF
FD67  8D2A02                      STA     CDTMF3

FD6A  A511                OCO1    LDA     BRKKEY  ;BREAK key flag
FD6C  F0BC ^FD2A                  BEQ     PBC     ;if BREAK during write leader, proc:

FD6E  AD2A02                      LDA     CDTMF3
FD71  D0F7 ^FD6A                  BNE     OCO1    ;if not done waiting

                          ;       Initialize buffer pointer.

FD73  A900                        LDA     #0
FD75  853D                        STA     BPTR    ;buffer pointer
```

```
                      ;       Indicate success.

FD77  A001           0C02     LDY     #SUCCES ;indicate success
FD79  60                      RTS             ;return



                      **      CGB - Perform Cassette GET-BYTE
                      *
                      *       ENTRY   JSR     CGB
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83

      = FD7A          CGB     =       *               ;entry

                      ;       Check for EOF.

FD7A  A53F                    LDA     FEOF            ;EOF flag
FD7C  3033 ^FD81              BMI     RCB3            ;if at EOF already

                      ;       Check for end of buffer.

FD7E  A63D                    LDX     BPTR            ;buffer pointer
FD80  EC8A02                  CPX     BLIM            ;buffer limit
FD83  F008 ^FD8D              BEQ     RCB             ;if end of buffer, read blo:

                      ;       Get next byte.

FD85  BD0004                  LDA     CASBUF+3,X      ;byte
FD88  E63D                    INC     BPTR            ;increment pointer
FD8A  A001                    LDY     #SUCCES         ;indicate success

FD8C  60             CGB1     RTS                     ;return



                      **      RCB - Read Cassette Block
                      *
                      *       ENTRY   JSR     RCB
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83

      = FD8D          RCB     =       *       ;entry

                      ;       Perform READ.

FD8D  A952                    LDA     #'R'    ;read
```

```
FD8F  203FFE                JSR     SCB         ;perform SIO on cassette buffer
FD92  98                    TYA
FD93  30F7 ^FD8C            BMI     CGB1        ;if SIO error

FD95  A900                  LDA     #0
FD97  853D                  STA     BPTR        ;reset pointer
FD99  A280                  LDX     #$80        ;default number of bytes

                     ;      Check for header.

FD9B  ADFF03                LDA     CASBUF+2
FD9E  C9FE                  CMP     #EOT
FDA0  F00D ^FDAF            BEQ     RCB2        ;if header, read again

                     ;      Check for last record.

FDA2  C9FA                  CMP     #DT1
FDA4  D003 ^FDA9            BNE     RCB1                ;if not last data record

FDA6  AE7F04                LDX     CASBUF+130          ;number of bytes

                     ;      Set number of bytes.

FDA9  8E8A02         RCB1   STX     BLIM

                     ;      Perform cassette GET-BYTE.

FDAC  4C7AFD                JMP     CGB                 ;perform cassette GET-BYTE,:

                     ;      Set EOF flag.

FDAF  C63F           RCB2   DEC     FEOF                ;set EOF flag

                     ;      Exit.

FDB1  A088           RCB3   LDY     #EOFERR             ;end of file indicator
FDB3  60                    RTS                         ;return


                     **     CPB - Perform Cassette PUT-BYTE
                     *
                     *      ENTRY   JSR     CPB
                     *
                     *      MODS
                     *              Original Author Unknown
                     *              1. Bring closer to Coding Standard (object :
                     *                 R. K. Nordin 11/01/83

      = FDB4         CPB    =       *                   ;entry

                     ;      Move data to buffer.

FDB4  A63D                  LDX     BPTR                ;buffer pointer
FDB6  9D0004                STA     CASBUF+3,X          ;data
```

```
FDB9  E63D                  INC     BPTR            ;increment buffer pointer
FDBB  A001                  LDY     #SUCCES         ;assume success

                    ;       Check buffer full.

FDBD  E07F                  CPX     #127            ;offset to last byte of buf:
FDBF  F001 ^FDC2            BEQ     CPB1            ;if buffer full

FDC1  60                    RTS                     ;return

                    ;       Write cassette buffer.

FDC2  A9FC          CPB1    LDA     #DTA    ;indicate data record type
FDC4  207CFE                JSR     WCB     ;write cassette buffer
FDC7  A900                  LDA     #0
FDC9  853D                  STA     BPTR    ;reset buffer pointer
FDCB  60                    RTS             ;return



                    **      CST - Perform Cassette STATUS
                    *
                    *       ENTRY   JSR     CST
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = FDCC        CST     =       *       ;entry
FDCC  A001                  LDY     #SUCCES ;indicate success
FDCE  60                    RTS             ;return



                    **      CCL - Perform Cassette CLOSE
                    *
                    *       ENTRY   JSR     CCL
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = FDCF        CCL     =       *       ;entry

                    ;       Cheeck mode.

FDCF  AD8902                LDA     WMODE   ;WRITE mode
FDD2  3008 ^FDDC            BMI     CCL2    ;if writing

                    ;       Process reading.
```

```
FDD4  A001                LDY     #SUCCES ;indicate success

                    ;     Exit.

FDD6  A93C          CCL1  LDA     #MOTRST
FDD8  8D02D3              STA     PACTL   ;stop motor
FDDB  60                  RTS             ;return

                    ;     Process writing.

FDDC  A63D          CCL2  LDX     BPTR             ;buffer pointer
FDDE  F00A ^FDEA          BEQ     CCL3             ;if no data bytes in buffer

FDE0  8E7F04              STX     CASBUF+130       ;number of bytes
FDE3  A9FA                LDA     #DT1             ;indicate data record type
FDE5  207CFE              JSR     WCB              ;write cassette buffer
FDE8  30EC ^FDD6          BMI     CCL1             ;if error, exit

                    ;     Zero buffer.

FDEA  A27F          CCL3  LDX     #127             ;offset to last byte in buf:
FDEC  A900                LDA     #0

FDEE  9D0004        CCL4  STA     CASBUF+3,X       ;zero byte
FDF1  CA                  DEX
FDF2  10FA ^FDEE          BPL     CCL4             ;if not done

                    ;     Write cassette buffer.

FDF4  A9FE                LDA     #EOT    ;indicate EOT record type
FDF6  207CFE              JSR     WCB     ;write cassette buffer

                    ;     Exit.

FDF9  4CD6FD              JMP     CCL1    ;exit



                    **    AUB - Alert User with Beep
                    *
                    *     ON ENTRY A= FREQ
                    *
                    *     ENTRY   JSR     AUB
                    *
                    *     MODS
                    *             Original Author Unknown
                    *             1. Bring closer to Coding Standard (object :
                    *                R. K. Nordin 11/01/83


      = FDFC        AUB   =       *                ;entry

                    ;     Initialize.

FDFC  8540                STA     FREQ             ;frequency
```

```
                         ;       Compute termination time of beep duration.

FDFE  A514          AUB1    LDA     RTCLOK+2          ;current time
FE00  18                    CLC
FE01  A662                  LDX     PALNTS
FE03  7D95FE                ADC     BEEPNX,X          ;add constant for 1 second :
FE06  AA                    TAX                       ;beep duration termination :

                         ;       Turn on speaker.

FE07  A9FF          AUB2    LDA     #$FF
FE09  8D1FD0                STA     CONSOL            ;turn on speaker
FE0C  A900                  LDA     #$00

                         ;       Delay.

FE0E  A0F0                  LDY     #$F0

FE10  88            AUB3    DEY
FE11  D0FD ^FE10            BNE     AUB3              ;if not done delaying

                         ;       Turn off speaker.

FE13  8D1FD0                STA     CONSOL            ;turn off speaker

                         ;       Delay.

FE16  A0F0                  LDY     #$F0

FE18  88            AUB4    DEY
FE19  D0FD ^FE18            BNE     AUB4              ;if not done delaying

                         ;       Check for beep duration termination time.

FE1B  E414                  CPX     RTCLOK+2          ;compare current time
FE1D  D0E8 ^FE07            BNE     AUB2              ;if termination time not re:

FE1F  C640                  DEC     FREQ              ;decrement frequency
FE21  F00E ^FE31            BEQ     AUB6              ;if all done, wait for anot:

                         ;       Compute termination time of beep separation.

FE23  8A                    TXA
FE24  18                    CLC
FE25  A662                  LDX     PALNTS
FE27  7D97FE                ADC     BEEPFX,X          ;add constant
FE2A  AA                    TAX                       ;beep separation terminatio:

                         ;       Wait for termination of beep separation.

FE2B  E414          AUB5    CPX     RTCLOK+2          ;compare current time
FE2D  D0FC ^FE2B            BNE     AUB5              ;if termination time not re:

                         ;       Beep again.

FE2F  F0CD ^FDFE            BEQ     AUB1              ;beep again
```

```
                     ;       Wait for key.

FE31  2036FE         AUB6    JSR     WFK              ;wait for key
FE34  98                     TYA                      ;status
FE35  60                     RTS                      ;return



                     **      WFK - Wait for Key
                     *
                     *       ENTRY   JSR     WFK
                     *
                     *       NOTES
                     *               Problem: bytes wasted by not doing LDA #hig:
                     *               and LDA #low[KGB-1].
                     *               Problem: bytes wasted by this being a subro:
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = FE36         WFK     =       *                ;entry
FE36  AD25E4                 LDA     KEYBDV+5         ;keyboard GET-BYTE routine :
FE39  48                     PHA                      ;put address on stack
FE3A  AD24E4                 LDA     KEYBDV+4
FE3D  48                     PHA
FE3E  60                     RTS                      ;invoke keyboard GET-BYTE r:



                     **      SCB - Perform SIO on Cassette Buffer
                     *
                     *       ENTRY   JSR     SCB
                     *
                     *       NOTES
                     *               Problem: byte wasted by JSR/RTS exit.
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


      = FE3F         SCB     =       *                ;entry
FE3F  8D0203                 STA     DCOMND           ;command
FE42  A900                   LDA     #high 131
FE44  8D0903                 STA     DBYTHI           ;buffer length
FE47  A983                   LDA     #low 131
FE49  8D0803                 STA     DBYTLO
FE4C  A903                   LDA     #high CASBUF
FE4E  8D0503                 STA     DBUFHI           ;buffer address
FE51  A9FD                   LDA     #low CASBUF
FE53  8D0403                 STA     DBUFLO
```

```
FE56  A960                  LDA     #$60            ;cassette bus ID
FE58  8D0003                STA     DDEVIC
FE5B  A900                  LDA     #0
FE5D  8D0103                STA     DUNIT
FE60  A923                  LDA     #35             ;timeout
FE62  8D0603                STA     DTIMLO
FE65  AD0203                LDA     DCOMND          ;command
FE68  A040                  LDY     #GETDAT         ;assume SIO GET-DATA comman:
FE6A  C952                  CMP     #READ
FE6C  F002 ^FE70            BEQ     SCB1            ;if READ command

FE6E  A080                  LDY     #PUTDAT         ;SIO PUT-DATA command

FE70  8C0303       SCB1     STY     DSTATS          ;SIO command
FE73  A53E                  LDA     FTYPE           ;IRG type
FE75  8D0803                STA     DAUX2           ;second auxiliary informati:
FE78  2059E4                JSR     SIOV            ;vector to SIO
FE7B  60                    RTS                     ;return


                   **       WCB - Write Cassette Buffer
                   *
                   *        ENTRY   JSR     WCB
                   *
                   *        NOTES
                   *                Problem: byte wasted by JSR/RTS exit.
                   *
                   *        MODS
                   *                Original Author Unknown
                   *                1. Bring closer to Coding Standard (object :
                   *                   R. K. Nordin 11/01/83


      = FE7C       WCB      =       *               ;entry
FE7C  8DFF03                STA     CASBUF+2        ;record type
FE7F  A955                  LDA     #$55
FE81  8DFD03                STA     CASBUF+0
FE84  8DFE03                STA     CASBUF+1
FE87  A957                  LDA     #'W'            ;write
FE89  203FFE                JSR     SCB             ;perform SIO on cassette bu:
FE8C  60                    RTS                     ;return
```

```
                     **      NTSC/PAL Constant Tables

FE8D  04             WLEADH  DB      high WLEADN      ;high NTSC WRITE file leade:
FE8E  03                     DB      high WLEADP      ;high PAL WRITE file leader

FE8F  80             WLEADL  DB      low WLEADN       ;low NTSC WRITE file leader
FE90  C0                     DB      low WLEADP       ;low PAL WRITE file leader

FE91  02             RLEADH  DB      high RLEADN      ;high NTSC READ file leader
FE92  01                     DB      high RLEADP      ;high PAL READ file leader

FE93  40             RLEADL  DB      low RLEADN       ;low NTSC READ file leader
FE94  E0                     DB      low RLEADP       ;low PAL READ file leader

FE95  1E             BEEPNX  DB      BEEPNN           ;NTSC beep duration
FE96  19                     DB      BEEPNP           ;PAL beep duration

FE97  0A             BEEPFX  DB      BEEPFN           ;NTSC beep separation
FE98  08                     DB      BEEPFP           ;PAL beep separation
```

```
FE99                  **      PIN - Initialize Printer
                      *
                      *       ENTRY   JSR     PIN
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


     = FE99           PIN     =       *       ;entry
FE99  A91E                    LDA     #30     ;30 second timeout
FE9B  8D1403                  STA     PTIMOT  ;printer timeout
FE9E  60                      RTS             ;return



                      **      Printer Handler Address Data
                      *
                      *       NOTES
                      *               Problem: bytes wasted by tables and code.  :
                      *               Immediate instructions should be used.


FE9F  EA02            PSTB    DW      DVSTAT  ;status buffer address

FEA1  C003            PPRB    DW      PRNBUF  ;printer buffer address



                      **      PST - Perform Printer STATUS
                      *
                      *       ENTRY   JSR     PST
                      *
                      *       MODS
                      *               Original Author Unknown
                      *               1. Bring closer to Coding Standard (object :
                      *                  R. K. Nordin 11/01/83


     = FEA3           PST     =       *       ;entry

                      ;       Get status.

FEA3  A904                    LDA     #4      ;4 bytes for status
FEA5  8DDF02                  STA     PBUFSZ  ;buffer size
FEA8  AE9FFE                  LDX     PSTB    ;address of status buffer
FEAB  ACA0FE                  LDY     PSTB+1
FEAE  A953                    LDA     #STATC  ;status command
FEB0  8D0203                  STA     DCOMND  ;command
FEB3  8D0A03                  STA     DAUX1
FEB6  2014FF                  JSR     SDP     ;set up DCB for printer
FEB9  2059E4                  JSR     SIOV    ;vector to SIO
FEBC  3003 ^FEC1              BMI     PSP     ;if error, return
```

```
                        ;       Exit.

FEBE  2044FF                    JSR     STS     ;set printer timeout from status
                        ;       JMP     PSP     ;return



                        **      PSP - Perform Printer SPECIAL
                        *
                        *       PSP does nothing.
                        *
                        *       ENTRY   JSR     PSP
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = FEC1            PSP     =       *       ;entry
FEC1  60                        RTS             ;return



                        **      POP - Perform Printer OPEN
                        *
                        *       ENTRY   JSR     POP
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = FEC2            POP     =       *       ;entry
FEC2  20A3FE                    JSR     PST     ;perform printer STATUS
FEC5  A900                      LDA     #0
FEC7  8DDE02                    STA     PBPNT   ;clear printer buffer pointer
FECA  60                        RTS             ;return



                        **      PPB - Perform Printer PUT-BYTE
                        *
                        *       ENTRY   JSR     PPB
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = FECB            PPB     =       *               ;entry
```

```
                        ;       Initialize.

FECB  48                        PHA                     ;save data
FECC  BD4103                    LDA     ICDNO,X         ;device number
FECF  8521                      STA     ICDNOZ          ;device number
FED1  204BFF                    JSR     PPM             ;process print mode

                        ;       Put data in buffer.

FED4  AEDE02                    LDX     PBPNT           ;printer buffer pointer
FED7  68                        PLA                     ;saved data
FED8  9DC003                    STA     PRNBUF,X        ;put data in buffer
FEDB  E8                        INX

                        ;       Check for buffer full.

FEDC  ECDF02                    CPX     PBUFSZ          ;printer buffer size
FEDF  F015 ^FEF6                BEQ     PPP             ;if buffer full, perform PU:

                        ;       Update printer buffer pointer.

FEE1  8EDE02                    STX     PBPNT           ;printer buffer pointer

                        ;       Check for EOL.

FEE4  C99B                      CMP     #EOL
FEE6  F003 ^FEEB                BEQ     PPB1            ;if EOL, space fill

                        ;       Exit.

FEE8  A001                      LDY     #SUCCES         ;indicate success
FEEA  60                        RTS                     ;return

                        ;       Space fill buffer.

FEEB  A920              PPB1    LDA     #' '            ;indicate space fill
                        ;       JMP     FPB             ;fill printer buffer, retur:


                        **      FPB - Fill Printer Buffer
                        *
                        *       ENTRY   JSR     FPB
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83

      = FEED            FPB     =       *               ;entry

                        ;       Fill printer buffer.

