ATARI HOME COMPUTER SYSTEM

TECHNICAL REFERENCE NOTES

includes:

Operating System User’s Manual
Operating System Source Listing
and
Hardware Manual

TO ALL PERSONS RECEIVING THIS DOCUMENT

Reproduction is forbidden without the specific written permission of
ATARI, INC. Sunnyvale, CA 94086. No right to reproduce this document,
nor the subject matter thereof, is granted unless by written agreement with,
or written permission from the Corporation.
TO ALL PERSONS RECEIVING THIS DOCUMENT

Reproduction is forbidden without the specific written permission of ATARI, INC. Sunnyvale, CA 94086. No right to reproduce this document, nor the subject matter thereof, is granted unless by written agreement with, or written permission from the Corporation.

Every effort has been made to ensure that this manual accurately documents this product of the ATARI Home Computer Division. However, due to the ongoing improvement and update of the computer software and hardware, ATARI, INC. cannot guarantee the accuracy of printed material after the date of publication and disclaims liability for changes, errors, or omissions.
# PREFACE

18

# 1 INTRODUCTION

18

## GENERAL DESCRIPTION

18

### OF THE ATARI COMPUTER SYSTEM

20

Conventions Used in This Manual

<table>
<thead>
<tr>
<th>Conventions</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hexadecimal Numbers</td>
<td>20</td>
</tr>
<tr>
<td>Memory Addresses</td>
<td>20</td>
</tr>
<tr>
<td>Kilobytes of Memory</td>
<td>20</td>
</tr>
<tr>
<td>Pascal as an Algorithm-Specification Language</td>
<td>20</td>
</tr>
<tr>
<td>Memory Layouts</td>
<td>20</td>
</tr>
<tr>
<td>Backus-Naur Form (BNF)</td>
<td>21</td>
</tr>
<tr>
<td>OS-Equate Filenames</td>
<td>21</td>
</tr>
</tbody>
</table>
2 OPERATING SYSTEM FUNCTIONAL ORGANIZATION

Input/Output Subsystem
Interrupt Processing
Initialization

Power-Up
System Reset

Floating Point Arithmetic Package

3 CONFIGURATIONS

Program Environments

Blackboard Mode
Cartridge
Diskette-Boot
Cassette-Boot

RAM Expansion
Peripheral Devices

Game Controllers
Program Recorder
Serial Bus Devices

4 SYSTEM MEMORY UTILIZATION

RAM Region

Page 0
Page 1
OS Data Base
User Workspace
Boot Region
Screen Display List and Data
Free Memory Region
Cartridges A and B
Mapped I/O
Resident OS and Floating Point Package ROM
Central Data Base Description
Memory Dynamics

System Initialization Process
Changing Screen Modes

5 I/O SUBSYSTEM

Central I/O Utility
CIO Design Philosophy

DEVICE INDEPENDENCE
DATA ACCESS METHODS
MULTIPLE DEVICE/FILE CONCURRENCY
UNIFIED ERROR HANDLING
DEVICE EXPANSION

CIO CALLING MECHANISM

HANDLER ID -- ICHID [0340]
DEVICE NUMBER -- ICDNO [0341]
COMMAND BYTE -- ICCMD [0342]
STATUS -- ICSTA [0343]
BUFFER ADDRESS
  ICBAL[0344] AND ICBAH [0345]
PUT ADDRESS --
  ICPTL [0346] AND ICPTH [0347]
BUFFER LENGTH/BYTE COUNT --
  ICBL [0348] and ICBLH [0349]
AUXILIARY INFORMATION --
  ICAX1 [034A] and ICAX2 [034B]
REMAINING BYTES (ICAX3-ICAX6)
CIO Functions

OPEN -- Assign Device/Filename to IOCB and Ready for Access

CLOSE -- Terminate Access to Device/File and Release IOCB

GET CHARACTERS -- Read n Characters (Byte-Aligned Access)

PUT CHARACTERS -- Write n Characters (Byte-Aligned Access)

GET RECORD -- Read Up To n Characters (Record-Aligned Access)

PUT RECORD -- Write Up To n Characters (Record-Aligned Access)

GET STATUS -- Return Device-Dependent Status Bytes

SPECIAL -- Special Function

Device/Filename Specification

I/O Example

Device Specific Information

Keyboard Handler

CIO Function Descriptions

Theory of Operation

Display Handler (S:)

Screen Modes

TEXT MODE 0

TEXT MODES 1 AND 2

GRAPHICS MODES (Modes 3 Through 11)

SPLIT-SCREEN CONFIGURATIONS

CIO Function Descriptions

User-Alterable Data Base Variables

Theory of Operation

Screen Editor (E:)

CIO Function Descriptions

User-Alterable Data Base Variables

Cassette Handler (C:)

CIO Function Descriptions

Theory of Operation

File Structure
Printer Handler (P:)
CIO Function Descriptions 76
Theory of Operation 76

Disk File Manager (D:)
CIO Function Descriptions 78
Device/Filename Specification 79
Filename Wildcarding 81
Special CIO functions 82
Theory of Operation 84
FMS Diskette Utilization 87

FMS BOOT RECORD FORMAT 89
BOOT PROCESS MEMORY MAP 90
VOLUME TABLE OF CONTENTS 92
FILE DIRECTORY FORMAT 93
FMS FILE SECTOR FORMAT 94

Non-CIO I/O 95
Resident Device Handler Vectors 96
Resident Diskette Handler 96
Diskette Handler Commands 97

Serial Bus I/O 99

6 INTERRUPT PROCESSING 101
Chip-Reset 102
Nonmaskable Interrupts 103
Stage 1 VBLANK Process 104
Stage 2 VBLANK Process 105

Maskable Interrupts 107
Interrupt Initialization 108
System Timers 109
Usage Notes 110

POKEY Interrupt Mask 111
Setting Interrupt and Timer Vectors 112
Stack Content at Interrupt Vector Points 113
Miscellaneous Considerations 114

Flowcharts 115
7 SYSTEM INITIALIZATION

Power-Up Initialization (Coldstart) Procedure
System Reset Initialization (Warmstart) Procedure

8 FLOATING POINT ARITHMETIC PACKAGE

Functions/Calling Sequences

- ASCII to Floating Point Conversion (AFP)
- Floating Point to ASCII Conversion (FASC)
- Integer to Floating Point Conversion (IFP)
- Floating Point to Integer Conversion (FPI)
- Floating Point Addition (FADD)
- Floating Point Subtraction (FSUB)
- Floating Point Multiplication (FMUL)
- Floating Point Division (FDIV)
- Floating Point Logarithms (LOG and LOG10)
- Floating Point Exponentiation (EXP and EXP10)
- Floating Point Polynomial Evaluation (PLYEVL)
- Clear FRO (ZFRO)
- Clear Page-Zero Floating Point Number (ZF1)
- Load Floating Point Number to FRO (FLDOR and FLDOP)
- Load Floating Point Number to FR1 (FLD1R and FLD1P)
- Store Floating Point Number From FRO (FSTOR and FSTOP)
- Move Floating Point Number From FRO to FR1 (FMOVE)

Resource Utilization
Implementation Details

9 ADDING NEW DEVICE HANDLERS/PERIPHERALS

Device Table
CIO/Handler Interface
- Calling Mechanism
- Handler Initialization
- Functions Supported
- Error Handling
- Resource Allocation

- ZERO-PAGE RAM
- NONZERO-PAGE RAM
- STACK SPACE

Handler/SIO Interface
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling Mechanism</td>
<td>142</td>
</tr>
<tr>
<td>Functions Supported</td>
<td>144</td>
</tr>
<tr>
<td>Error Handling</td>
<td>144</td>
</tr>
<tr>
<td>Serial I/O Bus Characteristics and Protocol</td>
<td>145</td>
</tr>
<tr>
<td>Hardware/Electrical Characteristics</td>
<td>145</td>
</tr>
<tr>
<td>Serial Port Electrical Specifications</td>
<td>147</td>
</tr>
<tr>
<td>Bus Commands</td>
<td>147</td>
</tr>
<tr>
<td>COMMAND FRAME</td>
<td>148</td>
</tr>
<tr>
<td>COMMAND FRAME ACKNOWLEDGE</td>
<td>148</td>
</tr>
<tr>
<td>DATA FRAME</td>
<td>149</td>
</tr>
<tr>
<td>OPERATION COMPLETE</td>
<td>149</td>
</tr>
<tr>
<td>Bus Timing</td>
<td>150</td>
</tr>
<tr>
<td>Handler Environment</td>
<td>152</td>
</tr>
<tr>
<td>Bootable Handler</td>
<td>153</td>
</tr>
<tr>
<td>Cartridge Resident Handler</td>
<td>153</td>
</tr>
<tr>
<td>Flowcharts</td>
<td>153</td>
</tr>
<tr>
<td>10 PROGRAM ENVIRONMENT AND INITIALIZATION</td>
<td>157</td>
</tr>
<tr>
<td>Cartridge</td>
<td>157</td>
</tr>
<tr>
<td>Cartridge Without Booted Support Package</td>
<td>158</td>
</tr>
<tr>
<td>Cartridge With Booted Support Package</td>
<td>158</td>
</tr>
<tr>
<td>Diskette-Booted Software</td>
<td>159</td>
</tr>
<tr>
<td>Diskette-Boot File Format</td>
<td>159</td>
</tr>
<tr>
<td>Diskette-Boot Process</td>
<td>160</td>
</tr>
<tr>
<td>Sample Diskette-Bootable Program Listing</td>
<td>161</td>
</tr>
<tr>
<td>Program to Create Diskette-Boot Files</td>
<td>162</td>
</tr>
<tr>
<td>Cassette-Booted Software</td>
<td>164</td>
</tr>
<tr>
<td>Cassette-Boot File Format</td>
<td>165</td>
</tr>
<tr>
<td>Cassette-Boot Process</td>
<td>165</td>
</tr>
<tr>
<td>Sample Cassette-Bootable Program Listing</td>
<td>167</td>
</tr>
<tr>
<td>Program to Create Cassette-Boot Files</td>
<td>168</td>
</tr>
</tbody>
</table>
11 ADVANCED TECHNIQUES AND APPLICATION NOTES

Sound Generation

Capabilities 170
Conflicts With OS 170

Screen Graphics 171

Hardware Capabilities 171
OS Capabilities 171
Cursor Control 171
Color Control 171
Alternate Character Sets 172

Player/Missile Graphics 174

Hardware Capabilities 174
Conflicts With OS 174

Reading Game Controllers 174

Keyboard Controller Sensing 174
Front Panel Connectors as I/O Ports 176

Hardware Information: 176
Software Information: 177
Other Miscellaneous Software Information: 179
APPENDICES

Appendix A -- CIO COMMAND BYTE VALUES 180
Appendix B -- CIO STATUS BYTE VALUES 181
Appendix C -- SIO STATUS BYTE VALUES 182
Appendix D -- ATASCII CODES 183
Appendix E -- DISPLAY CODES (ATASCII) 184
Appendix F -- KEYBOARD CODES (ATASCII) 185
Appendix G -- PRINTER CODES (ATASCII) 186
Appendix H -- SCREEN MODE CHARACTERISTICS 188
Appendix I -- SERIAL BUS ID AND COMMAND SUMMARY 191
Appendix J -- ROM VECTORS 192
Appendix K -- DEVICE CHARACTERISTICS 194
  Keyboard 194
  Display 194
  ATARI 410(TM) Program Recorder 194
  ATARI 820(TM) 40-Column Impact Printer 195
  ATARI 810(TM) Disk Drive 197

Appendix L -- OS DATA BASE VARIABLE
  FUNCTIONAL DESCRIPTIONS 200
  Central Data Base Description 200

  FUNCTIONAL INDEX TO DATA BASE VARIABLE DESCRIPTIONS 201
  A. MEMORY CONFIGURATION 211
B. TEXT/GRAPHICS SCREEN

Cursor Control 212
Screen Margins 213
Text Scrolling 215
Attract Mode 215
Tabbing 216
Logical Text Lines 217
Split Screen 218
Displaying Control Characters 220
Escape (Display Following Control Character) 221
Display Control Characters Mode 221
Bit-Mapped Graphics 221
Internal Working Variables 222
Internal Character Code Conversion 224

C. DISKETTE HANDLER 225

D. CASSETTE 225

Baud Rate Determination 226
Cassette Mode 227
Cassette Buffer 227
Internal Working Variables 228

E. KEYBOARD 229

Key Reading and Debouncing 229
Special Functions 230
Start/Stop 230
Autorepeat 231
Inverse Video Control 232
Console Keys: [SELECT], [START], and [OPTION] 232

F. PRINTER 232

Printer-Buffer 233
Internal Working Variables 233
G. CENTRAL I/O ROUTINE (CIO) 233

User Call Parameters 233
I/O Control Block 233
Device Status 234
Device Table 235
CIO/Handler Interface Parameters 235
Zero-Page IOCB 235
Internal Working Variables 236

H. SERIAL I/O ROUTINE (SIO) 237

User Call Parameters 237
Device Control Block 237
Bus Sound Control 238
Serial Bus Control 238
Retry Logic 238
Checksum 239
Data Buffering 240
General Buffer Control 240
Command Frame Output Buffer 240
Receive/Transmit Data Buffering 241
SIO Timeout 241
Internal Working Variables 242

J. ATARI CONTROLLERS 243

Joysticks 243
Paddles 244
Light Pen 245
Driving Controllers 246

K. DISK FILE MANAGER 247

L. DISK UTILITY POINTER 248

M. FLOATING POINT PACKAGE 248

N. Power-Up and System Reset 249

RAM Sizing 249
Diskette/Cassette-Boot 250
Environment Control 251
P. INTERRUPTS
System Timers
Real Time Clock
System Timer 1
System Timer 2
System Timers 3, 4 and 5
RAM Interrupt Vectors
NMI Interrupt Vectors
IRQ Interrupt Vectors
Hardware Register Updates
Internal Working Variables

R. USER AREAS
Alphabetical List of Data Base Variables
Memory Address Ordered List of Data Base Variables
Floating Point Package Variables

INDEX
# TABLE OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Illustration Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>ATARI Home Computer Block Diagram</td>
<td>19</td>
</tr>
<tr>
<td>1-2</td>
<td>Memory Layout Chart</td>
<td>20</td>
</tr>
<tr>
<td>4-1</td>
<td>6502 System Memory Map</td>
<td>29</td>
</tr>
<tr>
<td>4-2</td>
<td>Mapped I/O</td>
<td>32</td>
</tr>
<tr>
<td>5-1</td>
<td>I/O Subsystem Structure Flow Diagram</td>
<td>35</td>
</tr>
<tr>
<td>5-2</td>
<td>CIO Calling Mechanism</td>
<td>38</td>
</tr>
<tr>
<td>5-3</td>
<td>An I/O Example</td>
<td>49</td>
</tr>
<tr>
<td>5-4</td>
<td>Keycode to ATASCII Conversion Table</td>
<td>53</td>
</tr>
<tr>
<td>5-5</td>
<td>Text Modes 1 and 2 Data Form</td>
<td>56</td>
</tr>
<tr>
<td>5-6</td>
<td>Graphics Modes 3-11 GET Data Form</td>
<td>58</td>
</tr>
<tr>
<td>5-7</td>
<td>Graphics Modes 3-11 PUT Data Form</td>
<td>59</td>
</tr>
<tr>
<td>5-8</td>
<td>Screen Display Block Diagram</td>
<td>64</td>
</tr>
<tr>
<td>5-9</td>
<td>Cassette Handler Record Format</td>
<td>74</td>
</tr>
<tr>
<td>5-10</td>
<td>Device/Filename Syntax</td>
<td>81</td>
</tr>
<tr>
<td>5-11</td>
<td>File Management Subsystem Diskette Sector</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>Utilization Map</td>
<td></td>
</tr>
<tr>
<td>5-12</td>
<td>File Management Subsystem Boot Record Format</td>
<td>90</td>
</tr>
<tr>
<td>5-13</td>
<td>File Management Subsystem Boot Process</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>Memory Map</td>
<td></td>
</tr>
<tr>
<td>5-14</td>
<td>File Management Subsystem Volume Table</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>of Contents</td>
<td></td>
</tr>
<tr>
<td>5-15</td>
<td>File Management Subsystem Volume Bit Map</td>
<td>93</td>
</tr>
<tr>
<td>5-16</td>
<td>File Directory Format</td>
<td>94</td>
</tr>
<tr>
<td>5-17</td>
<td>File Management Subsystem File Sector Format</td>
<td>95</td>
</tr>
<tr>
<td>5-18</td>
<td>Resident Device Handler Vectors</td>
<td>96</td>
</tr>
<tr>
<td>5-19</td>
<td>DVSTAT 4-Byte Operation Status Format</td>
<td>100</td>
</tr>
</tbody>
</table>
PREFACE

This manual describes the resident Operating System (OS) for the ATARI® Home Computer, for readers who are familiar with the internal behavior of the system. It discusses:

- System functions and utilization techniques
- Subsystem relationships and organization
- Characteristics of the ATARI peripheral devices that can be attached to the ATARI 400® and ATARI 800® Home Computer
- Advanced techniques for going beyond the basic OS capabilities
- The general features of the computer system hardware used by the OS.