FEED  9DC003            FPB1    STA     PRNBUF,X        ;byte of printer buffer
FEF0  E8                        INX
```

```
FEF1  ECDF02                CPX     PBUFSZ          ;printer buffer size
FEF4  D0F7 ^FEED            BNE     FPB1            ;if not done

                    ;       Perform printer PUT.

                    ;       JMP     PPP             ;perform printer PUT, retur:



                    **      PPP - Perform Printer PUT
                    *
                    *       ENTRY   JSR     PPP
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = FEF6        PPP     =       *       ;entry

                    ;       Clear printer buffer pointer.

FEF6  A900                  LDA     #0
FEF8  8DDE02                STA     PBPNT   ;clear printer buffer pointer

                    ;       Set up DCB.

FEFB  AEA1FE                LDX     PPRB    ;address of printer buffer
FEFE  ACA2FE                LDY     PPRB+1
FF01  2014FF                JSR     SDP     ;set up DCB for printer

                    ;       Perform PUT.

FF04  4C59E4                JMP     SIOV    ;vector to SIO, return



                    **      PCL - Perform Printer CLOSE
                    *
                    *       ENTRY   JSR     PCL
                    *
                    *       MODS
                    *               Original Author Unknown
                    *               1. Bring closer to Coding Standard (object :
                    *                  R. K. Nordin 11/01/83

      = FF07        PCL     =       *       ;entry

                    ;       Initialize.

FF07  204BFF                JSR     PPM     ;process print mode

                    ;       Check buffer pointer.
```

```
FF0A  A99B                    LDA     #EOL     ;indicate EOL fill
FF0C  AEDE02                  LDX     PBPNT    ;printer buffer pointer
FF0F  D0DC ^FEED              BNE     FPB      ;if buffer pointer non-zero, fill b:

                     ;        Exit.

FF11  A001                    LDY     #SUCCES ;indicate success
FF13  60                      RTS              ;return



                     **       SDP - Set Up DCB for Printer
                     *
                     *        ENTRY   JSR     SDP
                     *
                     *        MODS
                     *                Original Author Unknown
                     *                1. Bring closer to Coding Standard (object :
                     *                   R. K. Nordin 11/01/83


      = FF14         SDP      =       *        ;entry
FF14  8E0403                  STX     DBUFLO   ;low buffer address
FF17  8C0503                  STY     DBUFHI   ;high buffer address
FF1A  A940                    LDA     #PDEVN   ;printer device bus ID
FF1C  8D0003                  STA     DDEVIC   ;device bus ID
FF1F  A521                    LDA     ICDNOZ   ;device number
FF21  8D0103                  STA     DUNIT    ;unit number
FF24  A980                    LDA     #$80     ;SIO WRITE command
FF26  AE0203                  LDX     DCOMND   ;I/O direction
FF29  E053                    CPX     #STATC   ;STATUS command
FF2B  D002 ^FF2F              BNE     SDP1     ;if STATUS command

FF2D  A940                    LDA     #$40     ;SIO READ command

FF2F  8D0303         SDP1     STA     DSTATS   ;SIO command
FF32  ADDF02                  LDA     PBUFSZ
FF35  8D0803                  STA     DBYTLO   ;low buffer size
FF38  A900                    LDA     #0
FF3A  8D0903                  STA     DBYTHI   ;high buffer size
FF3D  AD1403                  LDA     PTIMOT
FF40  8D0603                  STA     DTIMLO   ;device timeout
FF43  60                      RTS              ;return
```

```
                        **      STS - Set Printer Timeout from Status
                        *
                        *       ENTRY   JSR     STS
                        *
                        *       NOTES
                        *               Problem: bytes wasted by this code's being :
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = FF44            STS     =       *               ;entry
FF44  ADEC02                    LDA     DVSTAT+2        ;timeout
FF47  8D1403                    STA     PTIMOT          ;set printer timeout
FF4A  60                        RTS                     ;return



                        **      PPM - Process Print Mode
                        *
                        *       PPM sets up the DCB according to the print mode.
                        *
                        *       ENTRY   JSR     PPM
                        *
                        *       MODS
                        *               Original Author Unknown
                        *               1. Bring closer to Coding Standard (object :
                        *                  R. K. Nordin 11/01/83


      = FF4B            PPM     =       *       ;entry

                        ;       Initialize.

FF4B  A057                      LDY     #WRITE  ;WRITE command
FF4D  A52B                      LDA     ICAX2Z  ;print mode

                        ;       Determine buffer size.

FF4F  C94E              PPM1    CMP     #NORMAL ;NORMAL mode
FF51  D004 ^FF57                BNE     PPM2    ;if not NORMAL mode

FF53  A228                      LDX     #NBUFSZ ;NORMAL mode buffer size
FF55  D00E ^FF65                BNE     PPM4    ;set buffer size

FF57  C944              PPM2    CMP     #DOUBLE ;DOUBLE mode
FF59  D004 ^FF5F                BNE     PPM3    ;if not DOUBLE mode

FF5B  A214                      LDX     #DBUFSZ ;DOUBLE mode buffer size
FF5D  D006 ^FF65                BNE     PPM4    ;set buffer size

FF5F  C953              PPM3    CMP     #SIDWAY ;SIDEWAYS mode
FF61  D00C ^FF6F                BNE     PPM5    ;if not SIDEWAYS mode, assume NORMA:
```

```
FF63  A21D                     LDX     #SBUFSZ ;SIDEWAYS mode buffer size

                    ;          Set buffer size.

FF65  8EDF02        PPM4       STX     PBUFSZ  ;set printer buffer size

                    ;          Set DCB command and mode.

FF68  8C0203                   STY     DCOMND  ;command
FF6B  8D0A03                   STA     DAUX1   ;print mode
FF6E  60                       RTS             ;return

                    ;          Assume NORMAL mode.

FF6F  A94E          PPM5       LDA     #NORMAL ;NORMAL mode
FF71  D0DC ^FF4F               BNE     PPM1    ;set buffer size
```

```
FF73                 **      VFR - Verify First 8K ROM
                     *
                     *       ENTRY   JSR     VFR
                     *
                     *       EXIT
                     *               C clear, if verified
                     *                 set, if not verified
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83

     = FF73          VFR     =       *       ;entry

                     ;       Initialize.

FF73  A200                   LDX     #0      ;offset to first region to checksum
FF75  868B                   STX     STCHK   ;initial sum is zero
FF77  868C                   STX     STCHK+1

                     ;       Checksum ROM.

FF79  20A9FF         VFR1    JSR     CRR     ;checksum region of ROM
FF7C  E00C                   CPX     #12
FF7E  D0F9 ^FF79             BNE     VFR1    ;if not done

                     ;       Compare result.

FF80  AD00C0                 LDA     $C000   ;low checksum in ROM
FF83  AE01C0                 LDX     $C001   ;high checksum in ROM
                     ;       JMP     VCS     ;verify checksum, return



                     **      VCS - Verify Checksum
                     *
                     *       ENTRY   JSR     VCS
                     *
                     *       MODS
                     *               Original Author Unknown
                     *               1. Bring closer to Coding Standard (object :
                     *                  R. K. Nordin 11/01/83


     = FF86          VCS     =       *       ;entry
FF86  C58B                   CMP     STCHK   ;low checksum
FF88  D006 ^FF90             BNE     VCS1    ;if low checksum bad

FF8A  E48C                   CPX     STCHK+1 ;high checksum
FF8C  D002 ^FF90             BNE     VCS1    ;if high checksum bad

FF8E  18                     CLC             ;indicate verified
FF8F  60                     RTS             ;return
```

```
FF90  38          VCS1     SEC                 ;indicate not verified
FF91  60                   RTS                 ;return



                  **       VSR - Verify Second 8K ROM
                  *
                  *        ENTRY    JSR      VSR
                  *
                  *        EXIT
                  *                 C clear, if verified
                  *                   set, if not verified
                  *
                  *        MODS
                  *                 Original Author Unknown
                  *                 1. Bring closer to Coding Standard (object :
                  *                    R. K. Nordin 11/01/83

       = FF92     VSR      =        *          ;entry
FF92  A200                 LDX      #0
FF94  868B                 STX      STCHK      ;initial sum is zero
FF96  868C                 STX      STCHK+1
FF98  A20C                 LDX      #12        ;offset to first region to checksum
FF9A  20A9FF               JSR      CRR        ;checksum region of ROM
FF9D  20A9FF               JSR      CRR        ;checksum region of ROM
FFA0  ADF8FF               LDA      $FFF8      ;low checksum from ROM
FFA3  AEF9FF               LDX      $FFF9      ;high checksum from ROM
FFA6  4C86FF               JMP      VCS        ;verify checksum, return



                  **       CRR - Checksum Region of ROM
                  *
                  *        ENTRY    JSR      CRR
                  *                 X = offset
                  *
                  *        MODS
                  *                 Original Author Unknown
                  *                 1. Bring closer to Coding Standard (object :
                  *                    R. K. Nordin 11/01/83

       = FFA9     CRR      =        *          ;entry

                  ;        Transfer range addresses.

FFA9  A000                 LDY      #0

FFAB  BDD7FF      CRR1     LDA      TARV,X
FFAE  999E00               STA      STADR1,Y
FFB1  E8                   INX
FFB2  C8                   INY
FFB3  C004                 CPY      #4         ;4 bytes for 2 addresses
FFB5  D0F4 ^FFAB           BNE      CRR1       ;if not done
```

```
                          ;        Checksum range.

FFB7  A000                         LDY     #0

FFB9  18              CRR2         CLC
FFBA  B19E                         LDA     (STADR1),Y
FFBC  658B                         ADC     STCHK
FFBE  858B                         STA     STCHK
FFC0  9002 ^FFC4                   BCC     CRR3    ;if low value non-zero

FFC2  E68C                         INC     STCHK+1 ;adjust high value

FFC4  E69E            CRR3         INC     STADR1  ;advance address
FFC6  D002 ^FFCA                   BNE     CRR4    ;if low address non-zero

FFC8  E69F                         INC     STADR1+1        ;adjust high address

FFCA  A59E            CRR4         LDA     STADR1  ;current address
FFCC  C5A0                         CMP     STADR2  ;end of range
FFCE  D0E9 ^FFB9                   BNE     CRR2    ;if not done

FFD0  A59F                         LDA     STADR1+1
FFD2  C5A1                         CMP     STADR2+1
FFD4  D0E3 ^FFB9                   BNE     CRR2    ;if not done

FFD6  60                           RTS             ;return


                          **       TARV - Table of Address Ranges to Verify

FFD7  02C000D0        TARV         DW      $C002,$D000     ;first 8K ROM, $C002 - $CFF:
FFDB  00500058                     DW      $5000,$5800     ;first 8K ROM, $D000 - $D7F:
FFDF  00D800E0                     DW      $D800,$E000     ;first 8K ROM, $D800 - $DFF:

FFE3  00E0F8FF                     DW      $E000,$FFF8     ;second 8K ROM, $E000 - $FF:
FFE7  FAFF0000                     DW      $FFFA,$0000     ;second 8K ROM, $FFFA - $FF:
```

```
FFEB                          FIX     $FFEE



                    **        Second 8K ROM Identification and Checksum


FFEE  100583                  DB      IDDAY,IDMON,IDYEAR                ;date (day,:
FFF1  02                      DB      IDCPU                             ;CPU series
FFF2  4242000001              DB      IDPN1,IDPN2,IDPN3,IDPN4,IDPN5     ;part numbe:
FFF7  02                      DB      IDREV                             ;revision n:
FFF8  0000                    DW      $0000                             ;reserved f:
```

```
FFFA                            FIX     $FFFA



                     **         6502 Machine Vectors


FFFA  18C0                      DW      NMI     ;vector to process NMI
FFFC  AAC2                      DW      RES     ;vector to process RESET
FFFE  2CC0                      DW      IRQ     ;vector to process IRQ



0000                            END


no ERRORs, 1783 Labels, $0B57 free.


   ABI     C5A7      62#15   62/54
 n ABUFPT  001C      10#37
   ACB     C66E      53/29   66#47
   ACB1    C678      66/52   67# 7
   ACB2    C69F      66/58   67/ 8   67#29
   ACB3    C6A0      66/60   67#33
   ACC     F612     301/25  311#15  336/53
   ACC1    F61A     311/18  311#22
   ACC2    F62D     311/25  311#35
   ACC3    F635     311/36  311#41
   ACC4    F649     311/46  311#54
   ACC5    F65E     311/59  312/ 5  312#10
   ACK     0041       7#31  240/28
   ACMI    0000       4#14   23/16   30/25   32/41   36/36   39/13   52/19
 n ACMISR  02D7      15#57
   ACMVAR  03ED      17#42   50/42
   ADB     C588      53/42   61#30
   ADB1    C598      61/35   61#49
   ADRESS  0064      11#44  275/27  275/60  277/ 6  277/ 8  277/24  277/25
                    277/28  277/30  279/42  279/44  279/48  279/50  281/17
                    283/53  283/55  288/43  290/ 5  290/ 7  298/ 6  298/ 8
                    298/10  298/14  298/15  302/24  302/26  306/45  307/45
                    307/48  307/51  307/56  308/ 5  308/57  308/60  309/ 5
                    309/ 7  309/10  309/15  309/16  309/31  309/34  309/38
                    309/40  309/42  322/41  324/ 5  324/ 6  324/21  324/34
                    324/36  324/39  324/43  324/47  324/52  324/57  325/ 6
                    325/ 9  328/52  332/45  332/47
   AFP     D800      23#25  138/37
   AFP1    D818     139#26  139/48  139/54  140/ 9  140/27  140/35  140/44
   AFP10   D86C     140#55  141/58
   AFP11   D88E     141/ 6  141#26
   AFP12   D898     141/27  141#39
   AFP13   D8A3     140/51  141#47
   AFP14   D8AD     141/48  141#57
```

```
  AFP15  D8B2     141/51  142# 5
  AFP16  D8B6     140/24  140/40  141/43  142#10
  AFP17  D8C3     142/13  142/15  142#20
  AFP18  D8CE     142/25  142#29
  AFP19  D8E4     142/40  142#50
  AFP2   D81C     139/22  139#31
  AFP20  D8E5     142/35  142#52
  AFP3   D837     139/38  139#58
  AFP4   D83E     139/60  140# 9
  AFP5   D841     139/12  140#13
  AFP6   D842     139/32  140#17
  AFP7   D856     140/30  140#34
  AFP8   D85A     140/18  140#39
  AFP9   D863     140/21  140#48
n ALLPUT D208      21#37
  ANTIC  D400      22#31   58/61
n APPEND 0001       5#34
  APPMHI 000E      10#24   50/32   50/32   50/34   50/34  307/55  307/61
  AST    56AA     131/28  131#52
  AST1   56B5     131/57  131#61
  ATACHR 02FB      16#30  281/44  282/16  282/58  286/ 9  287/33  288/17
                  288/24  288/59  291/45  294/ 8  294/26  294/31  299/37
                  320/32  329/20  336/48  336/61  337/ 9
  ATRACT 004D      11#26   34/26   38/13   38/23   38/26  107/60  119/50
                  345/43
  AUB    FDFC     350/38  352/22  356#56
  AUB1   FDFE     357# 7  357/60
  AUB2   FE07     357#15  357/40
  AUB3   FE10     357#23  357/24
  AUB4   FE18     357#34  357/35
  AUB5   FE28     357#55  357/56
  AUB6   FE31     357/43  358# 7
  AUDC1  D201      21#47  124/44  125/ 5  130/37  251/33  251/46  252/33
  AUDC2  D203      21#50  121/ 6  125/ 6  251/47
  AUDC3  D205      21#53   59/39  125/ 7  251/41
  AUDC4  D207      21#56   59/40  125/ 8
  AUDCTL D208      21#58   59/43  104/23  251/22
  AUDF1  D200      21#46  124/42  130/35  250/21
  AUDF2  D202      21#49  120/57  250/19
  AUDF3  D204      21#52  237/19  248/25  257/29  352/28
  AUDF4  D206      21#55  237/21  248/27  257/31  352/30
  AVV    ED2E     254/40  254/43  255#54
  AVV1   ED36     255/56  256# 5
  B00600 05CC       9#11  248/24  248/26  349/16  349/18  352/27  352/29
  B19200 0028       9#10  237/18  237/20
  BADIOC 0086       6#32  201/33  270/32
  BADMOD 0091       6#43  274/46
  BAI    C5BB      62#42   67/16
  BAI1   C5C0      62#51   64/14   64/28
  BAI2   C5C8      61/41   61/54   62#58
  BASICF 03F8      17#43   51/10   57/11
  BEEPFN 000A       8#15  360/23
  BEEPFP 0008       8#24  360/24
  BEEPFX FE97     357/50  360#23
  BEEPNN 001E       8#14  360/20
  BEEPNP 0019       8#23  360/21
  BEEPNX FE95     357/10  360#20
  BEL     F556     287/43   305#15   338/52
  BEL1    F558     305#18   305/20
  BFENHI  0035      11# 7   237/46   240/18   242/45   246/60   247/45
  BFENLO  0034      11# 6   237/43   240/15   242/43   246/58   247/41
  BIR     C092      34#18    60/21    60/23
n BIR1    C09E      34#30
  BITMSK  006E      11#56   316/45   317/60   318/22   319/36
  BLG     F758     300/ 6   304/42   318#41
  BLG1    F75A     318#58   321/56   325/53
  BLG2    F756     304/21   319#15
  BLIM    028A      15# 6   351/25   353/33   354/28
  BLKBDV  E471      24#43   196/42
  BLR     F732     317#18   323/ 7
  BMC     F74A     300/51   317/40   318#20
  BMG     F75D     300/12   311/58   319#32
  BMG1    F769     319/37   319#41
  BMI     F723     316#40   317/58   318/21   319/33
  BMP     F73C     304/27   317#39   322/ 5   322/16
  BMS     F73E     300/34   317#57
  BMSG    C43D      55#34    65/45    65/46
  BOOT?   0009      10#21    61/39    64/33    66/56    67/20
  BOOTAD  0242      14#12    63/28    63/30    64/54    64/57
  BOTSCR  028F      15#32   277/39   277/59   295/19   295/42   302/36   314/19
                   321/10   323/31   330/42
  BPTR    003D      11#14   351/24   352/60   353/32   353/39   354/10   354/60
                   355/ 5   355/20   356/15
  BRKABT  0080       6#26   258/21   291/ 7   315/25   351/42
  BRKKEY  0011      10#31    34/23    60/19   241/53   244/44   256/34   256/58
                   258/25   291/ 8   292/59   315/26   315/27   351/43   352/51
  BRKKY   0236      13#52    37/14    60/22    60/24
  BSP     F450     299#15   338/31
  BSP1    F45F     299/22   299#26
  BSP2    F46F     299/29   299/32   299#36
  BSP3    F477     299/18   299#40
  BUFADR  0015      10#33    69/48    70/ 6    70/11    70/42    70/44
  BUFCNT  006B      11#54   287/21   287/58   288/ 6   326/43   327/ 5   327/ 9
                   327/20   327/32
  BUFRFL  0038      11# 9   244/34   246/15   247/20
  BUFRHI  0033      11# 5   237/45   240/17   242/38   242/44   246/53   246/59
                   247/43   257/46   257/48
  BUFRLO  0032      10#61   237/41   240/13   241/47   242/35   242/42   243/21
                   246/45   246/50   246/57   247/39   257/37   257/39   257/43
                   257/45
  BUFSTR  006C      11#55   287/27   287/29   287/51   287/53   298/49   298/53
                   313/21   313/24   326/38   326/40   327/55   327/57
  COA     DC70     143/49   144/14   166#41
  COA1    DC76     166#51   167/15
  CAL     C7D2      72/ 6    73#52
  CART    BFFC      19#18    52/51    56/43
  CARTAD  BFFE      19#20    54/28    56/51
  CARTCK  03EB      17#37    58/35    58/36
  CARTCS  BFFA      19#17    54/ 7
  CARTFG  BFFD      19#19    53/36    53/58    56/46
  CASBUF  03FD      17#47    62/22    62/24    63/21    63/33    63/35    63/40
                   223/49   225/11   353/38   354/15   354/24   354/61   356/18
                   356/28   358/58   358/60   359/41   359/43   359/44
  CASET   0060       6#52   236/55   250/12   251/43
```

```
CASETV E440      24#24   55/15   60/47   195/26
CASFLG 030F      16#58  237/ 8  244/29   248/51
CASINI 0002      10#13   67/23   67/25    67/33
CASSBT 03EA      17#36   62/53   64/13    64/20    66/19    67/14    67/18
CASSET 0043       5#47   55/14
CAUX1  023C      13#56  237/33
CAUX2  023D      13#57  237/35
CBAUDH 02EF      16#17  255/38  257/30   349/19
CBAUDL 02EE      16#16  255/35  257/28   349/17
CBC    F8B1     287/50  326#31
CBC1   F8C3     326#47  327/12
CBC2   F8CB     326/49  326#55
CBC3   F8DA     326/56  326/60  327# 8
CBC4   F8EA     327/ 6  327#17  327/33
CBC5   F902     327/28  327#32
CBC6   F906     327/18  327/23  327#35
CBI    C5C9      62/47   63#15
CBI1   C5CB      63#21   63/24
CBI2   C5E8      63#38   64/ 8
CBI3   C5EA      63#40   63/43
CBI4   C607      64# 7   64/16
CBI5   C616      63/58   64#20
CBI6   C61E      64/21   64#27
CBR    ECC8     254#37  257/24
CBR1   ECF7     254#58  254/61
CBR2   ED11     255#21  255/24
CBR3   ED1F     255/29  255#33
CBT    F55F     305#37  339/16
CCA    F5AC     281/16  283/35  302/23   303/36   304/37   308#53   322/34
                328/46
CCA1   F5C3     309/ 8  309#12
CCA2   F5C8     309#15  309/18
CCA3   F5D9     309#26  309/29
CCA4   F5DF     309/24  309#31
CCA5   F5E5     309/32  309#36
CCA6   F609     309#55  311/28  311/31   311/33   311/39
CCC    F93C     289/27  294/23  329#15
CCC1   F93E     329#19  329/26
CCC2   F948     329/21  329#28
CCE    C4C9      46/37   52/55   58#19
CCE1   C4CD      58#29   58/31
CCL    FDCF     195/34  355#53
CCL1   FDD6     356# 9  356/21  356/39
CCL2   FDDC     355/58  356#15
CCL3   FDEA     356/16  356#25
CCL4   FDEE     356#28  356/30
CCO    E672     205/35  206/36  207/17   207/61   209/12   210/46   211#32
CCO1   E683     211#50  211/55
CCOMND 0238      13#55  237/31
CCR    F6CA     280/51  281/54  314/21   314/26   314#46   336/37   336/56
CCR1   F6D2     314/49  314#53
CCR2   F6E8     314/61  315#14
CCR3   F6EF     315/12  315#17
CCR4   F6F8     315/ 7  315#23
CCR5   F705     314/56  314/58  315/10   315/15   315/19   315/21   315#34
CCR6   F70A     315/28  315#39
CCR7   F715     315/43  315#47
```

```
  CDEVIC 023A    13#54   237/29  237/40  237/44
  CDN    F3F3   295#39   338/19
  CDTMA1 0226    13#37    43/24  258/44  258/46
  CDTMA2 0228    13#38    43/40
  CDTMF3 022A    13#40    40/37  351/15  351/18  352/49  352/54
n CDTMF4 022C    13#42
n CDTMF5 022E    13#44
  CDTMV1 0218    13#30    43/61   44/ 7   44/10   44/12   44/15   45/10
                 45/12
n CDTMV2 021A    13#31
  CDTMV3 021C    13#32    40/32   40/33
n CDTMV4 021E    13#33
n CDTMV5 0220    13#34
  CEL    F1B4   282#15   289/33
  CEL1   F1C1   282/18   282#23
  CEP    E695   205/19   205/30  206/21  207/ 9  207/48  209/43  212#20
  CEP1   E69F   212/26   212#35
  CEP2   E6BA   212/31   212#52
  CGB    FD7A   195/35   353#23  354/32
  CGB1   FD8C   353#42   354/ 7
  CH     02FC    16#31    41/40  105/13  106/11  122/39  122/46  122/51
                273/18   290/30  291/13  291/21  345/35
  CH1    02F2    16#20   344/28  345/36
  CHACT  02F3    16#21    40/15   52/37  274/56
  CHACTL D401    22#43    40/16
  CHAR   02FA    16#29   281/26  283/31  283/39  319/61  320/26
  CHBAS  02F4    16#22    40/13   52/39  274/52  346/18  346/21
  CHBASE D409    22#49    40/14
  CHKERR 008F     6#41   246/26
  CHKSNT 003B    11#12   241/42  242/50  242/58  243/49
  CHKSUM 0031    10#60   241/41  241/49  242/55  243/24  243/26  244/32
                246/21   246/47  246/49  257/41
  CHLINK 03F8    17#46   218/41  218/43  219/17  219/18  226/11  226/13
  CHM    F440   298#46   305/38  315/34  339/13
  CHSALT 026B    14#27   274/54  346/19  346/20
  CIA    F76A   280/53   319#57
  CIA1   F77B   320/13   320#20
  CIA2   F78A   320/10   320/16  320#32
  CIA3   F78D   320#34   321/11
  CIN    FCDB   195/39   349#15
  CIO    E4DF   196/15   201#15
  CIO1   E4EC   201/26   201#33
  CIO2   E4F1   201/29   201#38
  CIO3   E4F3   201#40   201/45
  CIOCHR 002F    10#57   201/19  207/60  208/ 9  208/21  208/44  208/49
                209/54   210/ 5  210/22  211/59
  CIOINV E46E    24#42    60/51  196/39
  CIOV   E456    24#34    53/12   65/59  196/15
  CIR    C0A0    33/21    33/38   34#49
  CIR1   C0AD    34/59    35#11
  CIR2   C0B8    35/12    35#21
  CIR3   C0C9    35/27    35#37
  CIX    00F2    12#49   140/48  141/23  142/ 6  142/10  144/17  144/38
                144/42   144/51  144/53  158/47  159/21  159/46  159/59
                160/22   160/55  161/42
  CKEY   03E9    17#35    61/13   67/ 7   67/19
  CLF    F400   296#15   299/26  327/15  327/25  338/22
  CLM     F41B     297#38  304/47  339/19
  CLN     F7E2     303/38  322#31
  CLN1    F7F1     322#41  322/43
  CLOSE   000C       5#13  201/56
  CLS     007D       6# 8  281/48
  CLT     CB56      99#25  218/61  226/52  231/ 7  234/13
  CLT1    CB5B      99#31   99/33
n CMCMD   0007      10#16
  CNTL1   009F       6#16   41/21  345/12
  CNTLF1  0083       6#12   41/24  344/37
  CNTLF2  0084       6#13   41/27  346/10
n CNTLF3  0093       6#14
  CNTLF4  0094       6#15   41/30  346/13
  COC     EF9C     273/57  274#36
  COC1    EFA7     274/42  274#51
  COC10   F080     277/45  277#52
  COC11   F0A2     278/ 5  278# 9  278/12
  COC12   F0C6     277/48  277/54  278#33
  COC13   F0DB     278/29  278/36  278/41  278#51  278/54
  COC14   F0F1     279/ 6  279#13
  COC15   F107     279#24  279/27
  COC16   F10F     278/60  279/ 9  279#31
  COC17   F154     274/47  280# 9
  COC18   F164     280/ 5  280#19
  COC19   F175     280/21  280#32
  COC2    EFDC     275/13  275#21
  COC3    EFEF     275#36  275/38
  COC4    EFF7     275#44  275/47
  COC5    F019     276# 9  276/12
  COC6    F03E     276/27  276#44
  COC7    F04C     276/24  276/30  276/50  276#59
  COC8    F057     277#13  277/15
  COC9    F06C     277/21  277#27
  COLAC   0072      11#59  334/39  335/55  335/57  335/58  335/60  336/ 9
n COLBK   D01A      20#43
  COLCRS  0055      11#34  287/28  287/54  296/16  296/17  296/56  297/16
                   297/17  298/52  299/20  299/27  299/58  303/35  304/36
                   309/20  309/22  309/47  311/17  311/20  311/22  311/38
                   312/46  315/ 6  315/14  315/18  322/33  326/10  326/41
                   326/58  327/26  332/27  333/40  333/42  336/16  336/17
                   336/21  336/24  336/27  336/30
  COLDST  0244      14#13   46/42   50/19   53/51   81/24  233/60
  COLDSV  E477      24#45  196/48
  COLINC  02F9      16#27  333/49  334/23  336/13
  COLOR0  02C4      15#39  134/61  275/45
  COLOR1  02C5      15#40   38/35  135/ 7
  COLOR2  02C6      15#41  135/10  347/13
n COLOR3  02C7      15#42
  COLOR4  02C8      15#44  119/44  135/13  276/55
  COLPF0  D016      20#38  107/56
  COLPF1  D017      20#39   38/38  119/43  347/19
  COLPF2  D018      20#40  119/45
n COLPF3  D019      20#41
  COLPM0  D012      20#33   40/ 7
n COLPM1  D013      20#34
n COLPM2  D014      20#35
n COLPM3  D015      20#36
```

```
                    ATARI CAMAC Assembler Ver  1.0A  Page 378
                                                    D1:OS.ASM

COLRSH 004F      11#28   38/31   38/36   40/ 5  347/15  347/15  347/17
COMPLT 0043       7#32  240/31
CONS1X EE19     254/56  254/59  260#20
CONS2X EE1B     256/ 7  260#23
CONSOL D01F      19#32   39/54   57/18   61/10  105/ 8  108/19  108/36
                120/16  120/27  120/38  120/51  331/34  357/16  357/28
COP    FCE6     195/33  349#53
COUNTR 007E      12#10  324/32  324/35  324/46  324/49  324/51  325/16
                325/18  325/22  334/54  335/10  335/21  335/22  337/15
                337/17  337/18  337/20
CPB    FDB4     195/36  354#56
CPB1   FDC2     355/11  355#17
CRE    F6BC     287/20  288/61  314#15
CRETRI 000D       9#26  237/13  238/18
CRETRY 029C      15#15  237/14  237/61  238/19
CRITIC 0042      11#20   38/50   83/13   83/13   83/50   83/50   88/38
                 88/38   88/42   88/42   89/19   89/19  236/50  239/33
CRM    F40A     296/18  296#38  339/22
CRR    FFA9     368/29  369/30  369/31  369#50
CRR1   FFAB     369#56  369/61
CRR2   FFB9     370#10  370/25  370/29
CRR3   FFC4     370/14  370#18
CRR4   FFCA     370/19  370#23
CRSINH 02F0      16#18   34/25  275/29  284/21
CRSROR 008D       6#39  315/35
CRT    F411     297#15  299/57  338/25
CSC    F420     280/25  281/51  297#55  338/28
CSC1   F429     298#10  298/12  298/17
CSC2   F438     298#24  298/27
CSO    EA2A     238/42  239/ 6  239/11  239#30  249/25  258/27
CSOPIV E47D      24#47   67/15  196/54
CSP    FCE5     195/38  349#37
CST    FDCC     195/37  355#36
CTB    F49A     300#49  338/46
CTIA   D000      19#28   58/60
CTIM   0002       9#28  253/58  253/59
CUP    F3E6     295#15  299/34  305/39  338/16
CUP1   F3EE     295#22  295/46
CUP2   F3F0     295/17  295#24  295/43
DATAER 009C       8#45   71/35   71/40   71/60   72/45   72/50
DAUX1  030A      16#53   62/20   64/ 5  221/15  224/51  237/32  361/57
                367/14
DAUX2  030B      16#54   62/18  221/13  237/34  248/31  248/55  249/19
                359/20
DAW    5387     117/46  118#32  124/10  124/13  130/44
DAW1   5387     118#34  118/41
DAW2   5389     118#36  118/38
DBE    C63E      62/51   64/12   65#41
DBL    E6BB     208/27  208/32  210/10  213#18
DBL1   E6C1     213/20  213#24
DBP    E6C8     208/60  213#42
DBP1   E6CE     213/44  213#48
DBS    F57A     276/10  276/45  305/56  307#32
DBS1   F58D     307/49  307#55
DBS2   F59B     307/59  308#10
DBS3   F59F     306/40  307/41  307/57  308/ 6  308#15
DBSECT 0241      14#11   63/57
```