It would be helpful to have a familiarity with programming concepts and terminology, assembly language programming in general, the Synertek 6502 in particular, and digital hardware concepts and terminology. You will be provided with the information you need to use the OS resources, without resorting to trial-and-error techniques or the OS listing. Supporting information for tasks that involve OS listing references is also provided.

This manual does not present a comprehensive description of the hardware used to provide OS capabilities. The programmer who needs to go beyond the capabilities described should consult the ATARI Home Computer Hardware Manual.
1 INTRODUCTION

GENERAL DESCRIPTION OF THE ATARI HOME COMPUTER SYSTEM

Operating systems in the ATARI® 400[TM] and ATARI 800[TM] Home Computer are identical. The primary differences between the two are:

- Physical packaging
- The ATARI 400 Computer console has one cartridge slot, the ATARI 800 Computer console has two cartridge slots
- The ATARI 400 Home Computer contains 16K RAM and cannot be expanded. The ATARI 800 Home Computer can be expanded to a maximum of 48K RAM.
- The ATARI 800 Computer has a monitor jack; the ATARI 400 Computer does not.

The Hardware Circuitry

- Produces both character and point graphics for black and white (B/W) or color television.
- Produces four independent audio channels (frequency controlled) which use the television sound system.
- Provides one bi-level audio output in the base unit.
- Interfaces with up to four Joysticks and eight Paddle Controllers.
- Interfaces with a serial I/O bus for expansion.
- Contains a built-in keyboard

Figure 1-1 presents a simplified block diagram of the hardware. See the hardware manual for supporting documentation.
Figure 1-1. ATARI Home Computer Block Diagram
CONVENTIONS USED IN THIS MANUAL

This manual uses the following special notations:

Hexadecimal Numbers

All two-digit numbers preceded by a dollar sign ($) designate hexadecimal numbers. All other numbers (except memory addresses) are in decimal form unless otherwise specified in the supporting text.

Memory Addresses

All references to computer memory and mapped I/O locations are in hexadecimal notation. Memory addresses may or may not be contained in square brackets. (Example: [D20F] and D20F are the same address.)

Kilobytes of Memory

Memory sizes are frequently expressed in units of kilobytes, such as 32K, where a kilobyte is 1024 bytes of memory.

PASCAL As an Algorithm-Specification Language

The PASCAL language (procedure block only) is used as the specification language in the few places where an algorithm is specified in detail. PASCAL syntax is similar to any number of other block-structured languages, and you should have no difficulty following the code presented.

Memory Layouts

Diagrams similar to Figure 1-2 are used whenever pictures of bytes or tables are presented:

```
7 6 5 4 3 2 1 0
+-----------+--- This is a single byte.
+-----------+
|           |
+-----------+--- This is a word (2 bytes).
|           |
+-----------+--- This is a block of memory of unspecified length.
|           |
+-----------+
```

Figure 1-2. Memory Layout Chart
Bit 7 is the most significant bit (MSB) of the byte, and Bit 0 is the least significant bit (LSB).

In tables and figures, memory addresses always increase toward the bottom of the figure.

Backus–Naur Form

A modified version of Backus–Naur Form (BNF) is used to express some syntactic forms, where the following metalinguistic symbols are used:

::= is the substitution (assignment) operator.
<> a metasyntactic variable.
| separates alternative substitutions.
[ ] an optional construct.

Anything else is a syntactic literal constant, which stands for itself.

For Example:

<device specification> ::= <device name> [ <device number> ]:
<device name> ::= C|D|E|K|P|R|S
<device number> ::= 1|2|3|4|5|6|7|8

A "device specification" consists of a mandatory "device name," followed by an optional "device number," followed by the mandatory colon character. The device name in turn must be one of the characters shown as alternatives. The device number (if it is present) must be a digit 1 through 8.

OS Equate Filenames

Operating System ROM (Read Only Memory) and RAM (Random Access Memory) vector names, RAM database variable names and hardware register names are all referred to by the names assigned in the OS program equate list. When one of these names is used, the memory address is usually provided, such as BOOTAD [0242].
This section describes the various subsystems of the resident OS in general terms.

Input/Output Subsystem

The Input/Output (I/O) subsystem provides a high-level interface between the programs and the hardware. Most functions are device-independent, such as the reading and writing of character data, yet provisions have been made for device-dependent functions as well. All peripheral devices capable of dealing with character data have individual symbolic names (such as K, D, P, etc.) and can be accessed using a Central I/O (CIO) routine.

A RAM data base provides access to controllers (joysticks and paddle controllers), which do not deal with character data. This RAM data base is periodically updated to show the states of these devices.

Interrupt Processing

The interrupt system handles all hardware interrupts in a common and consistent manner. By default, all interrupts are fielded by the OS. At your discretion, individual interrupts (or groups of interrupts) can be fielded by the application program.

Initialization

The system provides two levels of initialization: power up and system reset. The OS performs power-up initialization each time the system power is switched to ON, and system reset initialization is performed each time the [SYSTEM. RESET] key is pressed.

Power-Up

The OS examines and notes the configuration of the unit whenever the system power is switched to ON. The system performs the following tasks at power up:
o Determines the highest RAM address.
o Clears all of RAM to zeros.
o Establishes all RAM interrupt vectors.
o Formats the device table.
o Initializes the cartridge(s).
o Sets up the screen for 24 x 40 text mode.
o Boots the cassette if directed.
o Checks cartridge slot(s) for diskette-boot instructions.
o Boots the diskette if directed to do so and a disk drive unit is attached.
o Transfers control to the cartridge, diskette-booted program, cassette-booted program, or blackboard program.

[System. Reset]

Pressing the [SYSTEM. RESET] key causes the OS to perform these following tasks:
o Clears the OS portion of RAM.
o Rechecks top of RAM.
o Reestablishes all RAM interrupt vectors.
o Formats the device table.
o Initializes the cartridge(s).
o Sets up the screen for 24 x 40 text mode.
o Transfers control to the cartridge, a diskette-booted program, a cassette-booted program, or the blackboard program.

Note that [SYSTEM. RESET] does not perform all the power-up tasks listed in the power-up section.
FLOATING POINT ARITHMETIC PACKAGE

The OS ROM contains a Floating Point (FP) package that is available to nonresident programs such as ATARI BASIC. The package is not used by the other parts of the OS itself. The floating point numbers are stored as 10 BCD digits of mantissa, plus a 1-byte exponent. The package contains these routines:

- ASCII-to-FP and FP-to-ASCII conversion.
- Integer-to-FP and FP-to-integer conversion.
- FP add, subtract, multiply and divide.
- FP log, exp, and polynomial evaluation.
- FP number clear, load, store, and move.
3 CONFIGURATIONS

The ATARI 400 and ATARI 800 Home Computers support a wide variety of configurations, each with a unique operating environment:

- Cartridge(s) may or may not be inserted
- Memory can be optionally added to the ATARI 800 Computer console in 16K increments
- Many different peripheral devices can be attached to the serial I/O bus.

The OS accounts for all of these variables without requiring a change in the resident OS itself (see Section 2). The machine configuration is checked when power is first turned on and then is not checked again, unless system reset is used. A general discussion of some of the valid configurations follows.

PROGRAM ENVIRONMENTS

The OS allows one of four program types to be in control at any point in time:

- The OS blackboard (ATARI Memo Pad) program
- A cartridge-resident program
- A diskette-booted program
- A cassette-booted program

Control choice is based upon information in the cartridge(s), upon whether or not a disk drive is attached, and upon operator keyboard inputs. The exact algorithms are discussed in detail in Section 7.

Blackboard Mode

In blackboard mode, the screen is established as a 24 x 40 text screen. Anything entered from the keyboard goes to the screen without being examined, although all of the screen editing functions are supported. Blackboard mode is the lowest priority environment. You go there only by command from a higher
priority environment, or by default, if there is no other reasonable environment for the OS to enter. For example, typing BYE in BASIC causes the OS to enter the blackboard mode. The blackboard mode can be exited by pressing the [SYSTEM.RESET] key if it was entered from a higher environment.

Cartridge

An inserted cartridge normally provides the main control after initialization is complete (for example: ATARI BASIC, SUPER BREAKOUT[TM], BASKETBALL, COMPUTER CHESS, and others. All these cartridge programs interface directly with you in some way). Although a cartridge can provide a supporting function for some other program environment, this has not yet been done. Some cartridges (particularly keyboard-oriented ones) can change environments by entering special commands (such as "BYE") to go to blackboard mode or "DDS" to enter the disk utility. Other cartridges cannot change environments. Note that a hardware interlock prevents the removal or insertion of a cartridge with the power on; this feature causes the entire system to reinitialize with every cartridge change.

Diskette Boot

The diskette may or may not be booted when the system powers up with diskette-bootable software. This paragraph assumes that a diskette boot did occur. See Section 7 for boot condition explanations.

The diskette-booted software can take control as the Disk Utility Program (DUP) does under certain conditions, or can provide a supporting function as the File Management System (FMS) does. This environment is so flexible that it is difficult to generalize on its capabilities and restrictions. The only machine requirement (other than the disk drive) is that sufficient RAM be installed to support the program being booted.

Cassette-Boot

The cassette-boot environment is similar to the diskette-boot environment, although the cassette is limited as an I/O device. It is slow and can access only one file at a time in sequence. Note that the cassette-boot facility has no relation to the use of cassettes to store high-level language programs (e.g., programs written in ATARI BASIC), nor to the use of cassettes to store data.
RAM EXPANSION

Although you can expand RAM noncontiguously in the ATARI 800 Home Computer, the OS will only recognize RAM that is contiguous starting from location 0. Installation directions are provided with the purchased RAM modules. RAM can be added until it totals 48K. After 32K, additional RAM overlays first the right-cartridge addresses (32K to 40K) and then the left-cartridge addresses (40K to 48K). Note that in cases of conflict, the inserted cartridge has higher priority and disables the conflicting RAM in 8K increments. See Section 4 for a detailed discussion of system memory.

As a result of power-up, the OS will generate two pointers that define the lowest available RAM location and the highest available RAM location. The OS and diskette or cassette-booted software will determine the location of the lowest available RAM, while the number of RAM modules and the current screen mode will determine the highest available RAM.

PERIPHERAL DEVICES

Peripheral devices of several types can be added to the system using standard cables to either the serial bus or the connectors at the front of the computer console. The most common types deal with either transmission of bytes of data (usually serial bus) or transmission of sense information (usually game controllers).

Game Controllers

The OS periodically senses (50 or 60 times per second) the standard game controllers (Paddles and Joysticks) and the values read are stored in RAM. You can plug in, remove, and rearrange these controllers at will without affecting system operation, because the system will always try to read all of these controllers. The Driving Controllers are read, but not decoded, by the OS. Special instructions are required to read the keyboard controller (see Section 11).

Program Recorder

The ATARI 410ITM1 Program Recorder is a special peripheral. It uses the serial bus to send and receive data, but does not conform to the protocol of the other peripherals that use the serial bus. The Program Recorder must also be the last device on the serial bus, because it does not have a serial bus extender connector as the other peripherals do. There can never be more than one Program Recorder connected to any system for the same reason. The system cannot sense the presence or absence of the Program Recorder, so it can be connected and disconnected at will.

OPERATING SYSTEM CO16555 -- Section 3
Serial Bus Devices

A serial bus device conforms to the serial I/O bus protocol as defined in Section 9, but this does not include the Program Recorder. Each serial bus device has two identical connectors: a serial bus input, and a serial bus extender. Either connector can be used for either purpose. Peripherals can be "daisychained" by cabling them together in a sequential fashion. There are usually no restrictions on the cabling order because each device has a unique identifier. Where restrictions exist, they will be mentioned in Section 5.
4 SYSTEM MEMORY UTILIZATION

Memory in the system is decoded in the full 64K range of the 6502 microcomputer and there are no provisions for additional mapping to extend memory. Memory is divided into four basic regions (with some overlap possible): RAM, cartridge area, I/O region and the resident OS ROM. The regions and their address boundaries are listed below (all addresses are in hexadecimal):

0000-1FFF = RAM (minimum required for operation)
2000-7FFF = RAM expansion area
8000-9FFF = Cartridge B, Cartridge A (half of 16K size) or RAM
A000-BFFF = Cartridge A or RAM
C000-CFFF = Unused
D000-D7FF = Hardware I/O decodes
D800-DFFF = Floating Point Package (OS)
E000=FFFF = Resident Operating System ROM

Figure 4-1 6502 System Memory Map

This section will break these regions into even smaller functional divisions and provide detailed explanations of their usage.

RAM REGION

The OS and the control program share the RAM region. The RAM region can be further subdivided into the following sub regions for discussion purposes:

Page 0 = 6502 page zero address mode region.
Page 1 = 6502 stack region.
Pages 2-4 = OS database and user workspace.
Pages 5-6 = User program workspace.
Pages 7-XX = Bootable software area/free RAM.*
Pages XX-top of RAM = Screen display list and data.*

* Note that XX is a function of the screen graphics mode and the amount of RAM installed.

The paragraphs that follow describe how the OS uses RAM subregions, and presents user program recommendations.
The architecture of the 6502 microcomputer instruction set and addressing modes gives page 0 special significance. References to addresses in that page (0000 to 0OFF) are faster, require fewer instruction bytes, and provide the only mechanism for hardware indirect addressing. Page 0 should be used sparingly so that all possible users can have a portion of it. The OS permanently takes the lower half of page 0 (0000 to 007F). This portion can never be used by any outer environment unless the OS is completely disabled and all interrupts to the OS are eliminated.

The upper half of page 0 (0080 to 0OFF) is available to outer environments with the following restriction: the floating point package, if used, requires 00D4 through 00FF.

Page 1

Page 1 is the 6502 hardware stack region; JSR instructions, PHA instructions, and interrupts all cause data bytes to be written to page 1. Conversely RTS, PLA, and RTI instructions all cause data bytes to be read from page 1. The 256-byte stack is adequate for normal subroutine calls plus interrupt process nesting, so no restrictions have been made on page 1 usage. It is obvious that a stack of this size is totally inadequate for deeply recursive processes or for nested processes with large local environments to be saved. So, for sophisticated applications, software maintained stacks must be implemented.

The 6502 stack pointer is initialized at power-up or system reset to point to location 0FFF. The stack then pushes downward toward 0100. The stack will wrap around from 0100 to 01FF if a stack overflow condition occurs, because of the nature of the 6502’s 8-bit stack pointer register.

OS Data Base

Locations 0200 through 047F are allocated by the OS for working variables, tables and data buffers. Portions of this region can be used only after you determine that nonconflict with the OS is guaranteed. For example, the printer and cassette buffers could be used if I/O operations to these devices are impossible within the controlling environment. The amount of work involved in determining nonconflict seems to be completely out of line with the benefits to be gained (except for a few trivial cases) and it is recommended that pages 2 through 4 not be used except by the OS.
User Workspace

Locations 0480 through 06FF are dedicated for outer environment use except when the floating point package is used. The floating point package uses locations 057E through 05FF.

Boot Region

Page 7 is the start of the "boot region." When software is booted from either the diskette or the cassette, it can start at the lowest free memory address (that is 0700) and proceed upward (although it can also start at any address above 0700 and below the screen display list). The top of this region defines the start of the "free memory" region. When the boot process is complete, a pointer in the data base contains the address of the next available location above the software just booted. When no software has been booted, this pointer contains the value 0700.

Screen Display List and Data

When the OS is handling the screen display, the display list that defines the screen characteristics and the current data that is contained on the screen are placed at the high address end of RAM. The bottom of this region defines the end of the free memory region and its location is a function of the screen mode currently in effect. A pointer in the data base contains the address of the last available location below the screen region.

Free Memory Region

The free memory region is all the RAM between the end of the boot region and the start of the screen region. The outer level application is responsible for managing the free memory region.