```
  DBUFHI 0305     16#48   62/25   69/18   70/43  247/42  358/59  365/30
  DBUFLO 0304     16#47   62/23   69/16   70/41  247/38  358/61  365/29
  DBUFSZ 0014      9#31  366/56
  DBYTHI 0309     16#52   69/32   70/20  247/44  358/55  365/46
  DBYTLO 0308     16#51   69/31   70/18  247/40  358/57  365/44
  DCB    0300     16#42  221/ 7  224/44
  DCH    F4D5    302#15  338/55
  DCH1   F4D8    302#23  302/42
  DCH2   F4FE    302/33  302/37  302#44
  DCOMND 0302     16#45   61/52   66/29   68/44   68/56   69/42   69/53
                 237/30  358/53  359/11  361/56  365/36  367/13
  DCSORG E000      4#57   52/38  191/ 6  191/ 6  191/ 6  191/ 6  191/ 6
                 191/ 6  274/51
  DCT    C255     38/43   40/21   40/36   43#60
  DCT1   C262     44/ 5   44#12
  DCT2   C26F     44/ 8   44/13   44/16   44#21
  DDD    F565    277/27  305#54
  DDEVIC 0300     16#43   68/42  224/55  236/54  237/26  250/11  251/42
                 352/35  359/ 6  365/32
  DELTAC 0077     12# 5  334/14  334/17  334/24  334/26  334/27  334/29
                 334/30  334/33  334/50  334/53  334/58  335/56  335/59
  DELTAR 0076     11#61  333/53  333/60  334/ 7  334/59  335/ 5  335/31
  DERRF  03EC     17#38   50/26  280/ 9  280/11  280/13
  DERROR 0090      6#42  240/39
  DFLAGS 0240     14#10   63/22
  DFS    538E    113/28  113/38  116#39
  DIGRT  00F1     12#47  139/20  139/47  139/53  139/59  140/39  140/43
                 142/12  142/18
  DINDEX 0057     11#35  274/40  275/56  277/43  278/35  278/58  284/18
                 309/12  311/23  311/41  312/48  314/25  314/53  320/ 8
                 332/22
  DINITV E450     24#32   60/54  196/ 9
  DIO    C6B3     68#37  196/12
  DIO1   C6C4     68/40   68#50
  DIO10  C736     69/55   70#24
  DIO2   C6D4     68/58   69# 6
  DIO3   C6D6     68/61   69#10
  DIO4   C6EA     69/11   69#25
  DIO5   C6F0     69/21   69#30
  DIO6   C6FF     69/34   69#42
  DIO7   C710     69/44   69#53
  DIO8   C71C     69#60   70/ 8
  DIO9   C71E     70# 6   70/14
n DIRECT 0002      5#35
n DISK   0044      5#48
  DISKID 0031      6#50   68/41
  DISL1  513A    104/31  104/32  109#19  109/39
  DISL2  51D1    110#49  113/10  113/11
  DISL3  51ED    111/ 6  111#14  111/58  112/26
  DISL4  5215    111#45  122/ 6  122/ 7
  DISL5  5231    112# 8  127/16  127/17
  DISPLY 0053      5#52   55/20
  DLISTH D403     22#45   39/29
  DLISTL D402     22#44   39/31
  DLN    F520    303#54  338/40
  DLN0   F527    304#17  304/43
  DLN1   F527    304#15  328/58
  DLN2    F529    304#19  304/30
  DLP     DCC1    143/59  145/52  170#42
  DLW     5385    115/41  116/19  118#15  122/34  129/48  130/14
  DMACTL  D400     22#42   39/33
  DMASAV  02DD     16# 5  345/52  346/41
  DMASK   02A0     15#19  281/18  283/49  283/51  309/51
  DMW     5381    113/29  113/48  115/34  117#44  128/13
  DNACK   0088      6#37  240/45
  DOSINI  000C     10#23   63/34   63/36   65/22   67/22   67/24
  DOSVEC  000A     10#22   49/51   49/53   54/11
  DOUBLE  0044      7#20  366/53
  DQQ     F918    302/47  328#15
  DQQ1    F920    328/18  328#22
n DRAWLN  0011      5#25
  DRETRI  0001      9#27  237/10
  DRETRY  028D     15#30  237/11  239/10
  DRKMSK  004E     11#27   38/30   38/37   40/ 6  347/17
  DRS     53C3    114/26  115/36  115/46  118#58
  DSCTLN  02D5     15#56   68/19   68/21   69/25   69/26
  DSCTSZ  0080      4#49   68/18   68/20
  DSD     F578    277/19  307#15
n DSKFMS  0018     10#35
  DSKINV  E453     24#33   61/53   66/32  196/12
  DSKTIM  0246     14#15   68/17   68/43   69/49
n DSKUTL  001A     10#36
  DSPFLG  02FE     16#33  289/38
  DSR     EC84    239/31  242/10  252#18  258/16
  DSR1    EC92    252#33  252/36
  DSS     5399    113/47  113/57  116#59
  DSTAT   004C     11#25  274/59  279/61  284/42  286/ 8  287/32  288/11
                  288/26  292/58  306/39  307/40  308/11  315/24  315/39
  DSTATS  0303     16#46   69/30   70/24   83/51   83/52  238/13  238/34
                  239/35  248/19  359/18  365/42
  DT1     00FA      7#54  354/21  356/19
  DTA     00FC      7#53  355/17
  DTIMLO  0306     16#49   68/50  252/54  359/10  365/48
  DUNIT   0301     16#44   61/50   66/31   83/14   83/47  237/27  359/ 8
                  365/34
n DUNUSE  0307     16#50
  DVN     5685    127/57  128/20  128/30  128/40  128/44  128/48  128/58
                  129/ 5  129/ 9  129/19  129/23  129/27  129/37  129/41
                  131#15
  DVN1    5694    131#25  131/30  131/35
  DVNM    C28A     39/ 5   45#31   56/14  196/27
  DVSTAT  02EA     16#15   69/15   69/17  204/42  204/43  219/37  219/40
                  220/ 7  220/ 9  220/11  221/28  222/23  222/27  222/30
                  222/34  222/36  228/44  228/45  229/31  229/35  230/24
                  230/25  268/60  361/30  366/19
  DWQ     F923    299/24  328#38
  DWQ1    F932    328#52  328/56
  EBL     C629     64/27   64#49
  ECL     F22E    194/14  194/34  286#43
  EDITRV  E400     24#20   55/18   60/43  194/ 6
  EEXP    00ED     12#35  139/16  140/ 5  140/56  140/58  141/ 9  141/11
                  141/14  141/15  141/18  141/19  141/31  141/35  141/41
                  141/42  142/11  142/24  142/29  144/49  144/54  144/59
                  144/60  145/11  155/58  171/47
  EGB     F24A     194/15   287#15
  EGB1    F25C     287#31   287/45
  EGB2    F277     287/41   287#45
  EGB3    F27A     287/35   287#49
  EGB4    F288     287/22   287#58
  EGB5    F28C     288# 6   288/12
  EGB6    F29D     287/59   288/ 7   288#22
  EHC     E6EA     205/24   206/26   207/10   207/59   208/ 8   208/43   210/ 7
                   210/41   214#60
  ELL     F7C2     303/33   321#51
  ELL1    F7C5     321#55   322/10
  ELL2    F7D9     322/ 7   322#12
  EMS     5003      52/40   103#41
  END     C795      71/46    72#43    80/41
  END1    C7C0      73#12    73/21
  END2    C7C6      72/56    73#15
  END3    C7C8      72/46    72/51    73#17
  END4    C7C9      72/58    73#19
  ENDPT   0074      11#60   313/53   313/56   334/55   335/11   335/13   335/38
                   335/44   335/61   336/10
n ENTVEC  002C      10#55
  EOFERR  0088       6#34   292/54   354/40
  EOL     009B       6# 9    55/34    55/42   208/22   208/50   209/ 5   210/23
                   210/40   282/17   287/34   288/23   293/ 6  .310/33   312/26
                   313/14   363/30   365/ 6
  EOP     EF94     194/13   274#15   280/10   286/60   314/30
  EOT     00FE       7#55   354/16   356/34
  EPB     F280     194/16   288#58
  EPC     C9DC      85/17    85/34    85/51    86/17    86/34    86/51    88#31
  EPC1    C9ED      88#48    89/31
  EPC2    CA03      88/49    89#10
  EPC3    CA05      89/ 6    89#14
  EPC4    CA1B      88/56    89#27
  ERRFLG  023F      14# 6    14/ 8   237/53   238/28   238/37   238/51   240/ 7
                   240/57
  ERROR   0045       7#33   240/34
  ESC     F3E0     294#50   338/13
  ESCFLG  02A2      15#21   289/32   289/39   289/44   294/52
  ESIGN   00EF      12#41   141/26   141/53
  ESP     F22D     194/18   194/56   194/58   286#27
  ESR     EC40     244/38   250#45
  ESS     EC17     196/33   241/39   248/28   249#58
  ESS1    EC31     250/13   250#25
  EST     5009     103/26   103#58
  EXP     DDC0      23#45   181/17
  EXP1    DE03     182/35   182#60
  EXP10   DDCC      23#46   182/ 6
  EXP2    DE20     183/21   183#25
  EXP3    DE26     183#29   183/31
  EXP4    DE39     183/13   183#48
  EXP5    DE4A     183/49   183#59
  EXP6    DE4B     181/42   182/40   182/49   183/36   183/38   184# 6
  F1R     DD08     172/22   173#29
  F2R     DD11     172/44   174#20
  FADD    DA66      23#32   152/ 6   177/ 8   185/11   188/24   188/45
  FADD1   DA66     152#31   152/52
  FADD10  DAC5     153/55   154# 7
  FADD11 DACE     154#14  154/18
  FADD2  DA77     152#44  152/50
  FADD3  DA85     152/38  152#56
  FADD4  DA8E     152/56  153#10
  FADD5  DA98     153#20  153/24
  FADD6  DAA4     152/59  153#31
  FADD7  DAA7     153/27  153#35
  FADD8  DAB3     153/13  153#46
  FADD9  DAB6     153#49  153/53
  FALSE  0000       4# 8    4/ 9    4/12    4/13    4/14
  FARR   DD13     173/31  174#42
  FARR1  DD19     174#53  174/59
  FASC   D8E6      23#26  143/ 6
  FASC1  D914     143/55  143#59
  FASC10 D988     143/57  145#31
  FASC11 D99C     143/60  145/33  145#47
  FASC12 D9A9     145/48  145#59
  FASC2  D91A     143/36  144# 7
  FASC3  D920     143/40  143/43  144#13
  FASC4  D94C     144/40  144#44
  FASC5  D94F     144/30  144#49
  FASC6  D969     144/55  145# 7
  FASC7  D96B     145/ 5  145# 9
  FASC8  D972     145#13  145/18
  FASC9  D97A     145/15  145#20
  FCHFLG 00F0      12#44  139/27  140/23
  FDH    E716      91/16  216#26
  FDH1   E71A     216#36  216/42
  FDH2   E724     216/30  216#46
  FDH3   E728     216/37  216#52
  FDIV   DB28      23#34  156/23  183/55  185/25  187/58  188/57
  FDIV1  DB43     157#12  158/14
  FDIV2  DB45     157#14  157/18
  FDIV3  DB4E     157/ 8  157#22  157/36
  FDIV4  DB52     157#26  157/30
  FDIV5  DB65     157/33  157#40
  FDIV6  DB70     157#51  158/ 8
  FDIV7  DB74     157#55  157/59
  FDIV8  DB87     158/ 5  158#12
  FDL    FCC4     275/15  275/17  347#11
  FEOF   003F      11#16  350/36  353/27  354/36
  FHALF  DF6C     188/21  188/22  189#21
  FILDAT 02FD      16#32  336/60
  FILFLG 02B7      15#25  333/34  336/42
n FILLIN 0012       5#26
  FINE   026E      14#30  275/12  277/61  278/40  286/47  286/55  306/16
                  323/11
  FIX    mac       28#36  100/ 6  103/ 6  138/11  138/37  143/ 6  146/ 6
                  147/11  149/14  149/37  151/29  152/ 6  154/29  156/23
                  176/11  177/25  178/ 6  178/39  179/ 6  179/39  180/11
                  180/45  181/17  182/ 6  185/32  186/ 6  191/ 6  194/ 6
                  194/26  194/46  195/ 6  195/26  196/ 9  196/12  196/15
                  196/18  196/21  196/24  196/27  196/30  196/33  196/36
                  196/39  196/42  196/45  196/48  196/51  196/54  196/57
                  196/60  197/ 6  197/ 9  197/12  198/ 6  199/ 6  232/ 6
                  235/ 6  272/ 6  285/ 6  348/ 6  371/ 6  372/ 6
  FKDEF  0060      11#41  273/32  273/34  293/42
```

```
   FLD01   DD8F     178#28   178/31
   FLD0P   DD8D      23#37   178/ 6
   FLD0R   DD89      23#36   176/45   177/25   182/55   183/54   185/17
   FLD11   DD9E     179#28   179/31
   FLD1P   DD9C      23#39   179/ 6
   FLD1R   DD98      23#38   177/ 7   177/16   178/39   181/37   185/10   185/20
                    185/24   188/19   188/23   188/56
   FLPTR   00FC      12#56   177/45   177/46   178/28   178/59   178/60   179/28
                    179/60   179/61   180/35
   FM01    DDB8     181# 6   181/ 9
   FMOVE   DDB6      23#42   176/42   180/45   182/52   183/ 7   183/51   188/11
                    188/25
   FMPREC  0005       7#44   139/42   139/43   152/58   153/17   153/46   154/12
                    157/17   163/34   163/48   165/40   166/16   167/14   171/56
                    183/26   187/51
 n FMSZPG  0043      11#22
   FMUL    DADB      23#33   154/29   176/49   181/41   183/ 8   183/44   188/12
                    188/20
   FMUL1   DAF1     155#18   155/54
   FMUL2   DAF7     155#24   155/28
   FMUL3   DB01     155/25   155#32
   FMUL4   DB09     155#41   155/45
   FMUL5   DB13     155/42   155#49
   FMUL6   DB1A     155#58   158/22
   FMUL7   DB21     154/60   156# 7
   FMUL8   DB24     154/55   156#11   156/54
   FMUL9   DB26     155/10   156#16   156/49   156/61
   FNCNOT  0092       6#44    88/11   214/61   271/ 6
   FNZ     DCA4     143/50   144/15   169#24
   FNZ1    DCA6     169#32   169/42
   FNZ2    DCB4     169/34   169#46
   FNZ3    DCB8     169/37   169#51
   FOMAT   0021       7# 8    68/45    69/54
   FONE    DE8F     183/52   183/53   184#42
   FPB     FEED     363#56   365/ 8
   FPB1    FEED     363#60   364/ 6
   FPI     D9D2      23#28   147/11   182/45
   FPI1    D9EA     147#59   148/34
   FPI2    DA24     147/49   148#38
   FPI3    DA38     148/40   148#54
   FPI4    DA42     147/42   147/45   147/60   148/10   148/13   148/22   148/30
                    149# 6
   FPREC   0006       7#42     7/44   152/42   155/18   155/32   157/22   157/51
                    172/21   172/43   173/ 8   173/30   173/52   174/21   174/46
                    175/28   175/55   176/54   178/26   179/26   180/32   180/61
                    182/39   184/26   189/40
   FPSCR   05E6      18#35   182/42   182/43   182/53   182/54   185/12   185/13
                    185/22   185/23   188/ 8   188/ 9   188/17   188/18
 n FPSCR1  05EC      18#36
   FPTR2   00FE      12#57   176/36   176/37   176/43   176/44   176/53   176/55
                    176/58   176/60   177/ 5   177/ 6   184/60   184/61   185/ 8
                    185/ 9   185/18   185/19
   FR0     00D4      12#22   142/32   142/44   142/46   143/35   144/21   145/47
                    146/28   146/30   146/44   146/45   146/46   146/58   147/41
                    148/55   148/57   149/30   152/34   152/44   152/48   153/11
                    154/ 7   154/ 9   154/54   155/59   156/53   157/14   157/15
                    163/35   163/53   164/ 7   164/13   164/52   166/18   166/19
```

```
                   166/24  171/21  171/27  171/49  172/21  172/43  175/57
                   178/29  180/34  181/ 6  182/26  182/29  182/46  182/48
                   186/55  187/40  188/27  188/29  188/35  188/42  188/43
                   190/28
  FROM    00D5      12#23  139/37  139/42  139/43  153/20  153/22  153/38
                   153/49  153/51  154/15  154/16  162/21  163/38  163/45
                   163/46  163/59  166/58  167/10
  FR1     00E0      12#28  151/56  151/58  152/31  152/45  152/46  153/12
                   154/59  155/ 9  156/48  156/58  157/56  165/17  171/22
                   171/25  171/26  171/50  173/30  175/30  179/29  181/ 7
                   183/19  183/33  183/40  190/29
  FR1M    00E1      12#29  153/21  153/50  183/25  183/29  187/50  187/54
  FR2     00E6      12#31  157/27  161/61  171/55  174/21  175/31
  FRA10   DD01     155/27  172#20
  FRA1E   DD09     158/12  173# 7
  FRA20   DD05     155/44  172#42
  FRA2E   DD0F     157/40  173#51
  FRE     00DA      12#26  155/18  155/32  157/26  157/28  157/55  157/57
                   163/17  173/ 8  173/52  175/58
  FREQ    0040      11#17  356/60  357/42
  FRMADR  0068      11#51  320/55  320/58  321/17  321/25  321/28  321/29
                   321/32
  FRMERR  008C       6#38  245/57
  FRSIGN  00EE      12#37  171/24  171/46
  FRX     00EC      12#33  140/49  142/ 5  162/49  170/21  171/53
  FST01   DDAD     180#34  180/37
  FSTOP   DDA8      23#41  180/11
  FSTOR   DDA7      23#40  176/41  179/39  182/44  185/ 7  185/14  188/10
  FSUB    DA60      23#31  151/29  182/56  185/21
  FTX     C851      74/49   76#30
  FTYPE   003E      11#15   67/13  349/58  359/19
  GBY     C7CF      71/34   71/39   71/59   72/44   72/49   73#36
  GBYTEA  02CF      15#52   73/37  222/39  222/41
  GCL     C996      85#32  198/14
  GDC     F18F     280/52  281#15  284/16  301/17  301/32  302/39  327/17
                   336/57
  GDC1    F197     281#20  281/24
  GDC2    F19E     281/21  281#26
n GETCHR  0007       5#10
  GETDAT  0040       9#23   68/54  359/12
n GETREC  0005       5# 9
  GGB     C998      85#49  198/15
  GIN     C90C      82#15  198/19
  GIN1    C911      82#24   82/38
  GIN2    C928      82/29   82/33   82#37
  GINTLK  03FA      17#45   30/23   39/19   46/27
  GNC     DB94     139/31  140/50  141/ 5  141/57  158#45  160/60  161/26
                   161/34
  GND     DC89     148/24  148/38  170#19
  GNL     E833     223/57  224#37
  GNL1    E835     224#43  224/46
  GNLA    E851     224/43  225# 7  225/18
  GNLAL   000C     224/41  225#18
  GNS     C659      62/46   64/ 7   64/23   66#15
  GNS1    C661      66/20   66#28
  GOP     C991      85#15  198/13
  GPB     C9A0      86#15  198/16
  GPDVV  E48F      25# 8   61/ 6  198/ 6
  GPRIOR 026F      14#31   39/34  276/16  276/59
n GRACTL D01D      20#47
n GRAFM  D011      20#31
n GRAFP0 D00D      20#26
n GRAFP1 D00E      20#27
n GRAFP2 D00F      20#28
n GRAFP3 D010      20#29
  GSP    C9AA      86#49  198/18
  GST    C9A5      86#32  198/17
  GTO    EC9A     238/26  249/12  252#53
  HATABS 031A      17#12   52/ 9  212/35  212/37  216/36  266/54  267/12
                  267/41  267/43  267/45
n HDR    00FB       7#52
  HELP   0011       6#11   41/34  345/25
  HELPFG 02DC      15#61  106/36  119/54  119/60  345/30
  HIBYTE 0288      14#61   71/38   71/44   74/60   75/51   78/43   80/ 9
                   80/19
  HIG    C8B5      79#43   80/38   80/39
  HIG1   C8D1      80#11   80/22
  HIG2   C8D2      79/54   80#15
n HITCLR D01E      20#48
  HITONE 0005       9#13  250/20
  HIUSED 02CB      15#48   75/34   75/40   75/45   75/48
  HNDLOD 02E9      16#14  201/61  204/31  211/48  270/20
  HOLD1  0051      11#30  275/58  278/47  278/51  279/20  279/24  279/35
                  303/56  325/48  325/52  325/60  327/54
n HOLD2  029F      15#18
  HOLD3  029D      15#16  313/ 5  313/26  313/31
  HOLD4  02BC      15#29  336/49  337/ 8
  HOLDCH 007C      12# 8  291/19  293/36  293/61  294/18
n HPOSM0 D004      20#14
n HPOSM1 D005      20#15
n HPOSM2 D006      20#16
n HPOSM3 D007      20#17
n HPOSP0 D000      20# 9
n HPOSP1 D001      20#10
n HPOSP2 D002      20#11
n HPOSP3 D003      20#12
n HSCROL D404      22#46
  IBP    E6D1     208/14  209/ 7  210/ 9  214#15
  IBP1   E6D7     214/17  214#21
  IBS    C63B      61/45   64/32   65#21
  ICAX1  034A      17#29   53/11
  ICAX1Z 002A      10#51  207/37  209/32  274/16  274/18  277/52  280/19
                  290/58  350/ 5
n ICAX2  034B      17#30
  ICAX2Z 002B      10#52  273/55  349/57  366/43
  ICBAH  0345      17#24   53/ 9   65/51  211/42
  ICBAHZ 0025      10#46  211/43
  ICBAL  0344      17#23   53/ 7   65/49  211/40
  ICBALZ 0024      10#45  208/13  209/ 6  209/61  211/41  213/43  213/46
                  213/48  214/16  214/19  215/52  216/10  268/46  269/ 8
  ICBLH  0349      17#28  214/42
  ICBLHZ 0029      10#50  214/44
  ICBLL  0348      17#27   65/55  214/39
  ICBLLZ 0028      10#49  207/53  207/54  209/48  209/49  209/55  210/28
```

```
                 210/29  213/19  213/22  213/24  213/25  213/26  214/40
                 214/41  214/43
 ICCOM   0342     17#21   53/ 5   65/53
 ICCOMT  0017     10#34   91/29  203/33  205/29  212/39
 ICCOMZ  0022     10#43  201/55  203/20  207/36  208/15  208/37  209/31
                 210/16  210/34  333/22
 ICDNO   0341     17#20  363/ 8
 ICDNOZ  0021     10#42  215/61  216/ 5  268/47  363/ 9  365/33
 ICH     F49F    301#15  338/58
 ICH1    F4AC    301#22  301/35
 ICH2    F4C6    301/28  301#39
 ICH3    F4C9    301#41  301/45
 ICH4    F4D2    301/42  301#47
 ICHID   0340     17#19   91/23  200/22  207/15
 ICHIDZ  0020     10#41   91/24  201/49  204/19  206/31  206/55  207/16
                 212/24  216/53  268/55
 ICIDNO  002E     10#56   90/57   90/57   91/14   91/14   91/22   91/22
                 201/20  207/14  211/36  211/60  214/37  215/30  269/ 5
                 269/ 5  270/39  270/39
 ICIO    E4C1    196/39  200#15
 ICIO1   E4C3    200#21  200/32
 ICPTH   0347     17#26  200/26
 ICPTHZ  0027     10#48  205/34  206/33  268/59  270/58
 ICPTL   0346     17#25  200/24
 ICPTLZ  0026     10#47  205/32  206/35  268/57  270/60
 ICR     FD05    272/15  350#58
 ICR1    FD18    351#18  351/19
 ICS     C429     52/56   54#27
 ICSORG  CC00      4#56  100/ 6  100/ 6  100/ 6  100/ 6  100/ 6  100/ 6
                 274/53  346/23
 ICSPR   034C     17#31   90/58   91/15  269/ 6  269/ 9
 ICSPRZ  002C     10#53  201/44  205/31  205/33  211/54  212/36  212/38
                 212/42  212/45  212/46  212/47  215/25  215/27
n ICSTA  0343     17#22
 ICSTAZ  0023     10#44  206/20  208/56  211/16  211/61  215/ 6
 ICX     DB9D    141/24  158/48  159#19
 IDCPU   0002      4#26  371/14
 IDDAY   0010      4#23   29/14  371/13
 IDH     E6F4    215/ 5  215#23
 IDIO    C6A3     68#15  196/ 9
 IDMON   0005      4#24   29/14  371/13
 IDPN1   0042      4#27   29/16  371/15
 IDPN2   0042      4#28   29/16  371/15
 IDPN3   0000      4#29   29/16  371/15
 IDPN4   0000      4#30   29/16  371/15
 IDPN5   0001      4#31   29/16  371/15
 IDREV   0002      4#22   29/17  371/16
 IDYEAR  0083      4#25   29/14  371/13
 IFP     D9AA     23#27  146/ 6  182/51  188/37
 IFP1    D9B8    146#39  146/51
 IFP2    D9BE    146#44  146/48
 IGN     F2F8    290#28  291/37  291/54  292/ 9  292/23  292/34  292/45
                 293/23
 IHW     C4DA     56/55   58#52
 IHW1    C4E0     58#60   59/12
 IHW2    C4F0     59/ 7   59#11
 IIH     C00C     30#17  196/36
```

```
  IIN     E4DC    200/23  200/25  200#52  206/32  206/34
  IIO     E55C     91/30  205#15
  IIR     C030     32#18   56/ 7
  IIR1    C045     32/28   32#40
  IIR2    C052     32/53   32#61
  IIR3    C054     33# 6   33/17
  IIR4    C05F     33/ 8   33#13
  IIR5    C064     33/11   33#16
  IIR6    C06A     33/14   33#25
  IIR7    C07D     33/33   33#42
  ILN     F50C    303#15  338/43
  ILN1    F50D    303#32  312/ 8
  ILP     DA51    143/29  150#44
n IMASK   028B     15# 7
  INBUFF  00F3     12#50  145/38  145/40  145/41  145/43  145/55  150/46
                  150/48  158/50  159/51  160/23  170/44  170/46  170/47
                  170/49
  IND     E510     91/34  202#24
  IND1    E512    202#27  202/44
  INIML   0700      4#54   60/36   60/38
n INSCLR  0020      5#39
  INSDAT  007D     12# 9  301/18  301/30  301/33  310/50  311/48  311/61
                  312/27  313/ 9
  INTABS  0200     13#16   33/44   33/46   37/14   37/15   37/16   37/17
                   37/18   37/19   37/20   37/21   51/57
  INTEMP  022D     13#43   44/53   45/ 9
  INTINV  E46B     24#41   60/53  196/36
  INTZBS  0010     10#28   50/52
  INVFLG  02B6     15#24  292/ 6  292/ 8  294/27
  IOCB    0340     17#18  201/40  211/51  270/44
  IOCBAS  0020     10#40  201/41  201/44  211/50  211/54
  IOCBSZ  0010      4#44    4/46    4/47  200/29
  IOCFRE  00FF      9# 8  200/21  204/20  206/30  206/56
  IPH     C9CA     87#60   88/55
  IRA     F2AD    288#42  290/ 8
  IRI1    EB3A    245/53  246# 5
  IRI2    EB42    246/ 6  246#15
  IRI3    EB51    246/22  246#31  247/15
  IRI4    EB55    246#36  246/61  247/21
  IRI5    EB59    246/16  246#43
  IRI6    EB6D    246/51  246#57
  IRI7    EB81    247/ 9  247#19
  IRIR    EB2C     55/58  245#42  253/18
  IRQ     C02C     31#39  372/15
  IRQEN   D20E     22# 6   32/33   32/35   33/26   33/28  243/ 9  243/61
                  251/20  252/26  275/ 6
  IRQST   D20E     21#41   32/26   33/13
  ISIO    E95C    196/30  236#15
  IST     5086    103/42  103/59  105#58
  ISW     C535     52/15   60#15
  ITO     EB27    244/54  245#21  257/ 8
  IVN1    C0EE     38/11   38/15   38#21
  IVN10   C19D     40/34   40#39
  IVN11   C1B0     40/46   40/51   40#57
  IVN12   C1EE     41/ 5   41#45
  IVN13   C1F3     40/58   41/ 8   41/13   41/22   41/25   41/28   41/31
                   41/35   41/41   41#50
  IVN14   C21F      42#26    42/32
  IVN15   C232      42#43    42/61
  IVN2    C0FC      38/24    38#30
  IVN3    C114      38/44    38#50
  IVN4    C120      38/51    39# 5
  IVN5    C123      38/58    39# 9
  IVN6    C162      39/40    39#53
  IVN7    C167      39#60    40/ 9
  IVN8    C18B      40/22    40#28
  IVN9    C18D      40#30    40/40
  IVNM    C0E2      38# 6    56/13    196/24
n JMPERS  030E      16#57
  JVECK   028C      15# 8    33/45     33/47    33/50    35/22    35/31    35/37
  KBCODE  D209      21#38    41/17     41/39   344/27   345/ 7
  KBD     004B       5#50    55/23
  KBK     5527     123/27   125#42
  KEYBDV  E420      24#22    55/24     60/45   194/46   358/30   358/32
  KEYDEF  0079      12# 6   273/27    273/29   291/41
  KEYDEL  02F1      16#19    40/50     40/53   344/31   345/41
  KEYDIS  026D      14#29    33/37     41/12   344/36   344/44
  KEYREP  02DA      15#59    41/15     51/43
  KGB     F2FD     194/55   287/31    290#49
  KGB1    F2FD     290#53   291/15
  KGB10   F372     291/ 9   292#58
  KGB11   F376     290/60   293# 6
  KGB12   F378     292/50   293#11
  KGB13   F38C     293/19   293#23    293/31   293/51
  KGB14   F38F     293/12   293#27
  KGB15   F39F     293/37   293#41
  KGB16   F3A5     293/28   293#47
  KGB17   F3B4     291/49   293/48    293#61
  KGB18   F3CF     294/ 6   294/10    294/13   294/16   294#23
  KGB19   F3DA     293/ 7   293/57    294#31
  KGB2    F323     291/26   291#32    294/19
  KGB20   F3DD     294/24   294#35
  KGB3    F32A     291#45   293/43
  KGB4    F333     291/47   291#53
  KGB5    F345     291/59   292#13
  KGB6    F355     292/14   292#27
  KGB7    F359     292/19   292#32
  KGB8    F360     292/28   292#38
  KGB9    F36C     292/39   292#49
  KIR     FC19      55/57   344#18
  KIR1    FC2D     344/29   344#36
  KIR10   FC90     345/26   346#10
  KIR11   FCAE     346/24   346#31
  KIR12   FCB5     346/11   346#38
  KIR2    FC41     344/45   344#51
  KIR3    FC44     344/49   344#54
  KIR4    FC47     344/38   344#59
  KIR5    FC5C     345/13   345#24
  KIR6    FC67     345#35   346/14
  KIR7    FC6D     344/55   345/20    345/31   345#40   346/29   346/34
  KIR8    FC76     344/32   345#47
  KIR9    FC87     344/60   345/50    345#57   346/39   346/44
  KRK     5539     123/30   126#22
  KRPDEL  02D9      15#58    51/45    345/47
```

```
   KS8     551F     123/21   125#24
   KTK     552F     123/24   125#60
 n LBPR1   057E      18#30
   LBPR2   057F      18#31   143/31
   LBUFF   0580      18#32   143/52   143/53   144/ 8   144/28   144/34   144/35
                    144/36   144/37   145/31   150/45   150/47   168/44   169/32
                    169/47
   LCOUNT  0233      13#49    71/33    71/56    72/ 7    74/45    76/33    79/51
   LEDGE   0002       4#51    51/23
   LGCOEF  DF72     188/14   188/15   189#29   189/40
   LHO     E515     202/ 5   202#42
   LMARGN  0052      11#31    51/24   280/27   296/20   297/39   299/17   299/21
                    299/59   303/34   322/32   322/37   327/11   327/22   327/56
                    328/17   328/43   328/49   332/25
   LNBUG   0000       4#13    19/ 8    31/18    31/47    48/57
 n LNFLG   0000      10# 8
   LOADAD  02D1      15#53    73/ 7    73/10    75/12    75/15    77/34    77/42
                     78/58    80/16    80/18   222/35   222/37
   LOG     DECD      23#43   185/32
   LOG10   DED1      23#44   186/ 6
   LOG10E  DE89     181/35   181/36   184#34   188/54   188/55
   LOGC    DEE0     187#33   190/31
   LOGC2   DEEF     187/42   187#47
   LOGC3   DEF3     187/45   187#50
   LOGC4   DEF9     187#54   187/56
   LOGC5   DF46     188/30   188#37
   LOGC6   DF53     188/39   188#45
   LOGC7   DF64     188/50   188#61
   LOGCOL  0063      11#43   287/39   298/48   299/16   300/11   300/33   300/50
                    301/23   301/27   302/28   302/32   311/16   311/44   325/46
                    325/56   325/59   326/ 9   326/11   326/36   327/10   327/21
                    327/36   328/16   328/42
   LOGMAP  02B2      15#23   298/24   304/25   304/32   304/34   311/57   313/27
                    317/19   317/20   317/21   319/16   321/60   322/14
   LOGQ    DFF6     187/ 7   190#27
   LOGS    DED3     185/50   186#47
   LOGS1   DEDE     186/56   186/58   187#11
   LOO     C892      78#41    80/32    80/33    80/34    80/35
   LOO1    C89D      78/45    78#49
   LOTONE  0007       9#14   250/18
   LPENH   0234      13#50    39/27
   LPENV   0235      13#51    39/25
   LPH     E7DE      90/59   220/12   222#11
   LPH1    E7F8     222/25   222/28   222#34
   LTEMP   0036      11# 8    76/37    76/40    76/43    77/27    77/30    77/32
                     77/35    77/36    77/39    77/41    77/43    78/51    78/54
                     78/56    78/59    79/60    80/ 6    80/ 8    80/21
   M0E     DD34     171/58   175#54
   M0E1    DD36     175#57   175/60
 n M0PF    D000      19#36
 n M0PL    D008      19#46
   M12     DD26     171/51   175#27
   M121    DD2A     175#30   175/33
 n M1PF    D001      19#37
 n M1PL    D009      19#47
 n M2PF    D002      19#38
 n M2PL    D00A      19#48
```