CARTRIDGES A AND B

There are two 8K regions reserved for plug-in cartridges. Cartridge B, that is the right-hand cartridge slot found only in the ATARI 800 Home Computer, has been allocated memory addresses 8000 through 9FFF. Cartridge A (the left-hand cartridge slot in the ATARI 800 Computer console, and the only slot in the ATARI 400 Computer console) has been allocated memory addresses A000 through BFFF and optionally 8000 through BFFF, for 16K cartridges. If a RAM module is plugged into the last slot such as to overlay any of these addresses, the RAM takes precedence as long as a cartridge is not inserted. However, if a cartridge is inserted, it will disable the entire conflicting RAM module in the last slot in 8K increments.
MAPPED I/O

The 6502 performs input/output operations by addressing the external support chips as memory; some chip registers are read/write while others are read-only or write-only (the ATARI Home Computer Hardware Manual gives descriptions of all of the external registers). While the entire address space from D000 to D7FF has been allocated for I/O decoding, only the following subregions are used:

D000-D01F = CTIA
D200-D21F = POKEY
D300-D31F = PIA
D400-D41F = ANTIC

Figure 4-2. Mapped I/O

RESIDENT OS AND FLOATING POINT PACKAGE ROM

The region from D800 through FFFF always contains the OS and the floating point package. Care should be taken to avoid using any entry points that are not guaranteed not to move, to allow for the possibility that another, but functionally compatible, OS can be generated in the future. The OS contains many vectored entry points at the end of the ROM and in RAM that will not move. The floating point package is not vectored, but all documented entry points will be fixed: Do not use undocumented routines found by scanning the listing. A list of the fixed ROM vectors can be found in Appendix J.

CENTRAL DATA BASE DESCRIPTION

See Appendix L.

MEMORY DYNAMICS

The free memory region is the area between the end of the boot region and the start of the screen region. As such, its limits are variable. MEMLO [02E7] defines the bottom of the free region, and MEMTOP [02E5] defines the top of the region. This section presents the conditions that cause the setup or alteration of these variables.
System Initialization Process

The OS determines the extent of the lowest block of contiguous RAM, and saves the limits. The Screen Editor is then opened, thus setting a new (and lower) value in MEMTOP. Diskette or cassette-booted software might be brought into memory, that would probably set a new (and higher) value in MEMLO (see Section 7). MEMLO and MEMTOP will define the maximum amount of free memory available when the application program finally gets control. That amount of free memory can later decrease, as described in the next paragraph.

Changing Screen Modes

The Display Handler interprets the variable APPMHI 'O00E' to contain the address below which MEMTOP cannot extend. This allows you to protect the portion of free memory space that you are using from being overwritten as a result of screen mode change. The display handler will set the screen for mode O, update MEMTOP, and return an error status to you, if it determines that the screen memory will extend below APPMHI as a result of a screen mode change. In other cases the Display Handler effects the desired mode change and updates MEMTOP.
5 I/O SUBSYSTEM

This section discusses the I/O subsystem of the Operating System. The I/O subsystem comprises a collection of routines that allow you to access peripheral and local devices at three different levels. The CIO (Central I/O Utility), provides the highest level, device independent access to devices. The second level allows communication with the device handlers. The lowest level is the SIO (Serial I/O bus Utility) routine. Any lower level access to a device involves the direct reading and writing of the hardware registers associated with the device.

The data byte is the basic unit of input/output. A data byte can contain either "binary" (non text) information, or encoded text information. The text encoding scheme supported by the OS is called ATASCII, derived from the words "ATARI ASCII." Most ATASCII codes are the same as ASCII, with the primary deviations being the control codes. Appendix D shows the ATASCII character set, and Appendices E, F, and G show device-specific implementations for the display, keyboard, and printer.

The structure of the I/O subsystem is shown on the following page.
Where: ---- shows a control path. **** shows the data structure required for a path.

Note the following:

- The Keyboard/Display/Screen Editor Handlers don’t use SIO.
- The Diskette handler cannot be called directly from CIO.
- The DCB is shown twice in the diagram.

Figure 5-1  I/O Subsystem Structure Flow Diagram
CENTRAL I/O UTILITY

The Central I/O Utility provides you with a single interface in which to access all of the system peripheral devices in a device-independent manner. The minimum unit of data transfer is the data byte. The CIO also supports multiple byte transfers. All I/O operations are performed on a "return-to-user-when-complete" basis; there is no way to initiate concurrent "overlapped" I/O processes.

I/O is organized by "files," where a file is a sequential collection of data bytes. A file can or may not contain textual data and it can or may not be organized by "records," where a record is a contiguous group of bytes terminated by an EOL (End of Line) character. Some files are synonymous with a device (as with the printer and the Screen Editor), while other devices can contain multiple files, each with a unique name (as with the disk drive).

CIO allows you to access up to eight independent device/files at one time, because there are eight I/O Control Blocks (IOCB's) in the system. Each of the IOCB's can be assigned to control any device/file because there are no preferred assignments, except that IOCB #0 is assigned to the Screen Editor at power-up and system reset.

To access a peripheral, you first set up an IOCB for the OPEN command, that supplies the system name for the device to be accessed (e.g. K:, for the keyboard, P:, for the printer, D:STARS for a diskette file named 'STARS', etc.). You then call the CIO, telling it to examine the IOCB to find the OPEN information. CIO attempts to find the specified device/file and returns a status byte indicating the success of the search. If the specified device/file can be found by CIO, then CIO stores control information in the IOCB. The IOCB is now used for as long as that file is open.

Once a file is open, it can then be accessed using data-read or data-write types of commands; in general, reading can proceed until there is no more data to read (End of File) and writing can proceed until there is no more medium to store data on (End of Medium), although neither reading nor writing need proceed to that point. The reading and writing of data generally occurs into and out of user-supplied data buffers (although a special case allowing single byte transfers using the 6502 A register is provided).

When there are no more accesses to be performed on an open device/file, you perform the close operation. This accomplishes two functions:

- It terminates and makes permanent an output file (essential for diskette and cassette).

- It releases that IOCB to be used for another I/O operation.
CIO Design Philosophy

The CIO utility was designed specifically to meet the following design criteria.

- The transfer of data is device independent.
- Byte-at-a-time, multiple byte and record-aligned accesses are supported.
- Multiple device/files can be accessed concurrently.
- Error handling is largely device independent.
- New device handlers can be added without altering the system ROM.

Device Independence

CIO provides device independence by having a single entry point for all devices (and for all operations) and by having a device-independent calling sequence. Once a device/file is opened, data transfers occur with no regard to the actual device involved. Uniform rules for handling byte- and record-oriented data transfers allow the actual device storage block sizes to be transparent to you.

Data Access Methods

The CIO supports two file access methods: byte-aligned and record-aligned.

Byte-aligned accesses allow you to treat the device/file as a sequential byte stream; any number of bytes can be read or written and the following operation will continue where the prior one left off. Records are of no consequence in this mode, and reads or writes can encompass multiple records if desired.

Record-aligned accesses allow you to deal with the data stream at a higher level, that of the data record or "line of text." Each and every write operation creates a single record (by definition). Each read operation assures that the following read operation will start at the beginning of a record. Record-aligned accesses cannot deal with portions of more than one record at a time. Record-aligned accesses are useful only with text data or with binary data guaranteed not to contain the EOL character ($9B) as data.

Note that any file can be accessed using the byte-aligned access method, regardless of how the file was created. But not all files can be successfully read using record-aligned accesses; the file
must contain EOL characters at the end of each record and at no other place.

Multiple Device/File Concurrency

Up to eight device/files can be accessed concurrently using CIO, each operating independently of the others.

Unified Error Handling

All error detection and recovery occurs within the CIO subsystem. The status information that reaches you is in the form of a status byte for each device/file. Error codes are device independent as much as possible (see Appendix B).

Device Expansion

Devices are known by single character names such as K or P, and a number of device handlers are part of the resident system ROM. However, additional device handlers can be added to the system using the RAM-resident device table; this is normally done at power-up time as with the diskette boot process, but can be done at any point in time.

CIO Calling Mechanism

The input/output control block (IOCB) is the primary parameter passing structure between you and CIO. There are eight IOCB's in the system, arranged linearly in RAM as shown below:

```
+-------+  low address [0340]
| IOCB 0 |
+-------+
| IOCB 1 |
+-------+
   =
+-------+
| IOCB 6 |
+-------+
| IOCB 7 |
+-------+  high address
```

Figure 5-2  CIO Calling Mechanism
One IOC B is required for each open device/file. Any IOC B can be used to control any device/file, although IOC B 0 is normally assigned to the Screen Editor (E:). You perform a typical I/O operation by:

- Inserting appropriate parameters into an IOC B of your choosing
- Putting the IOC B number times 16 into the 6502 X register
- Performing a JSR to the CIO entry point CIOV [E456].

CIO returns to you when the operation is complete or if an error was encountered. The operation status is in the IOC B used, as well as in the 6502 Y register. The 6502 condition codes will also reflect the value in the Y register. In some cases a data byte will be in the 6502 A register. The X register will remain unchanged for all operations and conditions. An example is shown below:

```
IOCB2X = $20       ; INDEX FOR IOC B #2.
LDX #IOCB2X
JSR CIOV
CPY #0            ;(optional)
BMI ERROR
```

This sector describes each IOC B byte, with its file name and address. Each IOC B is 16 bytes long. Some bytes can be altered by you and some are reserved for use by CIO and/or the device handlers.

**Handler ID -- ICHID [0340]**

The handler ID is an index into the system device table (see Section 9) and is not user-alterable. This byte is set by CIO as the result of an OPEN command and is left unchanged until the device/file is closed, at that time CIO will set the byte to $FF.

**Device Number -- ICDNO [0341]**

The device number is provided by CIO as the result of an OPEN command and is not user-alterable. This byte is used to distinguish between multiple devices of the same type, such as D1: and D2:.
Command Byte -- ICCMD [0342]

You set the command byte. It specifies the command to be performed by the CIO. This byte is not altered by CIO.

Status -- ICSTA [0343]

The CIO conveys operation status to you with the command status byte as a result of each and every CIO call. Each and every CIO call updates the command status byte. The most significant (sign) bit is a one for error conditions and zero for non-error conditions, and the remaining bits represent an error number. See Appendix B for a list of status codes.

Buffer Address -- ICBAL [0344] and ICBAH [0345]

You set this 2-byte pointer; it is not altered by CIO. The pointer contains the address of the beginning (low address) of a buffer that:

- Contains data for read and write operations
- Contains the device/filename specification for the OPEN command.

You can alter the pointer at any time.

PUT Address -- ICPTL [0346] and ICPTH [0347]

The CIO sets this 2-byte pointer at OPEN time to the handler's PUT CHARACTER entry point (-1). The pointer was provided to accommodate the people writing the ATARI BASIC cartridge, and has no legitimate use in the system. This variable is set to point to CIO's "IOCB not OPEN" routine on CLOSE, Power-up and [SYSTEM.RESET].

Buffer Length/Byte Count -- ICBLL [0348] and ICBLH [0349]

You set this 2-byte count to indicate the size of the data buffer pointed to by ICBAL and ICBAH for read and write operations. It is not required for OPEN. After each read or write operation, CIO will set this parameter to the number of bytes actually transferred into or out of the data buffer. For record-aligned access, the record length can well be less than the buffer length. Also an end of file condition or an error can cause the byte count to be less than the buffer length.

Auxiliary Information -- ICAX1 [034A] and ICAX2 [034B]

OPERATING SYSTEM CD16555 -- Section 5
You set these 2-bytes. They contain information that is used by the OPEN command process and/or is device-dependent.

For OPEN, two bits of ICAX1 are always used to specify the OPEN direction as shown below, where R is set to 1 for input (read) enable and W is set to 1 for output (write) enable.

```
7 3 2 0
+++++++-++++++
| | | |WIR| | |
+++++++-++++++
```

ICAX1 is not altered by CIO. You should not alter ICAX1 once the device/file is open.

The remaining bits of ICAX1 and all of ICAX2 contain only device-dependent data and are explained later in this section.

Remaining Bytes (ICAX3-ICAX6)

The handler reserves the four remaining bytes for processing the I/O command for CIO. There is no fixed use for these bytes. They are not user-alterable except as specified by the particular device descriptions. These bytes will be referred to as ICAX3, ICAX4, ICAX5 and ICAX6, although there are no equates for those names in the OS equate file.

CIO Functions

The CIO supports records and blocks and the handlers support single bytes. All of the system handlers support one or more of the eight basic functions subject to restrictions based upon the direction of data transfer (e.g. one cannot read data from the printer). The basic functions are: OPEN, CLOSE, GET CHARACTERS, PUT CHARACTERS, GET RECORD, PUT RECORD, GET STATUS, and SPECIAL.

OPEN -- Assign Device/Filename to IOCB and Ready for Access

A device/file must be opened before it can be accessed. This process links a specific IOCB to the appropriate device handler, initializes the device/file, initializes all CIO control variables, and passes device-specific options to the device handler.
You set up the following IOCB parameters prior to calling CIO for an OPEN operation:

COMMAND BYTE = $03

BUFFER ADDRESS = pointer to a device/filename specification.

AUX1 = OPEN direction bits, plus device-dependent information.

AUX2 = device-dependent information.

After an OPEN operation, CIO will have altered the following IOCB parameters:

HANDLER ID = index to the system device table; this is used only by CIO and must not be altered.

DEVICE NUMBER = device number taken from the device/filename specification and must not be altered.

STATUS = result of OPEN operation; see Appendix B for a list of the possible status codes. In general, a negative status will indicate a failure to open properly.

PUT ADDRESS = pointer to the PUT CHARACTERS routine for the device handler just opened.

It is recommended that this pointer not be used.

CLOSE -- Terminate Access to Device/File and Release IOCB.

You issue a CLOSE command after you are through accessing a given device/file. The CLOSE process completes any pending data writes, goes to the device handler for any device-specific actions, and then releases the IOCB.

You set the following IOCB parameter prior to calling CIO:

COMMAND BYTE = $0C

The CIO alters the following IOCB parameters as a result of the CLOSE operation:

HANDLER ID = $FF

STATUS = Result of CLOSE operation.

PUT ADDRESS = pointer to "IOCB not OPEN" routine.
GET CHARACTERS -- Read n Characters (Byte-Aligned Access)

The specified number of characters are read from the device/file to the user-supplied buffer. EOL characters have no termination features when using this function; there can be no EOL, or many EOL's, in the buffer after operation completion. There is a special case provided that passes a single byte of data in the 6502 A register when the buffer length is set to zero.

You set the following IOCB parameters prior to calling CIO:

COMMAND BYTE = $07

BUFFER ADDRESS = pointer to data buffer.

BUFFER LENGTH = number of bytes to read; if this is zero, the data will be returned in the 6502 A register only.

The CIO alters the following IOCB parameters as a result of the GET CHARACTERS operation:

STATUS = result of GET CHARACTERS operation.

BYTE COUNT/BUFFER LENGTH = number of bytes read to the buffer. The BYTE COUNT will always equal the BUFFER LENGTH except when an error or an end-of-file condition occurs.

PUT CHARACTERS -- Write n Characters (Byte-Aligned Access)

The specified number of characters are written from the user-supplied buffer to the device/file. EOL characters have no buffer terminating properties, although they have their standard meaning to the device/file receiving them; no EOL's are generated by CIO. There is a special case that allows a single character to be passed to CIO in the 6502 A register if the buffer length is zero.

You set the following IOCB parameters prior to initiating the PUT CHARACTERS operation:

COMMAND BYTE = $08

BUFFER ADDRESS = pointer to data buffer.

BUFFER LENGTH = number of bytes of data in buffer.

The CIO alters the following IOCB parameter as a result of the PUT CHARACTERS operation:

STATUS = result of PUT CHARACTERS operation.
GET RECORD -- Read Up To n Characters (Record-Aligned Access)

Characters are read from the device/file to the user-supplied buffer until either the buffer is full or an EOL character is read and put into the buffer. If the buffer fills before an EOL is read, then the CIO continues reading characters from the device/file until an EOL is read, and sets the status to indicate that a truncated record was read. No EOL will be put at the end of the buffer.

You set the following IOCB parameters prior to calling CIO:

- **COMMAND BYTE** = $05
- **BUFFER ADDRESS** = pointer to data buffer.
- **BUFFER LENGTH** = maximum number of bytes to read (including the EOL character).

The CIO alters the following IOCB parameters as a result of the GET RECORD operation:

- **STATUS** = result of GET RECORD operation.
- **BYTE COUNT/BUFFER LENGTH** = number of bytes read to data buffer; this can be less than the maximum buffer length.

PUT RECORD -- Write Up To n Characters (Record-Aligned Access)

Characters are written from the user-supplied buffer to the device/file until either the buffer is empty or an EOL character is written. If the buffer is emptied without writing an EOL character to the device/file, then CIO will send an EOL after the last user-supplied character.

You set the following IOCB parameters prior to calling CIO:

- **COMMAND BYTE** = $09
- **BUFFER ADDRESS** = pointer to data buffer.
- **BUFFER LENGTH** = maximum number of bytes in buffer.

The CIO alters the following IOCB parameter as a result of the PUT RECORD operation:

- **STATUS** = result of PUT RECORD operation.
GET STATUS -- Return Device-Dependent Status Bytes

The device controller is sent a STATUS command, and the controller returns four bytes of status information that are stored in DVSTAT [02EA].