```
n M3PF    D003       19#39
n M3PL    D006       19#49
  MAXDEV  0021        4#43   212/25   216/34   266/60   267/18
  MAXIOC  0080        4#47   200/31   201/28   270/17
  MEMERR  009D        8#46    76/10
  MEMLO   02E7       16#13    60/37    60/39   219/36   219/39   220/ 8   220/10
                    230/51   230/53   230/55   230/57
  MEMTOP  02E5       16#12    60/30    60/32    75/57    76/ 5   219/43   219/45
                    279/49   279/51
n MINTLK  03F9       17#44
  MLN     F78E      303/37   320#49
  MLN1    F79F      321# 9   321/36
  MLN2    F7A7      321#17   321/20
  MLTTMP  0066       11#46   308/55   309/27   309/48
  MOTRGO  0034        9#19   248/38   249/ 5   351/ 5   352/37
  MOTRST  003C        9#20   236/17   249/22   258/17   356/ 9
  MXDMOD  0010        5#38   277/53   278/25
  NOE     DC04      155/60   163#33
  NOE1    DC0A      163#38   163/55
  NOE2    DC10      163#45   163/49
  NOE3    DC28      163/39   163/60   164#13
  NOE4    DC31      164/16   164#25
  NOE5    DC38      163/36   164/26   164#34
n NACK    004E        7#34
  NBUFSZ  0028        9#30   366/50
  NCOMHI  003C        9#17   236/20   244/39   258/19
  NCOMLO  0034        9#16   237/50
  NEWADR  028E       15# 9    74/52    74/58    75/13    75/16    75/19    75/23
                     76/36    76/39    77/26    77/29    78/50    78/53    79/59
                     80/ 5
  NEWCOL  02F6       16#25   333/41   333/43   334/12   334/15
  NEWROW  02F5       16#24   333/39   333/51
  NGFLAG  0001       10# 9    49/10    49/29    49/36    50/10    52/28
  NLCOEF  000A      188/13   189#40
  NMI     C018       30#46    30/48   372/13
  NMI1    C020       30/53    30#59
  NMIEN   D40E       22#51    30/20   104/28   104/61   106/ 7   275/11   275/21
                    286/53
  NMIRES  D40F       22#52    31/14
  NMIST   D40F       22#38    30/52
  NOCKSM  003C       11#13   240/20   247/ 8   247/14
  NOCLIK  02DB       15#60   291/25   293/16   293/18
n NODAT   0000        9#22
  NONDEV  0082        6#28    89/10   202/25   216/46   268/51   270/23
  NORM    DC00      142/34   147/ 5   153/31   153/42   153/60   154/23   163#15
  NORMAL  004E        7#21   366/47   367/19
  NOTOPN  0085        6#31   200/53   212/30
  NPCOEF  000A      182/60   184#26
  NSIGN   00EE       12#38   140/34   142/39
  NVALID  0084        6#30   203/19   333/29
  OCI     FCF7      196/54   350/ 8   350#30
  OCI1    EAFB      243/50   244# 8
  OCIR    EAEC       55/60   243#45   253/20
  OCO     FD34      350/11   352#15
  OCO1    FD6A      352#51   352/55
  OCO2    FD77      351/26   353# 7
  OLDADR  005E       11#40   309/39   309/43   316/17   316/21
```

```
   OLDCHR  005D     11#39   284/17   316/20
   OLDCOL  005B     11#38   334/13   334/16
 n OLDPAR  02CF     15#51
   OLDROW  005A     11#37   282/52   333/52   334/43
   OPEN    0003      5# 8    52/60    91/28   203/21
   OPNIN   0004      5#36    53/10
   OPNOT   0008      5#37    53/10
   OPNTMP  0066     11#47   276/18   276/39   277/37   278/26   278/33
   ORI1    EAB5    242/36   242#42
   ORI2    EACE    242/51   243# 6
   ORI3    EAD7    242/59   243#13   243/27
   ORI4    EADB    242/46   243#20
   ORIR    EAAD     55/59   242#29   253/19
   OVRRUN  008E      6#40   246/10
 n P0PF    D004     19#41
 n P0PL    D00C     19#51
   P10COF  DE4D    182/61   183/ 5   184#15   184/26
 n P1PF    D005     19#42
 n P1PL    D00D     19#52
 n P2PF    D006     19#43
 n P2PL    D00E     19#53
 n P3PF    D007     19#44
 n P3PL    D00F     19#54
   PACTL   D302     22#22    34/58    59/21    59/28   236/18   248/39   249/ 6
                   249/23   258/18   351/ 6   352/38   356/10
   PADDL0  0270     14#33    42/27
 n PADDL1  0271     14#34
 n PADDL2  0272     14#35
 n PADDL3  0273     14#36
   PADDL4  0274     14#37    42/29
 n PADDL5  0275     14#38
 n PADDL6  0276     14#39
 n PADDL7  0277     14#40
 n PAI     C42C     54#49
   PAL     D014     20# 5    51/30
   PALNTS  0062     11#42    51/44   248/29   248/53   254/53   256/ 6   351/10
                   352/42   357/ 9   357/49
   PARMBL  02C9     15#46    71/26
   PBC     FD2A    351#41   352/52
   PBC1    FD2E    350/39   351#45   352/23
   PBCTL   D303     22#23    35/11    58/58    59/17    59/22    59/29   236/21
                   237/51   244/40   258/20
 n PBI     D100     21# 8
 n PBIRAM  D600     23# 8
   PBK     EDC7    241/58   244/49   256/37   256/59   258#15
   PBPNT   02DE     16# 6   362/44   363/14   363/26   364/30   365/ 7
   PBUFSZ  02DF     16# 7   361/52   363/21   364/ 5   365/43   367/ 9
   PCC     E51A    201/51   203#15
   PCC1    E529    203/27   203#33
   PCH     F2BE    287/37   289#25
   PCH1    F2C6    289#32   289/40
   PCH2    F2CF    289/28   289#38
   PCH3    F2E5    289/50   289#61
   PCI     EB9D    236/60   248#15
   PCI1    EBBC    248/32   248#36
   PCI2    EBC6    248#41   248/42
   PCI3    EBD4    248/20   248#50
```

```
  PCI4   EBE6    248/56  248#60
  PCI5   EBF0    249# 8  249/ 9
  PCI6   EC04    248/46  249#19
  PCI7   EC0E    249/20  249#25
  PCL    FF07    195/14  364#55
  PCOLR0 02C0     15#34   39/61
n PCOLR1 02C1     15#35
n PCOLR2 02C2     15#36
n PCOLR3 02C3     15#37
  PCS    C2C8     46/28   46/38   46/43   47/38   47/42   48#18  196/48
  PDEVN  0040      6#51  365/31
  PDID1  D803     24# 8   82/27
  PDID2  D80B     24#11   82/31
  PDIMSK 0249     14#18   32/52
  PDIOV  D805     24# 9   83/28
  PDIRQV D808     24#10   84/37
  PDVI   D1FF     21#12   32/51
  PDVMSK 0247     14#16   83/16   87/31
  PDVS   D1FF     21#16   82/25   82/44   83/37   84/36   84/43   87/26
                  87/38   89/16
  PDVV   D80D     24#12   82/35   87/61   88/ 7
  PENH   D40C     22#36   39/26
  PENV   D40D     22#37   39/24
  PHC    E89E     91/ 8  220/16  229#15
  PHE    EEBC    197/ 6  266#41
  PHE1   EEC2    266#54  266/61
  PHE2   EED3    267#12  267/19
  PHE3   EEE5    266/55  267#31
  PHE4   EEEB    267/13  267#40
  PHENTV E486     24#50  197/ 6
  PHG    E816    222/38  222/40  223#37
  PHG1   E81F    223#48  223/58
  PHG2   E827    223/44  223#55
  PHI    E898    197/12  228#43
  PHINIV E48C     24#52  197/12
  PHL    CA29     90#52  202/43  270/51
  PHL1   CA54     90/60   91/ 9   91/17   91#34
  PHO    EEF9    205/51  268#44
  PHO1   EF07    268/49  268#54
  PHP    E7BE    219/26  219/59  220#53  268/48
  PHP1   E7C1    221# 6  221/ 9
  PHPA   E7D4    221/ 6  221#24  221/33
  PHPAL  000A    220/61  221#33
  PHQ    E8C0    228/28  230#15
  PHQ1   E8CE    229/26  230#30
  PHQ2   E8D1    230/20  230#36
  PHQ3   E8DD    230/37  230#49
  PHR    E739     53/46  218#32
  PHR1   E745    218#48  219/12
  PHR2   E762    218/37  219#16
  PHR3   E76E    219#24  219/60  220/19
  PHR4   E776    218/54  219/ 5  219/ 8  219#31
  PHR5   E777    219/27  219#35
  PHR6   E799    219#51  220/13  220/17
  PHR7   E79B    219/20  219#58
  PHR8   E7A2    219/47  220# 7
  PHU    E915    197/ 9  230/26  233#44
  PHU1   E939     233/61  234#16
  PHU2   E953     234/11  234/14  234#37
  PHU3   E955     233/49  234#42
  PHUNLV E489      24#51  197/ 9
  PHW    E894     219/ 7  228#25
  PHX    E900     230/19  231#29
  PIA    D300      22#15   59/ 9
  PIN    FE99     195/19  361#15
  PIO    C933      83# 7  196/18  235/15
  PIO1   C943      83#23   83/31
  PIO2   C958      83/17   83/24   83#42
  PIO3   C95E      83/38   83#46
  PIR    C96E      60/58   60/60   84#18
  PIR1   C970      84#24   84/28
  PIR2   C976      84/25   84#32
  PLO    F1CA     282/23  282#40  299/38  336/38  337/ 5
  PLO0   F1CA     282#44  282/45
  PLO1   F1D1     282#51  282/54
n PLOT   0050       7#22
  PLY1   DD58     176#49  177/17
  PLY2   DD6F     176/56  177# 5
  PLY3   DD88     176/47  176/50  177/ 9  177/12  177#19
  PLYARG 05E0      18#34  176/39  176/40  177/14  177/15  185/ 5  185/ 6
                  185/15  185/16
  PLYCNT 00EF      12#40  176/38  176/46  177/11
  PLYEVL DD40      23#35  176/11  183/ 6  188/16
n PMBASE D407      22#48
  PMD    50D0     104/34  104/36  107#24
  PMD1   50D3     107#34  109/11
  PMD2   50E9     107/42  107/49  107#55
  PMD3   50FA     107/37  108#14
  PMD4   5114     108/15  108#36
  PMD5   511F     108/21  108/32  108/38  108#43
  PMD6   5120     108/51  108#57
  PMD7   5137     108/59  109#11
  PMI    C471      49/ 5   56#35
  PMI1   C484      56/41   56/44   56/47   56#55
  PMI2   C49A      57/ 9   57#18
  PMI3   C4A1      57/14   57#24
  PMI4   C4A9      57/12   57/20   57#33
  PMI5   C4B2      57#40   57/52
  PMI6   C4C8      57/44   57/49   57#56
  POD    530A     106/37  106/39  119#34
  POD1   53F0     119/55  120#11
  POD10  5448     120/12  121#10
  POD2   540C     120/18  120#23
  POD3   540E     120/21  120#25
  POD4   541C     120/29  120#34
  POD5   541E     120/32  120#36
  POD6   542C     120/40  120#45
  POD7   542E     120/43  120#47
  POD8   5443     120/54  120#61
  POD9   5445     120/59  121# 6
  POKEY  D200      21#24   59/ 5
  POKMSK 0010      10#30   32/34   33/10   33/27  243/ 6  243/ 8  243/58
                  243/60  250/28  250/53  251/19  252/24  252/25  274/61
                  275/ 5
  POP     FEC2     195/13   362#41
  PORTA   D300      22#19    35/ 6    41/50    41/59    59/24    59/31
  PORTB   D301      22#20    35/16    49/57    49/59    50/12    50/14    50/61
                    52/33    52/35    56/59    56/61    57/24    57/26    59/ 6
                    59/19    59/26    59/30    81/26    81/28   117/25   117/28
                   344/26   345/57
  POT0    D200      21#28    42/26
n POT1    D201      21#29
n POT2    D202      21#30
n POT3    D203      21#31
n POT4    D204      21#32
n POT5    D205      21#33
n POT6    D206      21#34
n POT7    D207      21#35
  POTGO   D20B      21#61    42/36
  PPB     FECB     195/16   362#60
  PPB1    FEE8     363/31   363#40
  PPD     F223      49/50    49/52   196/42   196/57   285#15
  PPM     FF4B     363/10   364/59   366#38
  PPM1    FF4F     366#47   367/20
  PPM2    FF57     366/48   366#53
  PPM3    FF5F     366/54   366#59
  PPM4    FF65     366/51   366/57   367# 9
  PPM5    FF6F     366/60   367#19
  PPO     E576     204/32   204/37   205#50
  PPP     FEF6     363/22   364#25
  PPRB    FEA1     361#32   364/34   364/35
  PPTMPA  024C      14#20    88/ 9    88/35    88/60    89/20
  PPTMPX  024D      14#21    88/10    88/36    89/21    89/22
  PRINTR  0050       5#51    55/11
  PRINTV  E430      24#23    55/12    60/46   195/ 6
  PRIOR   D01B      20#45    39/35
  PRNBUF  03C0      17#33   361/32   363/16   363/60
  PRS     C2CA      46/48    48#42
  PRS10   C346      50/43    50#47
  PRS11   C34B      50#54    50/56
  PRS12   C350      50/20    50#60
  PRS13   C35A      51/ 6    51#10
  PRS14   C383      51/32    51#39
  PRS15   C389      51/37    51#43
  PRS17   C393      51#56    51/59
  PRS18   C39E      52# 8    52/11
  PRS21   C3C4      52/29    52#44
  PRS22   C3DC      52/49    52/52    52#60
  PRS23   C3FA      53/13    53#21    53/22    53/25
  PRS24   C400      53/34    53#42
  PRS25   C413      53/38    53#50
  PRS26   C426      53/56    53/60    54#11
  PRS3    C2E4      49#24    49/39    49/46
  PRS4    C2EE      49/27    49#31
  PRS5    C2F8      49/34    49#38
  PRS6    C310      50/ 5    50#10
  PRS7    C31F      50/ 8    50#12
  PRS8    C32E      49/15    50#24
  PRS9    C33C      50/27    50#40    50/48
  PRVOPN  0081       6#27   204/25
  PSP     FEC1     195/15   195/18   361/60   362#25
  PST     FEA3     195/17  361#47  362/42
  PSTB    FE9F     361#30  361/53  361/54
  PTE     EC11     249#40  258/43  258/45
  PTIMOT  0314      17# 5  361/17  365/47  366/20
  PTL     EF26     268/56  268/58  270#10
  PTL1    EF38     270#27  270/33  270/52
  PTL2    EF3D     270/15  270/18  270#32
  PTL3    EF41     270/21  270#37
  PTL4    EF46     270#44  270/49
  PTO     C24F      38/46   43#23
  PTRIG0  027C      14#47   42/54
  PTRIG1  027D      14#48   42/47
n PTRIG2  027E      14#49
n PTRIG3  027F      14#50
  PTRIG4  0280      14#51   42/56
  PTRIG5  0281      14#52   42/49
n PTRIG6  0282      14#53
n PTRIG7  0283      14#54
  PTT     C252      40/24   43#39
  PUPBT1  033D      17#14   47/36   51/15
  PUPBT2  033E      17#15   47/40   51/17
  PUPBT3  033F      17#16   47/44   51/19
  PUPDIV  E480      24#48  196/57
  PUPVL1  005C       8#35   47/37   51/14
  PUPVL2  0093       8#36   47/41   51/16
  PUPVL3  0025       8#37   47/45   51/18
  PUTCHR  0008       5#12  205/28
  PUTDAT  0080       9#24   69/ 6  359/16
  PUTREC  0009       5#11   65/52
  PUTSEC  0050       7# 9   68/57
  PWS     C290      46#18   47/46  196/45
  PWS1    C2A1      46/33   46#42
  QTEMP   00D9      12#24  157/35  157/44  157/45  157/46  157/47  158/ 7
  RAMLO   0004      10#14   49/21   49/22   49/25   49/26   49/32   49/33
                    49/43   49/44   63/29   63/31   63/41   63/48   63/50
                    63/51   63/53   64/56   64/59   65/ 6
  RAMSIZ  02E4      16#11   52/47   60/29  115/60  273/20
  RAMSYS  0000       4#12   57/30
  RAMTOP  006A      11#53  273/21  275/51  275/59  278/16  279/14  298/16
                   320/53  324/11  324/23  325/14
n RANDOM  D20A      21#39
  RBLOKV  E47A      24#46   66/24  196/51
  RCB     FD8D     196/51  353/34  353#57
  RCB1    FDA9     354/22  354#28
  RCB2    FDAF     354/17  354#36
  RCB3    FDB1     353/28  354#40
  RDONLY  0087       6#33  209/37
  READ    0052       7#10   66/28  359/13
  REC     EAFD     238/47  240/21  244#24  249/15
  REC1    EB06     244/30  244#34
  REC2    EB16     244#44  244/59
  REC3    EB1D     244/45  244#53
  RECLEN  0245      14#14   71/43   71/55   72/54   75/20
  RECVDN  0039      11#10  244/35  244/58  246/32
  REDGE   0027       4#52   51/25
  RELADR  024A      14#19   74/51   74/57   75/11   75/14
  RES     C2AA      47#18   53/17  372/14
```