You set the following IOCBI parameters prior to calling CIO:

```
COMMAND BYTE = $0D
```

```
BUFFER ADDRESS = pointer to a device/filename specification
if the IOCBI is not already OPEN; see the discussion of the
implied OPEN option below.
```

After a GET STATUS operation, CIa will have altered the following parameters:

```
STATUS = result of GET STATUS operation; see Appendix B for
a list of the possible status codes.
```

```
DVSTAT = the four-byte response from the device controller.
```

SPECIAL -- Special Function

Any command byte value greater than $0D is treated by CIa as a special case. Since CIa does not know what the function is, CIa transfers control to the device handler for complete processing of the operation.

The user sets the following IOCBI parameters prior to calling CIa:

```
COMMAND BYTE > $0D
```

```
BUFFER ADDRESS = pointer to a device/filename specification
if the IOCBI is not already open; see the discussion of the
implied OPEN option below.
```

Other IOCBI bytes can be set up, depending upon the specific SPECIAL command being performed.

After a SPECIAL operation, CIa will have altered the following parameters:

```
STATUS = result of SPECIAL operation; see Appendix B for a
list of the possible status codes.
```

Other bytes can be altered, depending upon the specific SPECIAL command.

OPERATING SYSTEM CO16555 -- Section 5
Implied OPEN Option

The GET STATUS and SPECIAL commands are treated specially by CI0; they can use an already open IOCB to initiate the process or they can use an unopened IOCB. If the IOCB is unopened, then the buffer address must contain a pointer to a device/filename specification, just as for the OPEN command; CI0 will then open that IOCB, perform the specified command and then close the IOCB again.

Device/Filename Specification

As part of the OPEN command, the IOCB buffer address parameter points to a device/filename specification, that is a string of ATASCII characters in the following format:

<specification> ::= <device>[<number>][<filename>]<eol>
<device> ::= C|D|E|K|P|R|S
<number> ::= 1|2|3|4|5|6|7|8
<filename> has device-dependent characteristics.
<eol> ::= $9B

The following devices are supported at this writing:

C = Cassette drive
D1 through D8 = Floppy diskette drives *
E = Screen Editor
K = Keyboard
P = 40-column printer
P2 = 80-column printer *
P1 through R4 = RS-232-C interfaces *
S = Screen display

Devices flagged by asterisks (*) are supported by nonresident handlers.

If <number> is not specified, it is assumed to be 1.

The following examples show valid device/filename specifications:

C: Cassette
D2:BDAT File "BDAT" on disk drive #2
D:HOLD File "HOLD" on disk drive #1
K: Keyboard
I/O Example

The example provided in this section illustrates a simple example of an I/O operation using the CIO routine.

This code segment illustrates the simple example of reading text lines (records) from a diskette file named TESTER on disk drive #1. All symbols used are equated within the program although many of the symbols are in the OS equate file.

The program performs the following steps:

1. Opens the file 'D1:TESTER' using IOCB #3.
2. Reads records until an error or EOF is reached.
3. Closes the file.

I/O EQUATES

EOL= $9B ; END OF LINE CHARACTER.
IOCB3= $30 ; IOCB #3 OFFSET (FROM IOCB #0).
ICHID= $0340 ; (HANDLER ID -- SET BY CIO).
ICDNO= ICHID+1 ; (DEVICE # -- SET BY CIO).
ICCOM= ICDNO+1 ; COMMAND BYTE.
ICSTA= ICCOM+1 ; STATUS BYTE -- SET BY CIO.
ICBAL= ICSTA+1 ; BUFFER ADDRESS (LOW).
ICBAH= ICBAL+1 ; BUFFER ADDRESS (HIGH).
ICPTL= ICBAH+1
ICPTH= ICPTL+1
ICBLH= ICPTH+1 ; BUFFER LENGTH (LOW).
ICBH= ICBLH+1 ; BUFFER LENGTH (HIGH).
ICAX1= ICBH+1 ; AUX 1.
ICAX2= ICAX1+1 ; AUX 2.
OPEN= $03 ; OPEN COMMAND.
GETREC= $05 ; GET RECORD COMMAND.
CLOSE= $0C ; CLOSE COMMAND.
OREAD= $04 ; OPEN DIRECTION = READ.
OWRIT= $08 ; OPEN DIRECTION = WRITE.
EOF= $88 ; END OF FILE STATUS VALUE.
CIOV= $E456 ; CIO ENTRY VECTOR ADDRESS.

FIRST INITIALIZE THE IOCB FOR FILE "OPEN".

LDX #IOCB3 ; SETUP TO ACCESS IOCB #3.
LDA  #OPEN               ; SETUP OPEN COMMAND.
STA  ICCOM, X

LDA  #NAME               ; SETUP BUFFER POINTER TO ...
STA  ICBAL, X            ; ... POINT TO FILENAME.

LDA  #NAME/256           ; SETUP FOR OPEN READ.
STA  ICBAH, X

LDA  #OREAD              ; SETUP BUFFER POINTER TO ...
STA  ICAX1, X

LDA  #0                  ; CLEAR AUX 2.
STA  ICAX2, X

"OPEN" THE FILE.

JSR  CIOV                ; PERFORM "OPEN" OPERATION.
BPL  TP10                ; STATUS WAS POSITIVE -- OK.
JMP  ERROR               ; NO -- "OPEN" PROBLEM.

; SETUP TO READ A RECORD.

TP10  LDA  #GETREC       ; SETUP "GET RECORD" COMMAND.
      STA  ICCOM, X

LDA  #BUFF                ; SETUP DATA BUFFER POINTER.
STA  ICBAL, X
LDA  #BUFF/256            ; SETUP MAX RECORD SIZE ...
STA  ICBAH, X             ; ... PRIOR TO EVERY READ.

LOOP  LDA  #BUFFSZ         ; SETUP MAX RECORD SIZE ...
      STA  ICBLL, X        ; ... PRIOR TO EVERY READ.
LDA  #BUFFSZ/256          ; SETUP MAX RECORD SIZE ...
      STA  ICBLH, X

JSR  CIOV                ; READ A RECORD.
BMI  TP20                ; MAY BE END OF FILE.

; A RECORD IS NOW IN THE DATA BUFFER "BUFF". IT IS TERMINATED BY

OPERATING SYSTEM CO16555 -- Section 5
AN EOL CHARACTER, AND THE RECORD LENGTH IS IN "ICBLL" and "ICBLH".
THIS EXAMPLE WILL DO NOTHING WITH THE RECORD JUST READ.

JMP LOOP ; READ NEXT RECORD.

NEGATIVE STATUS ON READ -- CHECK FOR END OF FILE.

TP20 CPY #EOF ; END OF FILE STATUS?
  BNE ERROR ; NO -- ERROR.
  LDA #CLOSE ; YES -- CLOSE FILE.
  STA ICCOM,X
  JSR CIOV ; CLOSE THE FILE.
  JMP * ; *** END OF PROGRAM ***

DATA REGION OF EXAMPLE PROGRAM

NAME . BYTE "D1:TESTER", EOL
BUFFSZ= 80 ; 80 CHARACTER RECORD MAX
          (INCLUDES EOL).
BUFF= * ; READ BUFFER.
**= **+BUFFSZ
.END

Figure 5-3 An I/O Example
Device-Specific Information

This section provides device-specific information regarding the device handlers that interface to CIO.

Keyboard Handler (K:)

The keyboard device is a read only device with a handler that supports the following CIO functions:

- OPEN
- CLOSE
- GET CHARACTERS
- GET RECORD
- GET STATUS (null function)

The Keyboard Handler can produce the following error statuses:

- $80 -- [BREAK] key abort.
- $88 -- end-of-file (produced by pressing [CTRL] 3).

The Keyboard Handler is one of the resident handlers. It has a set of device vectors starting at location E420.

The keyboard can produce any of the 256 codes in the ATASCII character set (see Appendix F). Note that a few of the keyboard keys do not generate data at the Keyboard Handler level. These keys are described below:

- [\/] - The ATARI key toggles a flag that enables/disables the inversion of bit 7 of each data character read. The Screen Editor editing keys are exempted from such inversion, however.

- CAPS - The [CAPS/LOWR] key provides three functions:
  - [SHIFT][CAPS/LOWR] -- Alpha caps lock.
  - [CAPS/LOWR] -- Alpha unlock.
The system powers up and will system reset to the alpha caps lock option.

Some key combinations are ignored by the handler, such as [CTRL] 4 through [CTRL] 9, [CTRL] 0, [CTRL] 1, [CTRL] /, and all key combinations in that the [SHIFT] and [CTRL] keys are depressed simultaneously.

The [CTRL] 3 key generates an EOL character and returns EOF status.

The [BREAK] key generates an EOL character and returns BREAK status.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions (described earlier in this section) are detailed below:

OPEN

The device name is K, and the handler ignores any device number and filename specification, if included.

There are no device-dependent option bits in AUX1 or AUX2.

CLOSE

No special handler actions.

GET CHARACTERS and GET RECORD

The handler returns the ATASCII key codes to CID as they are entered, with no facility for editing.

GET STATUS

The handler does nothing but set the status to $01.

Theory of Operation

Pressing a keyboard key generates an IRQ interrupt and vectors to the Keyboard Handler's interrupt service routine (see Section 6). The key code for the key pressed is then read and stored in data base variable CH [02FC]. This occurs whether or not there is an active read request to the Keyboard Handler, and effects a one-byte FIFO for keyboard entry. See Appendix L (GB) for a discussion of the auto repeat feature.
The Keyboard Handler monitors the CH variable for not containing the value $FF$ (empty state) whenever there is an active read request for the handler. When CH shows nonempty, the handler takes the key code from CH and sets CH to $FF$ again. The key code byte obtained from CH is not an ATASCII code and has the following form:

```
 7 0
+-+-+-+-+-+-+-+-+-+-+
|C|S| key code |
+-+-+-+-+-+-+-+-+-+-+
```

Where:  
C = 1 if the [CTRL] key is pressed.  
S = 1 if the [SHIFT] key is pressed.

The remaining six bits are the hardware key code.

The key code obtained is then converted to ATASCII using the first of the following rules that applies:

1. Ignore the code if the C and S bits are both set.
2. If the C bit is set, process the key as a [CTRL] code.
3. If the S bit is set, process the key as a [SHIFT] code.
4. If [CTRL] lock is in effect, process alpha characters as CTRL codes, all others as lowercase.
5. IF [SHIFT] lock is in effect, process alpha characters as SHIFT codes, all others as lowercase.
6. Else, process as lowercase character.

Then: If the resultant code is not a Screen Editor control code, and if the video inverse flag is set, then set bit 7 of the ATASCII code (will cause inverse video when displayed).
### KEY CODE TO ATASCII CONVERSION TABLE

<table>
<thead>
<tr>
<th>Key Code</th>
<th>Key Lwr.</th>
<th>[SHIFT]</th>
<th>[CTRL]</th>
<th>Key Code</th>
<th>Key Lwr.</th>
<th>[SHIFT]</th>
<th>[CTRL]</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>L</td>
<td>6C</td>
<td>4C</td>
<td>0C</td>
<td>,</td>
<td>2C</td>
<td>5B</td>
</tr>
<tr>
<td>01</td>
<td>J</td>
<td>6D</td>
<td>4D</td>
<td>0A</td>
<td>21</td>
<td>SPACE</td>
<td>20</td>
</tr>
<tr>
<td>02</td>
<td>;</td>
<td>3B</td>
<td>3A</td>
<td>7B</td>
<td>22</td>
<td>.</td>
<td>2E</td>
</tr>
<tr>
<td>03</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>23</td>
<td>N</td>
<td>6E</td>
</tr>
<tr>
<td>04</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>24</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>05</td>
<td>K</td>
<td>6B</td>
<td>4B</td>
<td>0B</td>
<td>25</td>
<td>M</td>
<td>6D</td>
</tr>
<tr>
<td>06</td>
<td>+</td>
<td>2B</td>
<td>5C</td>
<td>1E</td>
<td>26</td>
<td>/</td>
<td>2F</td>
</tr>
<tr>
<td>07</td>
<td>*</td>
<td>2A</td>
<td>5E</td>
<td>1F</td>
<td>27</td>
<td>/\</td>
<td>--</td>
</tr>
<tr>
<td>08</td>
<td>O</td>
<td>6F</td>
<td>4F</td>
<td>0F</td>
<td>28</td>
<td>R</td>
<td>72</td>
</tr>
<tr>
<td>09</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>29</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>0A</td>
<td>P</td>
<td>70</td>
<td>50</td>
<td>10</td>
<td>2A</td>
<td>E</td>
<td>65</td>
</tr>
<tr>
<td>0B</td>
<td>U</td>
<td>75</td>
<td>55</td>
<td>15</td>
<td>2B</td>
<td>Y</td>
<td>79</td>
</tr>
<tr>
<td>0C</td>
<td>RET</td>
<td>9B</td>
<td>9B</td>
<td>9B</td>
<td>2C</td>
<td>TAB</td>
<td>7F</td>
</tr>
<tr>
<td>0D</td>
<td>I</td>
<td>69</td>
<td>49</td>
<td>09</td>
<td>2D</td>
<td>T</td>
<td>74</td>
</tr>
<tr>
<td>0E</td>
<td>-</td>
<td>2D</td>
<td>5F</td>
<td>1C</td>
<td>2E</td>
<td>W</td>
<td>77</td>
</tr>
<tr>
<td>0F</td>
<td>=</td>
<td>3D</td>
<td>7C</td>
<td>1D</td>
<td>2F</td>
<td>Q</td>
<td>71</td>
</tr>
<tr>
<td>10</td>
<td>V</td>
<td>76</td>
<td>56</td>
<td>16</td>
<td>30</td>
<td>9</td>
<td>39</td>
</tr>
<tr>
<td>11</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>31</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>63</td>
<td>43</td>
<td>03</td>
<td>32</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>13</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>33</td>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td>14</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>34</td>
<td>BACKS</td>
<td>7E</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>62</td>
<td>42</td>
<td>02</td>
<td>35</td>
<td>8</td>
<td>3B</td>
</tr>
<tr>
<td>16</td>
<td>X</td>
<td>7B</td>
<td>5B</td>
<td>18</td>
<td>36</td>
<td>C</td>
<td>3C</td>
</tr>
<tr>
<td>17</td>
<td>Z</td>
<td>7A</td>
<td>5A</td>
<td>1A</td>
<td>37</td>
<td>&gt;</td>
<td>3E</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>34</td>
<td>24</td>
<td>--</td>
<td>3B</td>
<td>F</td>
<td>66</td>
</tr>
<tr>
<td>19</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>39</td>
<td>H</td>
<td>68</td>
</tr>
<tr>
<td>1A</td>
<td>3</td>
<td>33</td>
<td>23</td>
<td>9B*</td>
<td>3A</td>
<td>D</td>
<td>64</td>
</tr>
<tr>
<td>1B</td>
<td>6</td>
<td>36</td>
<td>26</td>
<td>--</td>
<td>3B</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>1C</td>
<td>[ESC]</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>3C</td>
<td>CAPS</td>
<td>--</td>
</tr>
<tr>
<td>1D</td>
<td>5</td>
<td>35</td>
<td>25</td>
<td>--</td>
<td>3D</td>
<td>G</td>
<td>67</td>
</tr>
<tr>
<td>1E</td>
<td>2</td>
<td>32</td>
<td>22</td>
<td>FD</td>
<td>3E</td>
<td>S</td>
<td>73</td>
</tr>
<tr>
<td>1F</td>
<td>1</td>
<td>31</td>
<td>21</td>
<td>--</td>
<td>3F</td>
<td>A</td>
<td>61</td>
</tr>
</tbody>
</table>

* [CTRL] 3 returns EOF status.

A complement of this table (ATASCII to keystroke) is given in Appendix F.

**Figure 5-4** Keycode to ATASCII Conversion Table
Display Handler (S:)

The display device is a read/write device with a handler that supports the following CIO functions:

OPEN
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS (null function)
DRAW
FILL

The Display Handler can produce the following error statuses:

$84 -- Invalid special command.
$8D -- Cursor out-of-range.
$91 -- Screen mode > 11.
$93 -- Not enough memory for screen mode selected.

The Display Handler is one of the resident handlers, and therefore has a set of device vectors starting at location E410.

Screen Modes

You can operate the display screen in any of 20 configurations (modes 1 through 8, with or without split screen; plus mode 0, and modes 9 through 11 without split screen). Mode 0 is the text displaying mode. Modes 1 through 11 are all graphics modes (although modes 2 and 3 do display a subset of the ATASCII character set). Modes 9 through 11 require a GTIA chip to be installed in place of the standard CTIA chip.