```
RES1    C2AD      47#28    47/29    47/32
RET     F665     300/ 5   311/42   311/49   311/54   312#43
RET1    F67B     312/51   312#57
RET2    F67E     312/55   312#59
RET3    F691     313/15   313#19   313/29
RET4    F69D     313/22   313#26
RET5    F6AB     311/52   312/60   313/ 7   313/10   313#34
RIR     C0CE      36#18    55/53   286/56   286/58
RIRGHI  0000       8#27   248/60
RIRGLN  0078       8#11   260/11
RIRGLP  0064       8#20   260/12
RIRGLX  EE13     248/58   260#11
RLEADH  FE91     351/12   360#14
RLEADL  FE93     351/11   360#17
RLEADN  0240       8# 9   360/14   360/17
RLEADP  01E0       8#18   360/15   360/18
RLR     C745      71#19   222/47
RLR1    C747      71#25    71/28
RLR2    C74F      71#32    71/57
RLR3    C77D      71#55    72/ 8
RLR4    C794      71/36    71/41    71/61    72#10
RMARGN  0053      11#32    51/26   296/39   297/18   299/28   311/30   314/48
                 314/51   322/36   326/59   327/27   328/47
ROD     F718     287/49   289/26   316#15
ROD1    F722     316/16   316#23
ROWAC   0070      11#58   313/52   313/54   313/55   313/57   334/42   335/17
                 335/30   335/32   335/35   335/37   335/43
ROWCRS  0054      11#33   282/51   287/26   287/52   295/16   295/22   295/40
                 295/41   298/51   299/31   301/44   302/35   303/57   304/17
                 308/58   309/ 6   311/55   312/47   312/59   313/32   314/55
                 318/42   321/ 5   322/ 9   322/12   325/47   326/39   326/55
                 327/14   327/30   328/20   329/46   330/19   330/50   330/53
                 333/38   334/44   335/48   335/50   336/51   336/55
ROWINC  02F8      16#26   333/48   333/59   335/49
RRC     F957     301/39   302/48   327/37   330#15   337/10
RRC1    F959     330#18   330/21
RSIRGN  000A       8#13   260/17
RSIRGP  0008       8#22   260/18
RSIRGX  EE17     248/54   260#17
RTCLOK  0012      10#32    38/10    38/14    38/17    38/27   256/51   257/23
                 357/ 7   357/39   357/55
RUNADR  02C9      15#47    71/51    71/53    72/48    72/53    73/ 6    73/ 9
                  73/12    73/13    73/53
RWS     F661     282/20   288/22   311/51   312#25   338/37
S0ER    DC62     155/49   158/18   166#15
S0ER1   DC64     166#18   166/21
S0L     DBEB     139/40   142/27   162#20   170/20
S0R     DC3A     153/36   164#51
S1R     DC3E     153/ 6   165#16
S2L     DBE7     161#60   171/52
SAL     DC9F     144/52   145/ 9   167/45   168#43
SAS     5510     104/26   124/11   124/24   124#60   130/52
SAVADR  0068      11#50   277/29   277/31   279/54   279/57   302/25   302/27
                 302/41   302/46
SAVIO   0316      17# 7   256/49   257/12   257/15
SAVMSC  0058      11#36   277/ 7   277/ 9   279/31   279/33   309/37   309/41
                 332/44   332/46
```

```
SBA     C73A     69/46   69/57   70#40
SBP     EB87    238/20  238/46  247#36  248/44  249/11
SBR     ED3D    249/14  256#32
SBR1    ED3D    256#34  256/47
SBR2    ED44    256/35  256#39
SBR3    ED4C    256/41  256#45
SBR4    ED68    256#58  257/13  257/17  257/26
SBR5    ED71    256/43  257# 7
SBR6    ED75    257/ 5  257#10
SBR7    ED96    257/20  257#28
SBS     F90C    295/24  327#52
SBS1    F917    327#59  328/44  328/53
SBT     5505    124/ 8  124/18  124#40
SBUFSZ  001D      9#32  367/ 5
SCB     FE3F    354/ 5  358#52  359/46
SCB1    FE70    359/14  359#18
SCC     F40C    296#55  297/40
SCC1    F40E    296/21  296#58  297/19  297/21
SCL     F997    298/47  312/44  332#15
SCL1    F9A1    332/20  332#25
SCL2    F9A3    332/23  332#27
SCR     F7F7    313/19  322#60
SCR1    F7FF    323#14  323/15
SCR2    F809    323#22  323/24
SCR3    F810    323#26  323/28
SCR4    F822    323/33  323#37  323/38
SCR5    F827    323/12  323#42
SCREDT  0045      5#49   55/17
SCRENV  E410     24#21   55/21   60/44  194/26
SCRFLG  02BB     15#28  301/20  301/41  313/20
SCRMEM  0093      6#45  308/10
SDA     F610    310/17  310#49
SDF     F569    278/10  278/23  306#15
SDI     F570    277/36  278/19  278/21  278/52  279/17  279/19  279/22
                279/25  279/32  279/34  279/37  279/39  279/41  279/47
                306/17  306#35  308/34  308/36
SDL     509E    104/33  106#27  113/13  122/ 9  127/19
SDLSTH  0231     13#47   39/28  106/50
SDLSTL  0230     13#46   39/30  106/48  279/43  279/45  279/53  279/56
SDMCTL  022F     13#45   39/32  106/ 5  106/35  106/53  274/57  280/33
                280/34  345/49  345/53  346/38  346/43
SDP     FF14    361/58  364/36  365#28
SDP1    FF2F    365/38  365#42
SEA     F60E    282/24  310#32
SEC     F20B    280/35  281/52  282/21  282/25  284#15  288/25  290/ 9
SEDS    C448     53/ 6   53/ 8   55#42
SEIOCB  0000      4#46   52/61   65/48
SEL     500C    104#15  120/ 7
SEL1    5043    104#46  104/48
SEL2    5049    104#52  104/54
SEL3    5059    105# 8  105/10
SEL4    5075    105#28  109/ 7
SEL5    507D    105/18  105#35
SEL6    5080    105/21  105#39
SEL7    5083    105/24  105#43
SEN     EA88    241#33  248/45  253/54
SEN1    EA9E    241#53  242/ 6
```

```
  SEN2    EAA5      241/54   242# 5
  SENDEV  E468       24#40   196/33   352/36
  SEP     F6AE      313#50   335/52   336/33
  SERIN   D20D       21#40   246/20   246/43
  SEROUT  D20D       22# 5    59/46   241/48   242/56   243/22
  SES     C8FC       81#21   285/16
  SETVBV  E45C       24#36   196/21   258/49   351/16   352/47
  SFL     E6D8      209/11   210/45   214#36
  SG8     F180      194/35   280#50   288/16
  SGNFLG  00F0       12#43   182/27   183/48   186/51   188/49
  SHC     E85D      226# 5   229/25   233/48
  SHC1    E86B      226#18   226/57
  SHC2    E87F      226/24   226/27   226#36
  SHC3    E887      226#44   226/53
  SHC4    E889      226/37   226/40   226#49
  SHFAMT  006F       11#57   281/20   283/41   309/52
  SHFLOK  02BE       15#31   273/24   292/18   292/22   292/33   292/44   294/15
  SHPDVS  0248       14#17    82/20    82/24    82/37    83/36    84/32    84/35
                     84/42    87/25    87/37    89/15
  SHT     E6FF      204/36   206/61   215#46
  SHT1    E70C      215/54   215#59
  SHT2    E70E      215/57   215#61
  SID     ECAF      237/52   238/21   253#38
  SID1    ECB1      253#44   253/50
  SID2    ECB3      253#46   253/47
  SIDWAY  0053        7#23   366/59
  SIL     DA5A      147/59   148/ 9   148/12   151#19
  SIN     EF6E      194/19   194/39   194/59   273#15
  SIO     E971       83/42   236#43
  SIO1    E983      236/56   237# 7
  SIO10   EA22      238/ 9   238/22   238/38   239#10
  SIO2    E980      237#13   239/15
  SIO3    E992      237#18   238/ 5
  SIO4    E9D9      237/54   237#61
  SIO5    E9E1      237/57   238#13
  SIO6    E9F3      238/14   238#26
  SIO7    EA0C      238/35   238#46
  SIO8    EA12      238/30   238#51
  SIO9    EA1C      238/52   238#61
  SIOINV  E465       24#39    60/52   196/30
  SIOV    E459       24#35    69/33   196/18   221/19   224/59   359/21   361/59
                    364/40
n SIZEM   D00C       20#24
n SIZEP0  D008       20#19
n SIZEP1  D009       20#20
n SIZEP2  D00A       20#21
n SIZEP3  D00B       20#22
  SKC     F983      291/28   293/21   305/18   331#25   348/15
  SKC1    F986      331#34   331/47
  SKC2    F98C      331#40   331/41
  SKCTL   D20F       22# 7    59/36   104/25   236/26   250/26   250/50
  SKRES   D20A       21#60   245/49   250/51
  SKSTAT  D20F       21#42    40/44    40/60   124/20   245/48   256/45   257/10
                    257/33   257/35
  SL8     DBA1      139/ 7   159#42
  SL81    DBA5      159#51   159/55
  SL82    DBAC      159/52   159#59
```

```
SLC     F88E     296/58  299/40  300/17  301/47  302/49  303/39  303/55
                 312/10  313/34  325#41  327/53  328/22
SLC1    F896     325#52  325/61
SLC2    F8A9     325/54  326# 8
SLD     53A4     114/28  116/18  117#23  120/ 5
SLFTSV  E483      24#49   81/30  196/60
SMEM1   5161     109/21  109#47
SMEM2   5171     109/28  110# 8
SMEM3   5181     109/37  110#29
SMEM4   51F5     111/16  111#26
SMEM5   5271     112/10  112#34
SML     DBED     162/ 5  162#40
SML2    DBEF     162#43  162/51
SMS     F9A6     297/59  323/42  332#43
SNL     DC9D     145/24  145/27  167/ 6  167/12  168#20
SNP     C9AF      83/23   87#15   88/48
SNP1    C9AF      87#19   87/33
SNP2    C9BB      87/20   87#31
SOP     EF8E     194/33  273#51
SOUNDR  0041      11#18  236/25  251/28
SPB     F1A4     194/36  281#43
SPB1    F1B1     281/49  281#54
SPECIL  000E       5#15  203/26  203/29
SPQ     F1E9     283#27  284/25  301/22
SPQ1    F1F2     283#41  283/45
SPQ2    F1FA     283/42  283#49
SQR10   DF66     188/ 5  188/ 6  189#13
SRC     F94C     301/16  302/19  326/35  329#43  336/47
SRC1    F94E     329#46  329/49
SRR     DC40     164/53  165#35
SRR1    DC46     165#40  165/52
SRR2    DC48     165#42  165/46
SRTIMR  022B      13#41   40/57   41/ 7   41/16   41/46  345/48
SSC     E670     201/34  202/27  204/27  205/52  207/44  209/39  211#15
SSD     F82A     304/38  323#58
SSD1    F82E     324#10  324/16
SSD2    F839     324/12  324#18
SSD3    F847     324/19  324#31
SSD4    F855     324/37  324#43  325/ 7  325/10
SSD5    F862     324#51  324/54
SSD6    F87C     324/24  324/59  325#14
SSD7    F889     325#22  325/24
SSE     F5A0     278/ 7  278/45  308#32
SSFLAG  02FF      16#34   34/24  282/44  345/17  345/19
SSKCTL  0232      13#48  236/24  250/ 6  250/25  250/47  250/49  257/34
SSM     5759     113/17  113/19  122/14  122/16  122/57  123/12  123/14
                 125/26  125/44  127/24  134#15
SSM1    5766     134#23  134/28
SSP     F9AF     194/38  333#17
SSP1    F9BE     333/27  333#32
SSP10   FA6F     335/41  335#47
SSP11   FA7C     335/39  335/45  335#54
SSP12   FA95     336/ 7  336#13
SSP13   FAAA     336/14  336#27
SSP14   FAB0     336/19  336/22  336/25  336/28  336#32
SSP15   FAB5     336/ 5  336/11  336#37
SSP16   FAC9     336#51  337/ 6
```

```
SSP17   FAE6     336/58  337# 8
SSP18   FAEF     336/43  337#14
SSP19   FB01     335/25  337/24  337#30
SSP2    F9BF     333/24  333#34
SSP3    F9F1     333/54  334#11
SSP4    FA19     334/18  334/31  334#37
SSP5    FA1F     334#41  334/46
SSP6    FA40     334/56  334/60  335# 9
SSP7    FA4D     335#21  337/26
SSP8    FA56     335/23  335#29
SSP9    FA61     335/33  335#37
SSR     EC56     250/30  251#15
SSR1    EC6A     251/29  251#33  251/36
SSR2    EC83     251/44  251#49
SST     F21E     194/17  194/37  194/53  194/54  194/57  280/55  284/19
                 284/22  284#41  286/48  294/35  315/47  331/10  337/30
SST1    F226     284/43  286# 7
SSV     EDE2     248/37  248/61  249/13  254/16  258#42
ST1K    008F      26#19  114/12  115/56  115/59
ST3000 3000       26#44  109/24  109/34  110/51  111/47  112/23  121/25
                 121/26  121/27  134/24  137/ 8  137/ 8  137/11  137/11
                 137/36  137/36  137/37  137/38  137/38  137/39  137/40
                 137/41  137/42  137/43  137/44  137/45
ST3002 3002       26#45  132/47
ST3004 3004       26#46  137/45
ST300B 300B       26#47  127/29
ST301C 301C       26#48  120/25
ST301E 301E       26#49  120/36
ST3020 3020       26#50  113/23  116/43  120/47  137/ 9  137/ 9
ST3021 3021       26#51  123/38  123/40
ST3022 3022       26#52  137/43
ST3024 3024       26#53  113/42  117/ 6  137/10  137/10
ST3028 3028       26#54  137/37
ST3038 3038       26#55  114/17  119/ 6  119/ 7  119/10  119/11
ST303C 303C       26#56  129/16  129/17
ST304C 304C       26#57  137/42
ST3052 3052       26#58  126/ 5  126/ 6
ST3062 3062       26#59  132/48
ST306D 306D       26#60  126/24  126/26
ST3072 3072       26#61  137/44
ST3092 3092       27# 5  137/40
ST309E 309E       27# 6  129/34  129/35
ST30AB 30AB       27# 7  137/41
ST30B7 30B7       27# 8  137/39
ST30C1 30C1       27# 9  132/43
ST30C2 30C2       27#10  132/51
ST30C7 30C7       27#11  128/41  128/42
ST30CA 30CA       27#12  128/55  128/56
ST30F8 30F8       27#13  128/37  128/38
ST3100 3100       27#14  112/14
ST3121 3121       27#15  132/44
ST3122 3122       27#16  132/49
ST313C 313C       27#17  129/24  129/25
ST3150 3150       27#18  127/36
ST3154 3154       27#19  128/17  128/18
ST3181 3181       27#20  132/45
ST3182 3182       27#21  132/50
```

```
  ST3186 3186     27#22  128/27  128/28
  ST318C 318C     27#23  129/20  129/21
  ST31B0 31B0     27#24  127/37
  ST31C2 31C2     27#25  132/52
  ST31CA 31CA     27#26  129/ 6  129/ 7
  ST31EE 31EE     27#27  129/38  129/39
  ST31F1 31F1     27#28  132/46
  ST3210 3210     27#29  127/38
  ST321A 321A     27#30  128/59  128/60
  ST3248 3248     27#31  128/45  128/46
  ST3270 3270     27#32  127/39
  ST32D0 32D0     27#33  127/40
  STACKP 0318     17# 9  236/48  258/23
  STADR1 009E     26#31  133/29  133/31  133/42  133/43  133/46  133/49
                 133/53  134/19  134/27  369/57  370/11  370/18  370/21
                 370/23  370/27
  STADR2 00A0     26#32  133/33  133/35  133/50  133/54  370/24  370/28
  STATC  0053      7#11   61/51   69/10   69/43  361/55  365/37
n STATIS 000D      5#14
  STATUS 0030     10#59  238/57  238/61  239/34  240/23  240/40  240/46
                 240/50  241/ 9  241/38  244/37  245/23  245/58  246/11
                 246/27  258/22
  STAUT  0082     26# 9  104/22  105/31  116/ 8  122/20  123/60  129/60
  STB    F495    300#32  338/49
  STBL   00A2     26#33  107/46  107/47  108/29
  STCDA  009A     26#28  127/49  127/56  127/59  127/61
  STCDI  0099     26#27  127/47  127/51  128/ 5  128/ 6  128/ 7
  STCHK  0088     26#16  368/24  368/25  368/53  368/56  369/27  369/28
                 370/12  370/13  370/16
  STH    5000    103#25  196/60
  STICK0 0278     14#42   41/61   42/43
  STICK1 0279     14#43   41/55
  STICK2 027A     14#44   42/ 6
  STICK3 027B     14#45   41/57
n STIMER D209     21#59
  STJMP  0083     26#10  106/ 9  106/49  106/51  111/18
  STK    5450    105/43  121#55  130/15
  STK1   5462    122#13  124/14  124/25
  STK10  54F8    124#20  124/22
n STK2   5470    122#25
  STK3   5483    122/21  122#39  122/41  122/44
  STK4   5491    122/30  122#50
  STK5   54A0    122/54  122#61
  STK6   54B0    123/ 7  123#18
  STK7   54D3    123#48  123/50
  STK8   54DE    123#60  125/27  125/45  126/ 7  126/27
  STK9   54F5    123/61  124#18
  STKST  008A     26#15  106/29  120/11
  STLM   00A4     26#35  114/ 7  114/27  114/29  114/31
  STM    5291    103/43  105/35  113# 6  122/35
  STM1   52A9    113#23  116/13
  STM10  5326    114#51  115/28
  STM11  5328    114#55  115/22
  STM12  532A    114#57  114/61
  STM13  5335    115#11  115/17
  STM14  5348    115/13  115#34
  STM15  5356    114/39  115#41
```