TEXT MODE 0

In text mode 0 the screen is comprised of 24 lines of 40 characters per line. Program alterable left and right margins limit the display area. They default to 2 and 39 (of a possible 0 and 39).
A program-controllable cursor shows the destination of the next character to be output onto the screen. The cursor is visible as the inverse video representation of the current character at the destination position.

The text screen data is internally organized as variable length logical lines. The internal representation is 24 lines when the screen is cleared. Each EDL marks the end of a logical line as text is sent to the screen. If more than 3 physical lines of text are sent, a logical line will be formed every 3 physical lines. The number of physical lines used to comprise a logical line (1 to 3) is always the minimum required to hold the data for that logical line.

The text screen "scrolls" upward whenever a text line at the bottom row of the screen extends past the right margin, or a text line at the bottom row is terminated by an EDL. Scrolling removes the entire logical line that starts at the top of the screen, and then moves all subsequent lines upward to fill in the void. The cursor also moves upward, if the logical line deleted exceeds one physical line.

All data going to or coming from the text screen is represented in 8-bit ATASCII code as shown in Appendix E.

TEXT MODES 1 AND 2

In text modes 1 and 2 the screen comprises either 24 lines of 20 characters (mode 1), or 12 lines of 20 characters (mode 2). The left and right margins are of no consequence in these modes and there is no visible cursor. There are no logical lines associated with the data and in all regards these modes are treated as graphics modes by the handler.

Data going to or coming from the screen is in the form shown below:

```
7 ++-----++-+-
! C ! D !
++-----++-+-+
```

Where: C is the color/character-set select field
D is a 5-bit truncated ATASCII code that selects the specific character within the set selected by the \( C \) field. See Appendix E for the graphics representations of the characters.

Data base variable CHBAS [02F4] allows for the selection of either of two data sets. The default value of $EO provides the capital letters, numbers and punctuation characters; the alternate value of $E2 provides lowercase letters and the special character graphics set.

Figure 5-5  Text Modes 1 and 2 Data Form

GRAPHICS MODES (Modes 3 Through 11)

The screen has varying physical characteristics for each of the graphics modes as shown in Appendix H. Depending upon the mode, a 1 to 16 color selection is available for each pixel and the screen size varies from 20 by 12 (lowest resolution) to 320 by 192 (highest resolution) pixels.

There is no visible cursor for the graphics mode output.

Data going to or coming from the graphics screen is represented as 1 to 8-bit codes as shown in Appendix H and in the GET/PUT diagrams following.

SPLIT-SCREEN CONFIGURATIONS

In split-screen configurations, the bottom of the screen is reserved for four lines of mode 0 text. The text region is controlled by the Screen Editor, and the graphics region is controlled by the Display handler. Two cursors are maintained in this configuration so that the screen segments can be managed independently.
To operate in split-screen mode, the Screen Editor must first be opened and then the Display Handler must be opened using a separate IDCB (with the split-screen option bit set in AUX1).

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions (described earlier in this section) are detailed below:

OPEN

The device name is S, and the handler ignores any device number and filename specification, if included.

The handler supports the following options:

```
7    0
+++++++
AUX1 ! !C:S:W:R! !
+++++++
```

Where:  
- \( C = 1 \) indicates to inhibit screen clear on OPEN.  
- \( S = 1 \) indicates to set up a split-screen configuration (for modes 1 through 8 only).  
- \( R \) and \( W \) are the direction bits (read and write).

```
7    0
+++++++
AUX2 ! !  mode !
+++++++
```

Where: \( \) mode is the screen mode (0 through 11).

Note: If the screen mode selected is 0, then the AUX1 C and S options are assumed to be 0.

You share memory utilization with the Display Handler information. Sharing is necessary because the Display Handler dynamically allocates high address memory for use in generating the screen display, and because different amounts of memory are needed for the different screen modes. Prior to initiating an OPEN command the variable APPMHI [000E] should contain the highest address of RAM you need. The Screen handler will open the screen only if no RAM is needed at or below that address.

Upon return from a screen OPEN, the variable MEMTOP [02E5] will contain the address of the last free byte at the end of RAM memory prior to the screen-required memory.
As a result of every OPEN command, the following screen variables are altered:

The text cursor is enabled (CRSINH = 0). The tabs are set to the default settings (2 and 39). The color registers are set to the default values (shown in Appendix H).

Tabs are set at positions 7, 15, 23, 31, 39, 47, 55, 63, 71, 79, 87, 95, 103, 111, 119.

CLOSE

No special handler actions.

GET CHARACTERS and GET RECORD

Returns data in the following screen mode dependent forms, where each byte contains the data for one cursor position (pixel); there is no facility for having the handler return packed graphics data.

```
7
+++++++0
1      ! ATASCII ! Mode 0
+++++++0
1      ! C ! D !
+++++++0

+ Modes 1,2 -- C = color/data set.
D = truncated ATASCII.

+++++++0
1      ! zero ! D !
+++++++0
Modes 3,5,7 -- D = color.

+++++++0
1      ! zero ! D !
+++++++0
Modes 4,6,8 -- D = color.

+++++++0
1      ! zero ! D !
+++++++0
Modes 9,10,11 -- D = data.
```

Figure 5-6 Graphics Mode 3-11 GET Data Form

The cursor moves to the next position as each data byte is returned. For mode 0, the cursor will stay within the specified margins; for all other modes, the cursor ignores the margins.
PUT CHARACTERS and PUT RECORD

The handler accepts display data in the following screen mode dependent forms; there is no facility for the handler to receive graphics data in packed form.

```
7  O
+++++ATASCII++++
! C ! D !
+++++++------+
Modes 1, 2 -- C = color/data set,
D = truncated ATASCII.

+++++------++
! ? ! D !
+++++++------+
Modes 3, 5, 7 -- D = color.

+++++------++
! ? !D !
+++++++------+
Modes 4, 6, 8 -- D = color.

+++++------++
! ? ! D !
+++++++------+
Modes 9, 10, 11 -- D = data.
```

Figure 5-7 Graphics Mode 3-11 PUT Data Form

NOTE: For all modes, if the output data byte equals $9B (EOL), that byte will be treated as an EOL character; and if the output data byte equals $7D (CLEAR) that byte will be treated as a screen-clear character.

The cursor moves to the next cursor position as each data byte is written. For mode 0, the cursor will stay within the specified margins; for all other modes, the cursor ignores the margins.

While outputting, the Display Handler monitors the keyboard to detect the pressing of the [CTRL] 1 key combination. When this occurs, the handler loops internally until that key combination is pressed again: This effects a stop/start function that freezes the screen display. Note that there is no ATASCII code associated with either the [CTRL] 1 key combination or the start/stop function. The stop/start function can be controlled only from the keyboard (or by altering database variable CH as discussed in Appendix L, E4).

OPERATING SYSTEM CO16555 -- Section 5
GET STATUS

No handler action except to set the status to $01.

DRAW

This special command draws a simulated "straight" line from the current cursor position to the location specified in ROWCRS [0054] and COLCRS [0055]. The color of the line is taken from the last character processed by the Display Handler or Screen Editor. To force the color, store the desired value in ATACHR [02FB]. At the completion of the command, the cursor will be at the location specified by ROWCRS and COLCRS.

The value for the command byte for DRAW is $11.

FILL

This special command fills an area of the screen defined by two lines with a specified color. The command is set up the same as in DRAW, but as each point of the line is drawn, the routine scans to the right performing the procedure shown below (in PASCAL notation):

\[
\text{WHILE PIXEL}[\text{ROW}, \text{COL}] = 0 \text{ DO BEGIN}
\text{PIXEL}[\text{ROW}, \text{COL}] := \text{FILDAT};
\text{COL} := \text{COL} + 1;
\text{IF COL} > \text{Screen right edge THEN COL} := 0
\text{END;}
\]

An example of a FILL operation is shown below:

```
+ 1
+---------------------------------+ +-------------------------------+
+---------------------------------+
4 +---------------------------------+
```

Where: '−' represents the fill operation.
'+' are the line points, with '+' for the endpoints.

1 -- set cursor and plot point.
2 -- set cursor and DRAW line.
3 -- set cursor and plot point.
4 -- set fill data value, set cursor, and FILL.
FILDAT [02FD] contains the fill data, and ROWCRS and COLCRS contain the cursor coordinates of the line endpoint. The value in ATACHR [02FB] will be used to draw the line; ATACHR always contains the last data read or written, so if the steps above are followed exactly, ATACHR will not have to be modified.

The value for the command byte for FILL is $12.

User-Alterable Data Base Variables

Certain functions of the Display Handler require you to examine and/or alter variables in the OS database. The following describes some of the more commonly used handler variables. (see Appendix L, B1-55, for additional descriptions).

Cursor Position

Two variables maintain the cursor position for the graphics screen or mode 0 text screen. ROWCRS [0054] maintains the display row number; and COLCRS [0055] maintains the display column number. Both numbers range from 0 to the maximum number of rows/columns, - 1. The cursor can be set outside of the defined text margins with no ill effect. You can read and write this region. The home position (0,0) for both text and graphics is the upper left corner of the screen.

ROWCRS is a single byte. COLCRS is maintained at 2-bytes, with the least significant byte being at the lower address.

When you alter these variables, the screen representation of the cursor will not move until the next I/O operation involving the display is performed.

Inhibit/Enable Visible Cursor Display

You can inhibit the display of the text cursor on the screen by setting the variable CRSINH [02FO] to any nonzero value. Subsequent I/O will not generate a visible cursor.

You can enable the display of the text cursor by setting CRSINH to zero. Subsequent I/O will then generate a visible cursor.

Text Margins

The text screen has user-alterable left and right margins. The OS sets these margins to 2 and 39. The variable LMARGIN [0052] defines the left margin, and the variable RMARGIN [0053] defines the right margin. The leftmost margin value is 0 and the
rightmost margin value is 39.

The margin values inclusively define the useable portion of the screen for all operations in that you do not explicitly alter the cursor location variables as described prior to this paragraph.

Color Control

The OS updates hardware color registers using data from the OS data base as part of normal Stage 2 VBLANK processing (see Section 6). Shown below are the data base variable names, the hardware register names, and the function of each register. See Appendix H for the mode dependent uses for the registers.

<table>
<thead>
<tr>
<th>Data Base</th>
<th>Hardware</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR0</td>
<td>COLPF0</td>
<td>PFO -- Playfield 0.</td>
</tr>
<tr>
<td>COLOR1</td>
<td>COLPF1</td>
<td>PF1 -- Playfield 1.</td>
</tr>
<tr>
<td>COLOR2</td>
<td>COLPF2</td>
<td>PF2 -- Playfield 2.</td>
</tr>
<tr>
<td>COLOR3</td>
<td>COLPF3</td>
<td>PF3 -- Playfield 3.</td>
</tr>
<tr>
<td>COLOR4</td>
<td>COLBK</td>
<td>BAK -- Playfield background.</td>
</tr>
<tr>
<td>PCOLRO</td>
<td>COLPMO</td>
<td>PMO -- Player/missile 0.</td>
</tr>
<tr>
<td>PCOLR1</td>
<td>COLPM1</td>
<td>PM1 -- Player/missile 1.</td>
</tr>
<tr>
<td>PCOLR2</td>
<td>COLPM2</td>
<td>PM2 -- Player/missile 2.</td>
</tr>
<tr>
<td>PCOLR3</td>
<td>COLPM3</td>
<td>PM3 -- Player/missile 3.</td>
</tr>
</tbody>
</table>

Theory of Operation

The Display Handler automatically sets up all memory resources required to create and maintain the screen display at OPEN time. The screen generation hardware requires that two distinct data areas exist for graphics modes: 1) a display list and 2) a screen data region. A third data area must exist for text modes. This data area defines the screen representation for each of the text characters. Consult the ATARI Home Computer Hardware Manual for a complete understanding of the material that is to follow.
The simplified block diagram below shows the relationships between the memory and hardware registers used to set up a screen display (without player/missile graphics) by the OS. Note that the hardware registers allow for many other possibilities.

DATA BASE VARIABLE

HARDWARE REGISTER
(Updated every VBLANK)

+----------+
| MEMTOP   |
+----------+

+----------+
| Display  |
| SDLSTL   |
+----------+

+----------+
| List     |
+----------+

= =

+----------+
| Screen Data |
| SAVMSC    |
+----------+

= =

+----------+
| Graphics  |
| and/or    |
| Text      |
+----------+

End of RAM memory

+----------+
| Specials and Numbers |
| E000       |
+----------+

+----------+
| Capital   |
| E100      |
+----------+

+----------+
| Special   |
| Graphics  |
| E200      |
+----------+

+----------+
| Lowercase |
| Letters   |
| E300      |
+----------+
Figure 5-8 Screen Display Block Diagram

The following relationships are present in the preceding diagram:

1. Data base variables SDLSTL/SDLSTH contain the address of the current display list. This address is stored in the hardware display list address registers DLISTL and DLISTH as part of the VBLANK process.

2. The display list itself defines the characteristics of the screen to be displayed and points to the memory containing the data to be displayed.

3. Data base variable CHBAS contains the MSB of the base address of the character representations for the character data (text modes only).

The default value for this variable is $EO. This variable declares that the character representations start at memory address E000 (the character set provided by the OS in ROM). Each character is defined as an 8X8 bit matrix, requiring 8 bytes per character. 1024 bytes are required to define the largest set, since a character code contains up to 7 significant bits (set of 128 characters). The OS ROM contains the default set in the region from E000 to E3FF.

All character codes are converted by the handler from ATASCII to an internal code (and vice versa), as shown below:

<table>
<thead>
<tr>
<th>ATASCII CODE</th>
<th>INTERNAL CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-1F</td>
<td>40-5F</td>
</tr>
<tr>
<td>20-3F</td>
<td>00-1F</td>
</tr>
<tr>
<td>40-5F</td>
<td>20-3F</td>
</tr>
<tr>
<td>60-7F</td>
<td>60-7F</td>
</tr>
<tr>
<td>80-9F</td>
<td>C0-DF</td>
</tr>
<tr>
<td>A0-BF</td>
<td>80-9F</td>
</tr>
<tr>
<td>C0-DF</td>
<td>A0-BF</td>
</tr>
<tr>
<td>E0-FF</td>
<td>E0-FF</td>
</tr>
</tbody>
</table>
The character set in ROM is ordered by internal code order. Three considerations differentiate the internal code from the external (ATASCII) code:

ATASCII codes for all but the special graphics characters were to be similar to ASCII. The alphabetic, numeric, and punctuation character codes are identical to ASCII.

In text modes 1 and 2 it was desired that one character subset include capital letters, numbers, and punctuation and the other character subset include lowercase letters and special graphics characters.

The codes for the capital and lowercase letters were to be identical in text modes 1 and 2.

Database variables COLOR0 through COLOR4 contain the current color register assignments. Hardware color registers receive these values as part of the stage 1 VBLANK process, thus providing synchronized color changes (see Appendix H).

Database variable SAVMSC points to the lowest memory address of the screen data region. It corresponds to the data displayed at the upper left corner of the display.

When the Display Handler receives an open command, it first determines the screen mode from the OPEN IOCB. Then it allocates memory from the end of RAM downward (as specified by database variable RAMTOP), first for the screen data and then for the display list. The screen data region is cleared and the display list is created if sufficient memory is available. The display list address is stored to the database.
The Screen Editor is a read/write handler that uses the Keyboard Handler and the Display Handler to provide "line-at-a-time" input with interactive editing functions, as well as formatted output.

The Screen Editor supports the following CIO functions:

OPEN
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS (null function)

See Keyboard Handler and Display Handler Sections for a discussion of Screen Editor error statuses.

The Screen Editor is one of the resident handlers, and therefore has a set of device vectors starting at location E400.

The Screen Editor is a program that reads key data from the Keyboard Handler and sends each character to the Display Handler for immediate display. The Screen Editor also accepts data from you to send to the Display Handler, and reads data from the Display Handler (not the Keyboard Handler) for you. In fact, the Keyboard Handler, Display Handler, and the Screen Editor are all contained in one monolithic hunk of code.

Most of the behaviors already defined for the Keyboard Handler and the Display Handler apply as well to the Screen Editor: The discussions in this Section will be limited to deviations from those behaviors, or to additional features that are part of the Screen Editor only. The Screen Editor deals only with text data (screen mode 0). This Section also explains the split-screen configuration feature.

The Screen Editor uses the Display Handler to read data from graphics and text screens on demand. You use the Screen Editor to determine when the program will read Screen data, and where upon the screen the data will be read from. You first locates the cursor on the screen to determine the screen area to be read; you then press the [RETURN] key to determine when the program will begin to read the data indicated.
When the [RETURN] key is pressed, the entire logical line within that the cursor resides is then made available to the calling program: Trailing blanks in a logical line are never returned as data, however. After all of the data in the line has been sent to the caller (this can entail multiple READ CHARACTERS functions if desired), an EOL character is returned and the cursor is positioned to the beginning of the logical line following the one just read.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN

The device name is E, and the Screen Editor ignores any device number and filename specification, if included.

The Screen Editor supports the following option:

```
  7
+--------++
AUX1 1 !W!R! !F!
+--------++
```

Where: R and W are the direction bits (read and write).

F = 1 indicates that a "forced read" is desired (see GET CHARACTER and GET RECORD for more information).

CLOSE

No special handler actions.

GET CHARACTER and GET RECORD

Normally the Screen Editor will return data only when you press the [RETURN] key at the keyboard. However, the "forced read" OPEN option allows you to read text data without intervention. When you command a READ operation, the Screen Editor will return data from the start of the logical line in which the text cursor is located, and then move the cursor to the beginning of the following logical line. A read of the last logical line on the screen will cause the screen data to scroll.

A special case occurs when characters are output without a terminating EOL, and then additional characters are appended to the end of the previous logical line.
that logical line from the keyboard. When the [RETURN] key is pressed, only the keyboard entered characters are sent to the caller, unless the cursor has been moved out of and then back into the logical line, in that case all of the logical line will be sent.

PUT CHARACTER and PUT RECORD

The Handler accepts ATASCII characters as one character per byte. Sixteen of the 256 ATASCII characters are control codes; the EOL code has universal meaning, but most of the other control codes have special meaning only to a display or print device. The Screen Editor processing of the ATASCII control codes is explained below:

CLEAR ($7D) -- The Screen Editor clears the current display of all data and the cursor is placed at the home position (upper left corner of the screen).

CURSOR UP ($1C) -- The cursor moves up by one physical line. The cursor will wrap from the top line of the display to the bottom line.

CURSOR DOWN ($1D) -- The cursor moves down by one physical line. The cursor will wrap from the bottom line of the display to the top line.

CURSOR LEFT ($1E) -- The cursor moves left by one column. The cursor will wrap from the left margin of a line to the right margin of the same line.

CURSOR RIGHT ($1F) -- The cursor moves right by one column. The cursor will wrap from the right margin of a line to the left margin of the same line.

BACKSPACE ($7E) -- The cursor moves left by one column (but never past the beginning of a logical line), and the character at that new position is changed to a blank ($20).
SET TAB ($9F) -- The Screen Editor establishes a tab point at the logical line position at which the cursor is residing. The logical line tab position is not synonymous with the physical line column position since the logical line can be up to 3 physical lines in length. For example, tabs can be set at the 15th, 30th, 45th, 60th, and 75th character positions of a logical line as shown below:

```
  0 2 9 19 29 39 Screen column #.
--L---------------------------R L/R = margins.
xx-------------------T---------T------ A logical line.
xx------T---------T------T------T x = inaccessible columns.
xx-------------------------------
```

Note the effect of the left margin in defining the limits of the logical line.

The Handler default tab settings are shown below:

```
  0 2 9 19 29 39 Screen column #.
--L---------------------------R L/R = margins.
xxT----T-------T-------T-------T-------T A logical line.
xx-----T-------T-------T-------T-------T x = inaccessible columns.
xx-----T-------T-------T-------T-------T
```

CLEAR TAB ($9E) -- The Screen Editor clears the current cursor position within the logical line from being a tab point. There is no "clear all tab points" facility provided by the Handler.

TAB ($7F) -- The cursor moves to the next tab point in the current logical line, or to the beginning of the next line if no tab point is found. This function will not increase the logical line length to accommodate a tab point outside the current length (e.g. the logical line length is 38 characters and there is a tab point at position 50).

INSERT LINE ($90) -- All physical lines at and below the physical line in that the cursor resides, are moved down by one physical line. The last logical line on the display can be truncated as a result. The blank physical line at the insert point becomes the beginning of a new logical line. A logical line can be split into two logical lines by this process, the last half of the original logical line being concatenated with the blank physical line formed at the insert point.

OPERATING SYSTEM CO16555 -- Section 5
DELETE LINE ($9C) -- The logical line in that the cursor resides is deleted and all data below that line is moved upward to fill the void. Empty logical lines are created at the bottom of the display.

INSERT CHARACTER ($FF) -- All physical characters at and behind the cursor position on a logical line are moved one position to the right. The character at the cursor position is set to blank. The last character of the logical line will be lost when the logical line is full and a character is inserted. The number of physical lines comprising a logical line can increase as a result of this function.

DELETE CHARACTER ($FE) -- The character on which the cursor resides is removed, and the remainder of the logical line to the right of the deleted character is moved to the left by one position. The number of physical lines composing a logical line can decrease as a result of this function.

ESCAPE ($1B) -- The next non-EOL character following this code is displayed as data, even if it would normally be treated as a control code. The sequence [ESC][ESC] will cause the second [ESC] character to be displayed.

BELL ($FD) -- An audible tone is generated; the display is not modified.

END OF LINE ($9B) -- In addition to its record termination function, the EOL causes the cursor to advance to the beginning of the next logical line. When the cursor reaches the bottom line of the screen, the receipt of an EOL will cause the screen data to scroll upward by one logical line.

GET STATUS

The Handler takes no action other than to set the status to $01.

User-Alterable Data Base Variables

Also see the Display Handler data base variable discussion.
Cursor Position

When in a split-screen configuration, ROWCRS and COLCRS are associated with the graphics portion of the display and two other variables, TXTROW [0290] and TXTCOL [0291], are associated with the text window. TXTROW is a single byte, and TXTCOL is 2-bytes with the least significant byte being at the lower address. Note that the most significant byte of TXTCOL should always be zero.

The home position (0,0) for the text window is the upper left corner of the window.

Enable/Inhibit of Control Codes in Text

Normally all text mode control codes are operated upon as received, but sometimes it is desirable to have the control codes displayed as if they were data characters. This is done by setting the variable DSPFLG [02FE] to any nonzero value before outputting the data containing control codes. Setting DSPFLG to zero restores normal processing of text control codes.
Cassette Handler (C:)

The Cassette device is a read or write device with a Handler that supports the following CIO functions:

OPEN
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS (null function)

The Cassette Handler can produce the following error statuses:

$80 -- [BREAK] key abort.
$84 -- Invalid AUX1 byte on OPEN.
$88 -- end-of-file.
$8A-90 -- SIO error set (see Appendix C).

The Cassette Handler is one of the resident handlers, and therefore has a set of device vectors starting at location E440.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN

The device name is C, and the Handler ignores any device number and filename specification, if included.

The Handler supports the following option:
Where: $C = 1$ indicates that the cassette is to be read/written without stop/start between records (continuous mode).

Opening the cassette for input generates a single audible tone, as a prompt for you to verify that the cassette player is set up for reading (power on; Serial Bus cable connected; tape cued to start of file; and PLAY button depressed). When the cassette is ready, you can press any keyboard key (except [BREAK]) to initiate tape reading.

Opening the cassette for output generates two closely spaced audible tones, as a prompt for you to verify that the cassette player is set up for writing (as above, plus RECORD button depressed). When the cassette is ready, you can press any keyboard key (except [BREAK]) to begin tape writing. There is no way for the computer to verify that the RECORD or PLAY button is depressed. It is possible for the file not to be written, with no immediate indication of this fact.

There is a potential problem with the cassette in that when the cassette is opened for writing, the motor keeps running until the first record (128 data bytes) is written. If 128 data bytes are written or the cassette is closed within about 30 seconds of the OPEN, and no other serial bus I/O is performed, then there is no problem. However, if those conditions are not met, some noise will be written to the tape prior to the first record and an error will occur when that tape file is read later. If lengthy delays are anticipated between the time the cassette file is opened and the time that the first cassette record (128 data bytes) is written, then a dummy record should be written as part of the file; typically 128 bytes of some innocuous data would be written, such as all zeros, all $FF$s, or all blanks ($20$).

The system sometimes emits whistling noises after cassette I/O has occurred. The sound can be eliminated by storing $03$ to SKCTL [D20F], thus bring POKEY out of the two-tone (FSK) mode.

CLOSE

The CLOSE of a tape read stops the cassette motor.

The CLOSE of a tape write does the following:

- Writes any remaining user data in the buffer to tape.
- Writes an end-of-file record.
- Stops the cassette motor.
GET CHARACTERS and GET RECORD

The Handler returns data in the following format:

```
7 0
+-+-+-+-+-+-+-+-+
| data byte |
+-+-+-+-+-+-+-+-+
```

PUT CHARACTERS and PUT RECORD

The Handler accepts data in the following format:

```
7 0
+-+-+-+-+-+-+-+-+
| data byte |
+-+-+-+-+-+-+-+-+
```

The Handler attaches no significance to the data bytes written, a value of $9B$ (EOL) causes no special action.

GET STATUS

The Handler does no more than set the status to $01$.

Theory of Operation

The Cassette Handler writes and reads all data in fixed-length records of the format shown below:

```
+-+-+-+-+-+-+-+-+
| 0 1 0 1 0 1 0 1 |
+-+-+-+-+-+-+-+-+
| 0 1 0 1 0 1 0 1 |
+-+-+-+-+-+-+-+-+
| control byte |
+-+-+-+-+-+-+-+-+
| 128 |
= data =
| bytes |
+-+-+-+-+-+-+-+-+
| checksum |
+-+-+-+-+-+-+-+-+
```

Figure 5-9 Cassette Handler Record Format
The control byte contains one of three values:

- $FC$ indicates the record is a full data record (128 bytes).
- $FA$ indicates the record is a partially full data record; you supplied fewer than 128 bytes to the record. This case can occur only in the record prior to the end-of-file. The number of user-supplied data bytes in the record is contained in the byte prior to the checksum.
- $FE$ indicates the record is an End-of file record; the data portion is all zeroes for an end-of-file record.

The SIO routine generates and checks the checksum. It is part of the tape record, but it is not contained in the Handler's record buffer CASBUF [03FD].

The processing of the speed-measurement bytes during cassette reading is discussed in Appendix L, D1-D7.

File Structure

The Cassette Handler writes a file to the cassette device with a file structure that is totally imposed by the Handler (soft format). A file consists of the following three elements:

- A 20-second leader of mark tone.
- Any number of data-record frames.
- An end-of-file frame.

The cassette-data record frames are formatted as shown below:

\[
frame = \text{pre-record write tone (PRWT)},
+ \text{data record},
+ \text{post record gap (PRG)}
\]

The nondata portions of a frame have characteristics that are dependent upon the write OPEN mode, i.e. continuous or start/stop.

- Stop/start PRWT = 3 seconds of mark tone.
- Continuous PRWT = .25 second of mark tone.
- Stop/start PRG = up to 1 second of unknown tones.
- Continuous PRG = from 0 to n seconds of unknown tones, where n is dependent upon your program timing.

The inter-record gap (IRQ) between any two records consists of the PRG of the first record followed by the PRWT of the second record.
Printer Handler (P:)

The Printer device is a write-only device with a Handler that supports the following CIO functions:

OPEN
CLOSE
PUT CHARACTERS
PUT RECORD
GET STATUS

The Printer Handler can produce the following error statuses:

$8A-90 -- SIO error set (see Appendix C).

The Printer Handler is one of the resident handlers, and therefore has a set of device vectors starting at location E430.

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

OPEN

The device name is P. The Handler ignores any device number and filename specification, if included.

CLOSE

The Handler writes any data remaining in its buffer to the printer device, with trailing blanks to fill out the line.

PUT CHARACTERS and PUT RECORD

The Handler accepts print data in the following format:

```
7   0
+++++++
```

The only ATASCII control code of any significance to the Handler is the EDL character. The printer device ignores bit 7 of every data byte and prints a subset of the remaining 128 codes. (see Appendix G for the printer character set).

The Handler supports the following print option:

OPERATING SYSTEM CD16555 -- Section 5
Where: $4E$ (N) selects normal printing (40 characters per line).
$53$ (S) selects sideways printing (29 characters per line).
$57$ (W) selects wide printing (not supported by printer device).

Any other value (including 00) is treated as a normal (N) print select, without producing an error status.

GET STATUS

The Handler obtains a 4-byte status from the printer controller and puts it in system location DVSTAT [02EA]. The format of the status bytes is shown below:

```
+-----------------+
| command stat.  |
+-----------------+    DVSTAT + 0
|                 |
+-----------------+    + 1
| AUX2 of prev.  |
+-----------------+    + 2
|                 |
+-----------------+    + 3
| timeout         |
+-----------------+    (unused)
|                 |
+-----------------+    
```

The command status contains the following status bits and condition indications:

bit 0: an invalid command frame was received.
bit 1: an invalid data frame was received.
bit 7: an intelligent controller (normally = 0).

The next byte contains the AUX2 value from the previous operation.

The timeout byte contains a controller provided maximum timeout value (in seconds).

Theory of Operation

The ATARI 820[TM] 40-Column Printer is a line-at-a-time printer rather than a character-at-a-time printer, so your data must be buffered by the Handler and sent to the device in records corresponding to one print line (40 characters for normal, 29 characters for sideways).
The printer device does not attach any significance to the EOL character, so the Handler does the appropriate blank fill whenever it sees an EOL.

Disk File Manager (D:)

The OS supports four unique File Management Subsystems at the time of this writing. Version IA is the original version. Version IB is a slightly modified version of IA and is the one described in this document. Most of this discussion applies as well to Version II, that handles a double-density diskette (720 256-byte sectors) in addition to the single-density diskette (720 128-byte sectors). Version III has all new file/directory/map structures and can possibly contain changes to your interface as well.

The File Management Subsystem includes a disk-bootable (RAM-resident) Disk File Manager (DFM) that maintains a collection of named files on diskettes. Up to 4 disk drives (D1: through D4:) can be accessed, and up to 64 files per diskette can be accessed. The system diskettes supplied by ATARI allow a single disk drive (D1) and up to 3 OPEN files, but you can alter these numbers as described later in this section.

The Disk File Manager supports the following CIO functions:

OPEN FILE
OPEN DIRECTORY
CLOSE
GET CHARACTERS
GET RECORD
PUT CHARACTERS
PUT RECORD
GET STATUS

NOTE
POINT
LOCK
UNLOCK
DELETE
RENAME
FORMAT
The Disk File Manager can produce the following error statuses:

- **$03**: Last data from file (EOF on next read).
- **$88**: End-of-file.
- **$8A-90**: SIO error set (see Appendix C).
- **$A0**: Drive number specification error.
- **$A1**: No sector buffer available (too many open files).
- **$A2**: Disk full.
- **$A3**: Fatal I/O error in directory or bitmap.
- **$A4**: Internal file # mismatch (structural problem).
- **$A5**: File name specification error.
- **$A6**: Point information in error.
- **$A7**: File locked to this operation.
- **$A8**: Special command invalid.
- **$A9**: Directory full (64 files).
- **$AA**: File not found.
- **$AB**: Point invalid (file not OPENed for update).

CIO Function Descriptions

The device-specific characteristics of the standard CIO functions are detailed below:

**OPEN FILE**

The device name is D. Up to four disk drives can be accessed (D1 through D4). The disk filename can be from 1 to 8 characters in length with an optional 1- to 3-character extension.

The OPEN FILE command supports the following options:

```
7 0
+++++++------
AUX1 ! W!R! A!
+++++++------
```

Where: W and R are the direction bits.

- **WR = 00** is invalid
- **01** indicates OPEN for read only.
- **10** indicates OPEN for write only.
- **11** indicates OPEN for read/write (update).

- **A = 1** indicates appended output when W = 1.

You may use these following valid AUX1 options:
OPEN Input (AUX1 = $04)

The indicated file is opened for input. Any wild-card characters are used to search for the first match. If the file is not found, an error status is returned, and no file will be opened.

OPEN Output (AUX1 = $08)

The indicated file is opened for output starting with the first byte of the file, if the file is not locked. Any wild-card characters are used to search for the first match. If the file already exists, the existing file will be deleted before opening the named file as a new file. If the file does not already exist, it will be created.

A file opened for output will not appear in the directory until it has been closed. If an output file is not properly closed, some or all of the sectors that were acquired for it can be lost until the disk is reformatted.

A file that is opened for output can not be opened concurrently for any other access.

OPEN Append (AUX1 = $09)

The indicated file is opened for output starting with the byte after the last byte of the existing file (that must already exist), if the file is not locked. Any wild-card characters are used to search for the first match.

If a file opened for append is not properly closed, the appended data will be lost. The existing file will remain unmodified and some or all of the sectors that were acquired for the appended portion can be lost until the diskette is reformatted.

OPEN Update (AUX1 = $0C)

The indicated file (that must already exist) will be opened for update provided it is not locked. Any wild-card characters are used to search for the first match.

The GET, PUT, NOTE and POINT operations are all valid, and can be intermixed as desired.