```
STM16   5359     115/30   115#45
STM17   535E     114/20   114/23   115/37   115#48
STM18   5368     115#55   116/24
STM19   5370     116/ 9   116#17
STM2    52C2     113/31   113#36
STM20   5388     115/49   116#22
STM3    52C4     113/34   113#38
STM4    52C7     113/25   113#42
STM5    52E0     113/50   113#55
STM6    52E2     113/53   113#57
STM7    52E5     113/44   113#61
STM8    52F7     114#16   115/61
STM9    5318     114#37   114/42
STMVAL  0093      26#22   115/ 8   115/12   115/15
STNOT   0098      26#26   127/15   130/41   130/48
STPAG   0090      26#20   114/10   114/11   114/38   114/58   115/11   115/26
                 115/58
STPASS  0087      26#12   104/59   107/35   107/58   108/46
STPC    0092      26#21   114/47   115/27
STRIG0  0284      14#56    42/12
STRIG1  0285      14#57    42/17
STRIG2  0286      14#58    42/14
STRIG3  0287      14#59    42/19
STS     FF44     362/ 7   366#18
STSEL   0086      26#11   105/15   105/29   105/60   108/25   108/27   108/43
STSKP   0094      26#23   121/60   122/25   122/27   122/28
STSMM   008D      26#17   114/ 5   114/18   115/48   115/52   116/23   118/61
                 119/ 9
STSMP   008E      26#18   114/ 9   114/16   115/53   118/60
STSPP   0088      26#13   108/14   108/31   108/41
STTIME  0080      26# 8   104/20   104/21   108/50   108/53   108/57
STTMP1  0095      26#24   123/39   123/41   123/49   123/54   123/56
STTMP2  009B      26#29   131/16   131/17   131/26   131/27   131/54   131/56
                 131/59
STTMP3  009D      26#30   131/21   131/29   131/32
STTMP4  00A3      26#34   131/23   131/34
STTMP5  00A5      26#36   117/24   117/27
STV     5557     105/39   116/20   126#57
STV1    555C     127# 9   130/ 6
STV2    5560     127#14   130/10
STV3    557D     127#35   127/42
STV4    5599     127#51   128/ 9
STV5    5663     129/56   130#10
STV6    5666     129/61   130#14
STVOC   0097      26#25   127/10   127/25   129/52   129/53   129/54   130/34
STW     ECC0     238/29   254#15
SUBTMP  029E      15#17   307/36   307/47
SUC     5773     104/30   113/15   122/ 5   127/ 5   134#58
SUCA    578C     134/60   135#18
SUCB    5790     135/ 6   135#23
SUCC    5794     135/ 9   135#28
SUCCES  0001       6#24    73/15   206/19   239/ 5   240/24   241/37   244/36
                 269/10   274/58   286/ 7   315/23   353/ 7   353/40   355/ 6
                 355/37   356/ 5   363/35   365/12
SUCD    5798     135/12   135#33
SUE     DCCF     155/ 5   156/56   171#20
SUP     DCE0     155/14   157/ 6   171#45
```

```
SUPERF  03E8      17#34   289/ 6   289/49   290/54   293/56
SVN     566C     128/23   128/33   128/51   129/12   129/30   129/44   130#30
SVP     C272      44#48   196/21   232/15
SVP1    C27C      44#61    45/ 5
SVR     572A     104/40   106/43   116/41   116/61   133#20
SVR1    5741     133#41   133/51   133/55
SVR2    574A     133/44   133#48
SWA     F962     287/19   287/38   288/18   288/27   288/60   289/34   290/10
                 315/45   330#38
SWA1    F968     330#50   330/57
SWA2    F980     330/44   331#10
SWPFLG  007B      12# 7   275/28   312/50   315/42   326/48   330/61   331/ 6
                 332/19
SYSVBV  E45F      24#37   196/24
SZA     F60A     280/54   302/30   310#15   327/ 8
TAB     F47A     299#55   338/34
TAB1    F47A     299#57   300/13
TAB2    F48B     299/60   300#11
TAB3    F492     300/ 7   300#17
TABMAP  02A3      15#22   275/36   304/25   311/57   317/59   317/61   318/24
                 318/25   319/16   319/35   321/60   322/14
TAGM    EE40     262#11   275/57
TAIC    F849     283/11   339#30
TARS    57DC     133/28   133/30   133/32   133/34   137# 8
TARV    FFD7     369/56   370#39
TAVD    568C     131/25   132#21
TBTM    EEB4     265#19   316/44
TCCR    F80D     289/56   289/61   290/ 6   329/19   338#12   338/60
TCCRL   0030     329/17   338#60
TCDA    5716     127/52   127/54   132#43   132/54
TCDAL   0014     128/ 8   132#54
TCKD    F851     273/26   273/28   339#54
TCVO    E72D     203/34   212/40   217#10
TDLE    EE2D     261#42   278/28   278/34
TDLV    EE5D     262#37   277/20
TDP     DC93     166/51   167#35
TDP1    DC9C     167/40   167#49
TDSC    F808     275/44   337#49
TDSM    EEAD     265#11   309/50
TEMP    023E      13#59    13/61   240/12   240/16   240/27
TEMP1   0312      16#60   219/38   219/41   219/44   219/46   222/17   224/50
                 224/53   226/ 9   226/10   226/26   231/33   231/36   231/37
                 254/48   255/ 7
TEMP2   0313      16#61   222/15   224/54   226/23
TEMP3   0315      17# 6   222/19   223/41   223/43   223/48   223/56   256/55
                 257/19
TEX     C7D5      74#44    80/30    80/31    80/40
TEX1    C7E6      74/47    74#57
TEX2    C7F9      74/61    75#10
TEX3    C80C      75/ 6    75#18
TEX4    C821      75/28    75#32
TEX5    C830      75/38    75#45
TEX6    C839      75/36    75/41    75#51
TEX7    C84C      75/60    76# 8
TEX8    C850      74/53    75/53    75/58    76/ 6    76#12    76/44
TFKD    FC11     273/31   273/33   343#47
THAV    C42E      52/ 8    55#11    55/26
```

```
  THAVL   000F     52/ 6   55#26
  TIAC    FB4D    320/28  339#41
  TIHV    C44B     51/56   55#53   56/16
  TIHVL   0026     51/54   56#16
  TIMER1  030C     16#56  254/42  254/44  254/47  254/51  256/52  256/53
  TIMER2  0310     16#59  254/38  254/39  254/41  254/45  254/49
  TIMFLG  0317     17# 8  244/53  248/41  249/ 8  249/42  256/40  256/61
                  258/51
  TIMOUT  008A      6#36  240/51  245/22
  TINDEX  0293     15#12  275/26
  TIRQ    C0CF     33/ 6   36#50   36/59
  TIRQL   0008     32/61   36#59
  TLSC    EE6D    263#11  309/13
  TMCC    EE7D    263#36  311/24  315/17
  TMNT    544A    114/37  121#22  121/29
  TMNTL   0006    114/35  121#29
  TMPCHR  0050     11#29  283/50  283/54
n TMPCOL  02B9     15#27
n TMPLBT  02A1     15#20
  TMPROW  02B8     15#26  329/47  330/18
  TMRC    EE8D    264#10  312/57  314/54
  TMSK    FB04    309/46  337#38
  TNC     DBAF    158/46  160#21
  TNDD    56B6    130/42  132# 8
  TOADR   0066     11#48  320/56  320/60  321/18  321/26  321/30
  TOIH    C0D7     33/42   37#14
  TONE1   0002      7#57  352/21
  TONE2   0001      7#58  350/37
  TPOS    CA21     84/34   87/32   89#43   89/52
  TPOSL   0008     83/21   84/22   88/46   89#52
  TPFV    EDF9    255/34  255/37  259#25
  TRAMSZ  0006     10#15   49/45   52/45   52/54   53/33   53/55   57/35
                   57/38   57/40   57/42   57/43   57/47   57/48   57/51
                   60/28
  TRIG0   D010     19#56   42/11
  TRIG1   D011     19#57   42/16
n TRIG2   D012     19#59
  TRIG3   D013     19#60   30/22   39/18   46/26   56/39
  TRNRCD  0089      6#35  208/55
  TRPR    C8E4     71/50   71/52   80#30
  TRSC    EE9D    264#36  309/23  309/45
  TRUE    FFFF      4# 9    4/11
  TSFR    FB3D    289/56  339#12
n TSIH    ECA9    253#18
  TSKP    5545    122/26  126#35  126/39
  TSKPL   0012    122/29  126#39
  TSMA    EE1D    261#17  276/ 7
  TSMC    579C    123/35  135#46
  TSTAT   0319     17#10  238/56  241/10
  TSTD    57F6    134/20  137#36
  TSTL    57EC    134/18  137#19
  TSTO    CA57     92# 8  134/16
  TTXT    CA61     92/ 8   92/ 9   92/10   92/11   92/12   92/13   92/14
                   92/15   92/16   92/17   92#25  134/23
  TVN     DBBB    139/11  160#51
  TVN1    DBCF    161#17  161/30  161/37
  TVN2    DBD0    160/26  161#21
  TVN3    D8D2     161/10   161/13   161#26
  TVN4    D8D8     161/ 7   161#34
  TVN5    D8E2     160/61   161/27   161/35   161#41
  TXT0    CA61      92/ 8    92#33    92/40
  TXT0L   0013      92#40   137/19
  TXT1    CA74      92/ 9    92#48    92/50
  TXT1L   0003      92#50   137/20
  TXT2    CA77      92/10    93# 8    93/15
  TXT2L   0013      93#15   137/21
  TXT3    C832      92/11    96#34    96/54
  TXT3L   0013      96#54   137/22
  TXT4    C845      92/12    97# 8    97/12    97/20
  TXT4L   0004      97#12    97/22   137/23
  TXT5    C845      92/13    97#20
  TXT5L   0004      97#22   137/24
  TXT6    C849      92/14    97#30    97/34
  TXT6L   0003      97#34   137/25
  TXT7    CA8A      92/15    93#23    96/26
  TXT7L   00A8      96#26   137/26
  TXT8    C84C      92/16    97#42    97/46
  TXT8L   0003      97#46   137/27
  TXT9    C84F      92/17    98# 8    98/12
  TXT9L   0007      98#12   137/28
  TXTCOL  0291      15#11   280/28
  TXTMSC  0294      15#13   275/53   275/55
n TXTOLD  0296      15#14
  TXTROW  0290      15#10   280/26   330/52   330/55
n USAREA  0480      18#12
  VBREAK  0206      13#21    35/33
  VCOUNT  D40B      22#35   104/52   256/50   257/22   277/13   323/26   323/37
                   331/38   331/40
  VCS     FF86     368#52   369/34
  VCS1    FF90     368/54   368/57   369# 5
n VDELAY  D01C      20#46
  VDSLST  0200      13#18    30/55   104/35   104/37   106/38   106/40   275/16
                   275/18   286/57   286/59
  VFR     FF73      49/61   113/30   368#19
  VFR1    FF79     368#29   368/31
  VGC     FFFF       4#11    41/56    42/ 5    42/13    42/18    42/28    42/48
                    42/55    51/49
  VIMIRQ  0216      13#29    31/49
  VINTER  0204      13#20    35/17
  VKEYBD  0208      13#22    37/15
  VPIRQ   0238      13#53    32/57    60/59    60/61
  VPRCED  0202      13#19    35/ 7
  VSCROL  D405      22#47    39/49
  VSERIN  020A      13#23    32/36    37/21
  VSEROC  020E      13#25    37/19
  VSEROR  020C      13#24    37/20
  VSFLAG  026C      14#28    39/39    39/44    39/47   323/14   323/18   323/22
  VSR     FF92      50/ 7   113/49   369#25
  VTIMR1  0210      13#26    37/18
  VTIMR2  0212      13#27    37/17
  VTIMR4  0214      13#28    37/16
  VVBLKD  0224      13#36    43/ 8
  VVBLKI  0222      13#35    31/20
  WARMST  0008      10#20    48/46    49/14    49/20    57/ 8    61/34    66/51
```

```
                218/36
WARMSV E474      24#44   196/45
WCA    EA37     239#59   254/17
WCA1   EA6F     240/35   240#45
WCA2   EA73     240/25   240/41   240#50
WCA3   EA80     240/52   241# 5
WCA4   EA82     240/29   240/32   240/58   241# 9
WCB    FE7C     355/18   356/20   356/35   359#40
WFK    FE36     358/ 7   358#29
WFR    C0DF      37#41    39/20
WFR1   C0DF      37#45    37/45
WIRGHI 0000       8#26   248/36
WIRGLN 00B4       8#10   260/ 8
WIRGLP 0096       8#19   260/ 9
WIRGLX EE11     248/34   260# 8
WLEADH FE8D     352/44   360# 8
WLEADL FE8F     352/43   360#11
WLEADN 0480       8# 8   360/ 8   360/11
WLEADP 03C0       8#17   360/ 9   360/12
WMODE  0289      15# 5   350/35   351/46   352/20   355/57
WOR    C86D      77#24    80/36    80/37
WOR1   C88A      77/37    77#41
WRITE  0057       7#12    68/60   366/42
WRONLY 0083       6#29   207/42
WSIRGN 000F       8#12   260/14
WSIRGP 000D       8#21   260/15
WSIRGX EE15     248/30   260#14
WSYNC  D40A      22#50    44/59   104/46   107/55   118/36   347/18
XCL    E57C     201/57   203/40   206#15
XCL1   E588     206/22   206#30
XFMFLG 00F1      12#46   182/25   182/47   183/12   187/39   187/47   188/28
                188/38
XFORM  DE95     184#59   188/ 7
XGT    E582     203/46   207#32
XGT1   E58A     207#44   207/49
XGT2   E58D     207/38   207#48
XGT3   E5D0     207/55   208# 8   208/33
XGT4   E5F0     208/17   208/23   208#32
XGT5   E5FB     208#43   208/51
XGT6   E60C     208/45   208#60
XGT7   E618     208/10   208/28   208/39   209#11
XIR    C0CD      35#57    55/54    55/55    55/56    55/61    56/ 5    56/ 6
XITVBV E462      24#38   196/27
XMTDON 003A      11#11   241/43   242/ 5   243/54
XOP    E53F     203/35   204#15
XOP1   E547     203/22   204#27   205/20   207/ 5
XOP2   E54A     204/21   204#31
XPT    E61E     203/48   209#27
XPT1   E626     209#39   209/44
XPT2   E629     209/33   209#43
XPT3   E63A     209/50   209#60   210/30
XPT4   E640     209/56   210# 7
XPT5   E659     210/18   210#28
XPT6   E66A     210/12   210/24   210/36   210#45
XSS    E597     203/43   206#51
XSS1   E5A2     206/57   207# 9
ZCHAIN 004A      11#24    99/31   218/42   218/44   218/50   218/53   218/58
```

```
                  218/59  218/60  226/12  226/14  226/19  226/22  226/49
                  226/50  229/32  229/36  229/37  229/38  229/40  229/42
                  230/43  230/45  230/52  230/56  231/ 6  231/ 9  231/31
                  231/34  233/54  233/56  233/58  233/59  234/ 8  234/10
                  234/17  234/20  234/23  234/25  234/28  234/31
  ZF1     DA46     23#30  149/37
  ZFR0    DA44     23#29  139/21  146/32  149/14  156/ 7  164/30  171/59
n ZHIUSE  02CD     15#49
  ZIOCB   0020     10#39  270/45
  ZLOADA  02D3     15#54  222/43
  ZTEMP1  00F5     12#52  147/55  148/33  155/20  155/24  155/37  155/41
                  155/53  157/ 7  158/13  171/57
  ZTEMP3  00F9     12#54  148/ 6  148/ 8  148/17  148/20  165/36  165/50
  ZTEMP4  00F7     12#53  146/29  146/31  146/39  146/40  147/36  147/37
                  148/ 5  148/ 7  148/16  148/18  148/19  148/21  148/26
                  148/27  148/28  148/32  148/45  148/47  148/48  148/50
                  148/54  148/56  151/21  151/22  152/33  152/37  165/37
                  165/38  165/51  165/58  166/45  166/54  167/39  174/47
                  174/58
  ZXLY    DA48    139/18  150#17
  ZXLY1   DA4A    150#21  150/24
  $            $   $  $$$$$  $   $   $$$ 
 $ $           $   $    $    $$  $  $   $
$   $          $   $    $    $ $ $  $   $
$   $          $   $    $    $  $$  $   $
$$$$$          $   $    $    $   $  $   $
$   $    $$     $ $     $    $   $  $   $
$   $    $$      $    $$$$$  $   $   $$$ 

DEST=A.VINO  USER=A.VINO  QUEUE=LPT  DEVICE=@p1


                PAGES=408


PATH=:UDD:SYSTEMS:OS:REV2:OS.PRN
  $    $$$$$  $   $  $$$$     $  
$ $ $  $      $$  $  $   $  $ $ $
 $$$   $      $ $ $  $   $   $$$ 
  $    $$$$   $  $$  $   $    $  
 $$$   $      $   $  $   $   $$$ 
$ $ $  $      $   $  $   $  $ $ $
  $    $$$$$  $   $  $$$$     $  
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$+


Atari S/W Development

AOS/VS REV 03.07
AOS/VS XLPT REV 03.00
```