If a file opened for update is not properly closed, a sector's worth of information can be lost to the file. A file opened for update can not be extended.
Device/Filename Specification

The Handler expects to find a device/filename specification of the following form:

D[<number>]:<filename><EOL>

where:

<number> ::= 1|2|3|4
<filename> ::= [[<primary>].<extension>]]<terminator>
<primary> ::= an uppercase alpha character followed by 0 to 7 alphanumeric characters. If the primary name is less than 8 characters, it will be padded with blanks; if it is greater than 8 characters, the extra characters will be ignored.
<extension> ::= Zero to 3 alphanumeric characters. If the extension name is missing or less than 3 characters, it will be padded with blanks; if it is greater than 3 characters, the extra characters will be ignored.
<terminator> ::= <EOL>!<blank>

Figure 5-10  Device/Filename Syntax

The following are all valid device/filenames for the diskette:

D1:GAME.SRC
D:MANUAL6
D:.WHY
D3:FILE.
D4:BRIDGE.002

Filename Wildcarding

The filename specification can be further generalized to include the use of the "wild-card" characters * and ?. These wildcard characters allow portions of the primary and/or extension to be abbreviated as follows:

The * character in the specification allows any filename character at that position to produce a "match." For example, WH? will match files named WHO, WHY, WH4, etc., but not a file named WHAT.
The * character causes the remainder of the primary or extension field in that it is used to be effectively padded with ? characters. For example, WH* will match WHO, WHEN, WHATEVER, etc.

Some valid uses of wild-card specifications are shown below:

- *.SRC: Files having an extension of SRC.
- BASIC.*: Files named BASIC with any extension.
- *.*: All files.
- H*.?: Files beginning with H and having a 0 or 1 character extension.

If wildcarding is used with an OPEN FILE command, the first file found (if any) that meets the specification will be the one (and only one) opened.

**OPEN DIRECTORY**

The OPEN DIRECTORY command allows you read directory information for the selected filename(s), using normal GET CHARACTERS or GET RECORD commands. The information read will be formatted as ATASCII records, suitable for printing, as shown below. Wildcarding can be used to obtain information for multiple files or the entire diskette.

The OPEN DIRECTORY command uses the same CIO parameters as a standard OPEN FILE command:

- **COMMAND BYTE** = $03
- **BUFFER ADDRESS** = pointer to device/filename specification.
- **AUX1** = $06

After the directory is opened, a record will be returned to the caller for each file that matches the OPEN specification. The record, that contains only ATASCII characters, is formatted as shown below:

```
1 123456789 12345678
+-----------------+
|l|b| primary name | ext |b|count|e|
+-----------------+
```
Where: $s = *$ or ' ', with * indicating file locked.

- $b = \text{blank.}$
- $\text{primary name} = \text{left-justified name with blank fill.}$
- $\text{ext} = \text{left-justified extension with blank fill.}$
- $b = \text{blank.}$
- $\text{count} = \text{number of sectors comprising the file.}$
- $e = \text{EOL ($9B$).}$

After the last filename match record is returned, an additional record is returned. This record indicates the number of unused sectors available on the diskette. The format for this record is shown below:

```
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
!count! F R E E   S E C T O R S !e!
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Where: $\text{count} = \text{the number of unused sectors on the diskette.}$

- $e = \text{EOL ($9B$).}$

The EOF statuses ($03$ and $88$) are returned as in a normal data file when the last directory record is read.

The opening of another diskette file while the directory read is open will cause subsequent directory reads to malfunction, so care must be taken to avoid this situation.

**CLOSE**

Upon closing a file read, the Handler releases all internal resources being used to support that file.

Upon closing a file write, the Handler:

- $\text{writes any residual data from its file buffer for that file}$
- $\text{to the diskette.}$
- $\text{updates the directory and allocation map for the associated}$
- $\text{diskette.}$
- $\text{releases all internal resources being utilized to support}$
- $\text{that file}$

**GET CHARACTERS and GET RECORD**

Characters are read from the diskette and passed to CIO as a raw data stream. None of the ATASCII control characters have any special significance. A status of $88$ is returned if an attempt is made to read past the last byte of a file.
PUT CHARACTERS and PUT RECORD

Characters are obtained from CIO and written to the diskette as a raw data stream. None of the ATASCII control characters have any special significance.

GET STATUS

The indicated file is checked and one of the following status byte values is returned in ICSTA and register Y:

- $01 -- File found and unlocked.
- $A7 -- File locked.
- $AA -- File not found.

Special CIO Functions

The DFM supports a number of SPECIAL commands, that are device specific. These are explained in the paragraphs that follow:

NOTE (COMMAND BYTE = $25)

This command returns to the caller the exact diskette location of the next byte to be read or written, in the variables shown below:

- ICAX3 = LSB of the diskette sector number.
- ICAX4 = MSB of the diskette sector number.
- ICAX5 = relative sector displacement to byte (0-124).

POINT (COMMAND BYTE = $26)

This command allows you to specify the exact diskette location of the next byte to be read or written. In order to use this command, the file must have been opened with the "update" option.

- ICAX3 = LSB of the diskette sector number.
- ICAX4 = MSB of the diskette sector number.
- ICAX5 = relative sector displacement to byte (0-124).
LOCK

This command allows you to prevent write access to any number of named files. Locked files can not be deleted, renamed, nor opened for output unless they are first unlocked. Locking a file that is already locked is a valid operation. The Handler expects a device/filename specification; then all occurrences of the filename specified will be locked, using the wild-card rules.

You set up these following I/OCB parameters prior to calling CIO:

COMMAND BYTE = $23

BUFFER ADDRESS = pointer to device/filename specification.

After a LOCK operation, the following I/OCB parameter will have been altered:

STATUS = result of LOCK operation; see Appendix B for a list of possible status codes.

UNLOCK

This command allows you to remove the lock status of any number of named files. Unlocking a file that is not locked is a valid operation. The Handler expects a device/filename specification; then all occurrences of the filename specified will be unlocked, using the wild-card rules.

You set up these following I/OCB parameters prior to calling CIO:

COMMAND BYTE = $24

BUFFER ADDRESS = pointer to device/filename specification.

After an UNLOCK operation, the following I/OCB parameter will have been altered:

STATUS = result of UNLOCK operation; see Appendix B for a list of possible status codes.

DELETE

This command allows you to delete any number of unlocked named files from the directory of the selected diskette and to deallocate the diskette space used by the files involved. The Handler expects a device/filename specification; then all occurrences of the filename specified will be deleted, using the wild-card rules.
You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $21

BUFFER ADDRESS = pointer to device/filename specification.

After a DELETE operation, the following IOCB parameter will have been altered:

STATUS = result of DELETE operation; see Appendix B for a list of possible status codes.

RENAME

This command allows you to change the filenames of any number of unlocked files on a single diskette. The Handler expects to find a device/filename specification that follows:

<device spec>:<filename spec>,<filename spec><EOL>

All occurrences of the first filename will be replaced with the second filename, using the wild-card rules. No protection is provided against forming duplicate names. Once formed, duplicate names cannot be separately renamed or deleted; however, an OPEN FILE command will always select the first file found that matches the filename specification, so that file will always be accessible. The RENAME command does not alter the content of the files involved, merely the name in the directory.

Examples of some valid RENAME name specifications are shown below:

D1:*.SRC,*.TXT
D: TEMP,FDATA
D2:F*,F*.OLD

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $20

BUFFER ADDRESS = pointer to device/filename specification.

After a RENAME operation, the following IOCB parameter will have been altered:

STATUS = result of RENAME operation; see Appendix B for a list of possible status codes.
FORMAT

Soft-sector diskettes must be formatted before they can store data. The FORMAT command allows you to physically format a diskette. The physical formatting process writes a new copy of every sector on the soft-sectored diskette, with the data portion of each sector containing all zeros. The FORMAT process creates an "empty" non system diskette. When the formatting process is complete, the FMS creates an initial Volume Table of Contents (VTOC) and an initial File Directory. The boot sector (#1) is permanently reserved as part of this process.

You set up these following IOCB parameters prior to calling CIO:

COMMAND BYTE = $FE

BUFFER ADDRESS = pointer to device specification.

After a FORMAT operation, the following IOCB parameter will have been altered:

STATUS = result of FORMAT operation; see Appendix B for a list of possible status codes.

To create a system diskette, a copy of the boot file must then be written to sectors #2-n. This is accomplished by writing the file named DOS.SYS. This is a name that is recognized by the FMS even though it is not in the directory initially.

Theory of Operation

The resident OS initiates the disk-boot process (see Section 10). The OS reads diskette sector #1 to memory and then transfers control to the "boot continuation address" (boot address + 6). The boot-continuation program contained in sector #1 then continues to load the remainder of the File Management Subsystem to memory using additional information contained in sector #1. The File Management Subsystem loaded, will contain a Disk File Manager, and optionally, a Disk Utilities (DOS) package.

When the boot process is complete, the Disk File Manager will allocate additional RAM for the creation of sector buffers. Sector buffers are allocated based upon information in the boot record as shown below:

Byte 9 = maximum number of open files; one buffer per (the maximum value is 8).

Byte 10 = drive select bits; one buffer per (1-4 only).

OPERATING SYSTEM CO16555 -- Section 5
The Disk File Manager will then insert the name D and the Handler vector table address in the device table.

NOTE: There is a discrepancy between the Disk File Manager's numbering of diskette sectors (0-719) and the disk controller's numbering of diskette sectors (1-720); as a result, only sectors 1-719 are used by the Disk File Manager.

The Disk File Manager uses the Disk Handler to perform all diskette reads and writes; the DFM's function is to support and maintain the directory/file/bitmap structures as described in the following pages:
FMS Diskette Utilization

The map below shows the diskette sector utilization for a standard 720 sector diskette.

```
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+
|    BOOT record |     Sector 1                |    BOOT record |     Sector 1                |    BOOT record |     Sector 1                |    BOOT record |     Sector 1                |    BOOT record |     Sector 1                |    BOOT record |     Sector 1                |    BOOT record |     Sector 1                |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+
|    FMS BOOT    |     Sector 2  -->          |    FMS BOOT    |     Sector 2  -->          |    FMS BOOT    |     Sector 2  -->          |    FMS BOOT    |     Sector 2  -->          |    FMS BOOT    |     Sector 2  -->          |    FMS BOOT    |     Sector 2  -->          |    FMS BOOT    |     Sector 2  -->          |
| file           |                        | file           |                        | file           |                        | file           |                        | file           |                        | file           |                        | file           |                        |
| DOS.SYS        |                        | DOS.SYS        |                        | DOS.SYS        |                        | DOS.SYS        |                        | DOS.SYS        |                        | DOS.SYS        |                        | DOS.SYS        |                        |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+
|    User        |     Sector n  -->  Note 1  |    User        |     Sector n  -->  Note 1  |    User        |     Sector n  -->  Note 1  |    User        |     Sector n  -->  Note 1  |    User        |     Sector n  -->  Note 1  |    User        |     Sector n  -->  Note 1  |    User        |     Sector n  -->  Note 1  |
| File           |                        | File           |                        | File           |                        | File           |                        | File           |                        | File           |                        | File           |                        |
| Area           |                        | Area           |                        | Area           |                        | Area           |                        | Area           |                        | Area           |                        | Area           |                        |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+
|    VTOC(note 2)|     Sector 359  ($167)    |    VTOC(note 2)|     Sector 360  ($168)    |    VTOC(note 2)|     Sector 361  ($169)    |    VTOC(note 2)|     Sector 368  ($170)    |    VTOC(note 2)|     Sector 719  ($2CF)    |    VTOC(note 2)|     Sector 720  ($2DO)    |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+
|    File        |                        |    File        |                        |    File        |                        |    File        |                        |    File        |                        |    File        |                        |    File        |                        |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+       +----------------+
|    User        |                        |    User        |                        |    User        |                        |    User        |                        |    User        |                        |    User        |                        |    User        |                        |
| File           |                        | File           |                        | File           |                        | File           |                        | File           |                        | File           |                        |
| Area           |                        | Area           |                        | Area           |                        | Area           |                        | Area           |                        | Area           |                        |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+
|    unused      |                        |    unused      |                        |    unused      |                        |    unused      |                        |    unused      |                        |
+----------------+                 +----------------+       +----------------+       +----------------+       +----------------+
```

Figure 5-11  File Management Subsystem Diskette Sector Utilization Map

NOTE 1 - If the diskette is not a system diskette, then your File Area starts at sector 2 and no space is reserved for the FMS BOOT file. However, "DOS" (DOS.SYS and DUP.SYS) may still be written to a diskette that has already used sectors "2-N."

NOTE 2 -- VTOC stands for Volume Table of Contents.
FMS Boot Record Format

The FMS BOOT record (sector #1) is a special case of diskette-booted software (see Section 10). The format for the FMS BOOT record is shown below:

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Boot Flag = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td># Sectors = 1</td>
</tr>
<tr>
<td>2</td>
<td>Boot Address</td>
</tr>
<tr>
<td></td>
<td>= 0700</td>
</tr>
<tr>
<td>4</td>
<td>Init Address</td>
</tr>
<tr>
<td>6</td>
<td>JMP = $4B</td>
</tr>
<tr>
<td></td>
<td>Boot Read</td>
</tr>
<tr>
<td></td>
<td>Continuation</td>
</tr>
<tr>
<td></td>
<td>Address</td>
</tr>
<tr>
<td>9 Note 1</td>
<td>Max Files = 3</td>
</tr>
<tr>
<td>10 Note 2</td>
<td>Drive Bits = 1</td>
</tr>
<tr>
<td>11 Note 3</td>
<td>Alloc Dirk = 0</td>
</tr>
<tr>
<td></td>
<td>Boot Image End</td>
</tr>
<tr>
<td></td>
<td>Address + 1</td>
</tr>
<tr>
<td></td>
<td>FMS Configuration Data</td>
</tr>
<tr>
<td>14 Note 4</td>
<td>Boot Flag &lt;&gt; 0</td>
</tr>
<tr>
<td>15 Note 5</td>
<td>Sector Count</td>
</tr>
<tr>
<td></td>
<td>DOS.SYS</td>
</tr>
<tr>
<td></td>
<td>Starting</td>
</tr>
<tr>
<td></td>
<td>Sector Number</td>
</tr>
<tr>
<td></td>
<td>Code for Second</td>
</tr>
<tr>
<td></td>
<td>Phase of Boot</td>
</tr>
</tbody>
</table>

Figure 5-12 File Management Subsystem Boot Record Format

OPERATING SYSTEM CO1655 -- Section 5
NOTE 1 - Byte 9 specifies the maximum number of concurrently open files to be supported. This value can range from 1 to 8.

NOTE 2 - Byte 10 specifies the specific disk drive numbers to be supported using a bit encoding scheme as shown below:

\[
\begin{array}{cccccccc}
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
+-+-+-+---+---+---+---+---
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 4 & 3 & 2 & 1 & 1 & \text{where a 1 indicates a selected drive.} \\
+-+-+-+---+---+---+---+---
\end{array}
\]

NOTE 3 - Byte 11 specifies the buffer allocation direction, this byte should equal 0.

NOTE 4 - Byte 14 must be nonzero for the second phase of the boot process to initiate. This flag indicates that the file DOS.SYS has been written to the diskette.

NOTE 5 - This byte is assigned as being the sector count for the DOS.SYS file. It is actually an unused byte.
The diagram below shows how the boot sector (part of file DOS.SYS) and following sectors are loaded to memory as part of the boot process.

```
+----------------+ Memory address 0700
| data from boot |
| = sector read by |
| = resident OS |
| +----------------+
| data from rest |
| of DOS.SYS |
| read by the |
| = program in the |
| = boot sector. |
| +----------------+ 077C
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
|                             |
Volume Table of Contents

The format for the FMS volume table of contents (VTOC, sector 360) is shown in the diagram below:

```
+----------------+ Byte 0 Note 1
| directory type |
+----------------+
+----------------+ 1 Note 2
| maximum (lo) |
+ sector # +
| = 02C5 (hi) |
+----------------+ 3 Note 3
| number of (lo) |
+ sectors +
| available (hi) |
+----------------+ 10
| = |
| = |
+----------------+ = volume bit map =
| |
+----------------+ 11
| |
| = = |
| |
+----------------+ 99
```

Figure 5-14 File Management Subsystem Volume Table of Contents

The volume bit map organization location follows:

```
7     0

| 1 2 3 4 5 6 7 |

| 8 9 . . . . . |
= =
| |
```

Figure 5-15 File Management Subsystem Volume Bit Map

At each map bit position, a 0 indicates the corresponding sector is in use and a 1 indicates that the sector is available.

NOTE 1 - The directory type byte must equal 0.

NOTE 2 - The maximum sector number is not used because it is incorrectly set to 709 decimal. The true maximum sector number is actually 719 for the DFM.
NOTE 3 - The number of sectors available is initially set to 709 after a diskette is freshly formatted; this number is adjusted as files are created and deleted to show the number of sectors available. The sectors that are initially reserved are 1 and 360-368.

File Directory Format

The FMS reserves eight sectors (361-368) for a file directory. Each sector containing directory information for up to eight files, thus providing for a maximum of 64 files for any volume. The format of a single 16-byte file entry is shown below:

```
+----------------+
| flag byte      |
+----------------+
| sector (lo)    |
| count          |
| (hi)           |
+----------------+
| starting (lo)  |
| sector         |
| number (hi)    |
+----------------+
| (1)            |
| (2)            |
| (3)            |
| (4)            |
| (5)            |
| (6)            |
| (7)            |
| (8)            |
+----------------+
| file (1)       |
| name (2)       |
| extension (3)  |
+----------------+
Figure 5-16 File Directory Format
```

Where the flag byte has the following bits assigned:

OPERATING SYSTEM CO16555 -- Section 5
bit 7 = 1 if the file has been deleted.
bit 6 = 1 if the file is in use.
bit 5 = 1 if the file is locked.
bit 0 = 1 if OPEN output.

The flag byte can take on the following values:

$00 = entry not yet used (no file).
$40 = entry in use (normal CLOSED file).
$41 = entry in use (OPEN output file).
$60 = entry in use (locked file).
$80 = entry available (prior file deleted).

Sector count is the number of sectors comprising the file.

FMS File Sector Format

The format of a sector in your data file is shown below:

```
| 7 | 0 |
+---+---
  |    |
  | 0  |
  |    |
  +----+
|     |
  |    |
  | file # | hi |
  +--------+
|        |
  | 125 |
  +----+
|      |
  | 126 |
  +----+
|      |
  | 127 |
  +----+
```

Figure 5-17 File Management Subsystem File Sector Format

The FMS uses the file # to verify file integrity. The file # is a redundant piece of information. The file number field contains the value of the directory position of that file. If a mismatch occurs between the file's directory position, and the file number as contained in each sector, then the DFM will generate the error $A4.

The forward pointer field contains the 10-bit value for the diskette sector number of the next sector of the file. The pointer equals zero for the last sector of a file.

The S bit indicates whether or not the sector is a "short sector" (a sector containing fewer than 125 data bytes). S is equal to 1 when the sector is short.
The byte-count field contains the number of data bytes in the sector.

Non-CIO I/O

Some portions of the I/O subsystem are accessed independently of the Central I/O Utility (CIO); this section discusses those areas.

Resident Device Handler Vectors

All of the OS ROM resident device handlers can be accessed via sets of vectors that are part of the OS ROM. These vectors increase the speed of I/O operations that utilize fixed device assignments, such as output to the Display Handler. For each resident Handler there is a set of vectors ordered as shown below:

```
+---------------+ +- OPEN --+ +0
+---------------+ +- CLOSE --+ +2
+---------------+ +- GET BYTE --+ +4
+---------------+ +- PUT BYTE --+ +6
+---------------+ +- GET STATUS --+ +8
+---------------+ +- SPECIAL --+ +10
+---------------+ +- JMP --+ +12
+---------------+ +- INIT --+
+---------------+ +- SPARE --+
+---------------+ +- BYTE --+
```

Figure 5-18  Resident Device Handler Vectors

See Section 9 for a detailed description of the data interface for each of these Handler entry points.

Each of the vectors contains the address (lo, hi) of the Handler entry point minus 1. A technique similar to the one shown below is required to access the desired routines:

OPERATING SYSTEM CD16555 -- Section 5
VTBASE=$E400 ; BASE OF VECTOR TABLE.
LDX #xx ; OFFSET TO DESIRED ROUTINE.
LDA data
JSR GOVEC ; SEND DATA TO ROUTINE.

LDX #yy ; OFFSET TO DIFFERENT ROUTINE.
JSR GOVEC ; GET DATA FROM ROUTINE.
STA data

GOVEC TAY ; SAVE REGISTER A.
LDA VBASE+1,X ; ADDRESS MSB TO STACK.
PHA
LDA VBASE,X ; ADDRESS LSB TO STACK.
PHA
TYA ; RESTORE REGISTER A.
RTS ; JUMP TO ROUTINE.

The JMP INIT slot in each set of vectors jumps to the Handler initialization entry (not minus 1).

The base address of the vector set for each of the resident handlers is shown below:

Screen Editor (E:) E400.
Display Handler (S:) E410.
Keyboard Handler (K:) E420.
Printer Handler (P:) E430.
Cassette Handler (C:) E440.

The resident diskette Handler is not CIO-compatible, so its interface does not use a vector set.

Resident Diskette Handler

The resident Diskette Handler (not to be confused with the Disk File Manager) is responsible for all physical accesses to the diskette. The unit of data transfer for this Handler is a single diskette sector containing 128 data bytes.

Communication between you and the Diskette Handler is effected using the system’s Device Control Block (DCB), that is also used for Handler/SIO communication (see Section 9). The DCB is 12 bytes long. Some bytes are user-alterable and some are for use by the Diskette Handler and/or the Serial I/O Utility (SIO). You supply the required DCB parameters and then do a JSR DSKINV [E453].
Each of the DCB bytes will now be described, and the system-equate file name for each will be given.

SERIAL BUS ID -- DDEVIC [0300]

The Diskette Handler sets up this byte to contain the Serial Bus ID for the drive to be accessed. It is not user-alterable.

DEVICE NUMBER -- DUNIT [0301]

You set up this byte to contain the disk drive number to be accessed (1 - 4).

COMMAND BYTE -- DCOMND [0302]

You set up this byte to contain the disk device command to be performed.

STATUS BYTE -- DSTATS [0303]

This byte contains the status of the command upon return to the caller. See Appendix C for a list of the possible status codes.

BUFFER ADDRESS -- DBUFL0 [0304] and DBUFI [0305]

This 2-byte pointer contains the address of the source or destination of the diskette sector data. You need not supply an address for the disk status command. The Disk Handler will obtain the status and insert the address of the status buffer into this field.

DISK TIMEOUT VALUE -- DTIMLO [0306]

The Handler supplies this timeout value (in whole seconds) for use by SIO.

BYTE COUNT -- DBYTL0 [0308] and DBYTHI [0309]

This 2-byte counter indicates the number of bytes transferred to or from the disk as a result of the most recent command, and is set up by the Handler.

SECTOR NUMBER -- DAUX1 [030A] and DAUX2 [030B]

This 2-byte number specifies the diskette sector number (1 - 720) to read or write. DAUX1 contains the least significant byte, and
DAUX2 contains the most significant byte.

Diskette Handler Commands

There are five commands supported by the Diskette Handler:

GET SECTOR (PUT SECTOR —*** not supported by current handler ***)
PUT SECTOR WITH VERIFY
STATUS REQUEST
FORMAT DISK

GET SECTOR (Command byte = $52)

The Handler reads the specified sector to your buffer and returns the operation status. You set the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $52.
DEVICE NUMBER = disk drive number (1-4).
BUFFER ADDRESS = pointer to your 128-byte buffer.
SECTOR NUMBER = sector number to read.

Upon return from the sector, several of the other DCB parameters will have been altered. The STATUS BYTE will be the only parameter of interest to you, however.

PUT SECTOR (Command byte = $50)

*** Not supported by current Handler ***
(But can be accessed through SIO directly.)

The Handler writes the specified sector from your buffer and returns the operation status. You set the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $50.
DEVICE NUMBER = disk drive number (1-4).
BUFFER ADDRESS = pointer to your 128-byte buffer.
SECTOR NUMBER = sector number to write.

Upon return from the operation, several of the other DCB parameters will have been altered. The STATUS BYTE will be the only one of interest to you, however.

OPERATING SYSTEM  CO16555  -- Section 5
PUT SECTOR WITH VERIFY (Command Byte = $57)

The Handler writes the specified sector from your buffer and returns the operation status. This command differs from PUT SECTOR in that the diskette controller reads the sector data after writing to verify the write operation. Aside from the COMMAND BYTE value, the calling sequence is identical to PUT SECTOR.

STATUS REQUEST (Command byte = $53)

The Handler obtains a 4-byte status from the diskette controller and puts it in system location DVSTAT [02EA]. The operation status format is shown below:

```
 7 o
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1 command stat. 1          DVSTAT + 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1 hardware stat. 1        + 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1 timeout 1              + 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 1 (unused) 1             + 3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5-19. DVSTAT 40-Byte Operation Status Format

The command status contains the following status bits:

Bit 0 = 1 indicates an invalid command frame was received.
Bit 1 = 1 indicates an invalid data frame was received.
Bit 2 = 1 indicates that a PUT operation was unsuccessful.
Bit 3 = 1 indicates that the diskette is write protected.
Bit 4 = 1 indicates active/standby.

The hardware status byte contains the status register of the INS1771-1 Floppy Diskette Controller chip used in the diskette controller. See the documentation for that chip to obtain information relating to the meaning of each bit in the byte.

The timeout byte contains a controller-provided maximum timeout value (in seconds) to be used by the Handler.

You set the following DCB parameters prior to calling the Diskette Handler:

- COMMAND BYTE = $53.
- DEVICE NUMBER = disk drive number (1-4).

Upon return from the operation, several of the other DCB parameters will have been altered. The STATUS BYTE will be the only one of
interest to you, however.

FORMAT DISK (Command Byte = $21)

The Handler commands the diskette controller to format the entire diskette and then to verify it. All bad sector numbers (up to a maximum of 63) are returned and put in the supplied buffer, followed by two bytes of all 1's ($FFFF). You set up the following DCB parameters prior to calling the Diskette Handler:

COMMAND BYTE = $21.

DEVICE NUMBER = disk drive number (1-4).

BUFFER ADDRESS = pointer to your 128-byte buffer.

Upon return, you might be interested in the following DCB parameters:

STATUS BYTE = status of operation.

BYTE COUNT = number of bytes of bad sector information in your buffer, not including the $FFFF terminator. If there are no bad sectors, the count will equal zero.

Serial Bus I/O

Input/Output to devices other than the keyboard, the screen, and the ATARI Computer controller port devices, must utilize the Serial I/O bus. This bus contains data, control, and clock lines to be used to allow the computer to communicate with external devices on this "daisychained" bus. Every device on the bus has a unique identifier and will respond only when directly addressed.

The resident system provides a Serial I/O Utility (SIO), that provides a standardized high-level program interface to the bus. SIO is utilized by the resident Diskette, Printer, and Cassette handlers, and is intended to be used by nonresident handlers (see Section 9), or by applications, as well. For a detailed description of the program/SIO interface and for a detailed bus specification refer to Section 9.
Section 6 describes system actions for the various interrupt causing events, defines the many RAM vectors and provides recommended procedures for dealing with interrupts.

The 6502 microcomputer processes three general interrupt types: chip-reset, nonmaskable interrupts (NMI) and maskable interrupts (IRQ). The IRQ interrupt type can be enabled and disabled using the 6502 CLI and SEI instructions. The NMI type cannot be disabled at the processor level; but the NMI interrupts other than SYSTEM.RESET key can be disabled at the ANTIC chip.

The system events that can cause interrupts are listed below:

- **chip-reset** - power-up

- **NMI** - Display list interrupt (unused by OS)
  - vertical-blank (50/60 Hz)
  - SYSTEM.RESET key

- **IRQ** - Serial bus output ready
  - Serial bus output complete
  - Serial bus input ready
  - Serial bus proceed line (unused by system)
  - Serial bus interrupt line (unused by system)
  - POKEY timers 1, 2 and 4
  - Keyboard key
  - [BREAK] key
  - 6502 BRK instruction (unused by OS)

Figure 6-1 List of System-Interrupt Events
The chip-reset interrupt is vectored via location FFFC to E477, where a JMP vector to the power-up routine is located. All NMI interrupts are vectored via location FFFA to the NMI interrupt service routine at E7B4, and all IRQ interrupts are vectored via location FFFE to the IRQ interrupt service routine at E6F3; at that point the cause of the interrupt must be determined by a series of tests. For some of the events there are built in monitor actions and for other events the corresponding interrupts are disabled or ignored. The system provides RAM vectors so that you can intercept interrupts when necessary.

CHIP-RESET

The OS generates chip-reset in response to a power-up condition. The system is completely initialized (see Section 7).

NONMASKABLE INTERRUPTS

When an NMI interrupt occurs, control is transferred through the ROM vector directly to the system NMI interrupt service routine. A cause for the interrupt is determined by examining hardware register NMIST [D40F]. The NMI makes a jump through the global RAM vector VDSLST [0200] if a display list interrupt is pending. The OS does not use display list interrupts, so VDSLST is initialized to point to an RTI instruction, and you must not change it before VDSLST generates a display interrupt.

If the interrupt is not a display-list interrupt, then a test is made to see if it is a [SYSTEM.RESET] key interrupt. If so, then a jump is made to the system reset initialization routine (see Section 7 for details of system reset initialization).

If the interrupt is neither a display list interrupt nor a [SYSTEM.RESET] key interrupt; then it is assumed to be a vertical-blank (VBLANK) interrupt, and the following actions occur:

Registers A, X and Y are pushed to the stack.

The interrupt request is cleared (NMIRES [D40F]).

A jump is made through the "immediate" vertical-blank global RAM vector VVBLKI [0222] that normally points to the Stage 1 VBLANK processor.

The following actions occur assuming that you have not changed VVBLKI.

OPERATING SYSTEM CO16555 -- Section 6
The stage 1 VBLANK processor is executed.

The OS tests to see if a critical code section has been interrupted. If so, then all registers are restored, and an RTI instruction returns from the interrupt to the critical section. A critical section is determined by examining the CRITIC flag [0042], and the processor I bit. If either are set, then the interrupted section is assumed to be critical.

If the interrupt was not from a critical section, then the stage 2 VBLANK processor is executed.

The OS then jumps through the "deferred" vertical-blank global RAM vector VBLKD [0224], that normally points to the VBLANK exit routine.

The following actions occur assuming that you have not changed VVBLKD.

o The 6502 A, X and Y registers are restored.

o An RTI instruction is executed.

NOTE: You can alter the deferred and immediate VBLANK RAM vectors, but still enable normal system processes; or restore original vectors without having to save them. The instruction at E45F is a JMP to the stage 1 VBLANK processor; the address at [E460,2] is the value normally found in VVBLKI. The instruction at E462 is a JMP to the VBLANK exit routine; the address at [E463,2] is the value normally found in VVBLKD. These ROM vectors to stage 1 VBLANK processor and to the VBLANK exit routine will accomplish your goal.

NOTE: Every VBLANK interrupt jumps through vector VVBLKI. Only VBLANK interrupts from noncritical code sections jump through vector VVBLKD.

Stage 1 VBLANK Process

The following stage 1 VBLANK processing is performed at every VBLANK interrupt:

The stage 1 VBLANK process increments the 3-byte frame counter RTCLK [0012-0014]; RTCLK+0 is the MSB and RTCLK+2 is the LSB. This counter wraps to zero when it overflows (every 77 hours or so), and continues counting.

The Attract mode variables are processed (see Appendix L, B10-12).

The stage 1 VBLANK process decrements the System Timer 1 CDTMV1 [0218,2] if it is nonzero; if the timer goes from
nonzero to zero then an indirect JSR is performed via CDTMA1 [0226,2].

Stage 2 VBLANK Process

The stage 2 VBLANK processing performs the following for those VBLANK interrupts that do not interrupt critical sections:

The stage 2 VBLANK process clears the 6502 processor I bit. This enables the IRQ interrupts.

The stage 2 VBLANK process updates various hardware registers with data from the OS data base, as shown below:

<table>
<thead>
<tr>
<th>Data Base Item</th>
<th>Hardware Register</th>
<th>Reason for Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDLSTH [0231]</td>
<td>DLISTH [D403]</td>
<td>Display list start</td>
</tr>
<tr>
<td>SDLSTL [0230]</td>
<td>DLISTL [D402]</td>
<td></td>
</tr>
<tr>
<td>SDMCTL [022F]</td>
<td>DMACTL [D400]</td>
<td></td>
</tr>
<tr>
<td>CHBAS [02F4]</td>
<td>CHBASE [D409]</td>
<td></td>
</tr>
<tr>
<td>CHACT [02F3]</td>
<td>CHACTL [D401]</td>
<td></td>
</tr>
<tr>
<td>GPRIOR [026F]</td>
<td>PRIOR [D01B]</td>
<td></td>
</tr>
<tr>
<td>COLOR1 [02C5]</td>
<td>COLPF1 [D017]</td>
<td></td>
</tr>
<tr>
<td>COLOR2 [02C6]</td>
<td>COLPF2 [D018]</td>
<td></td>
</tr>
<tr>
<td>COLOR3 [02C7]</td>
<td>COLPF3 [D019]</td>
<td></td>
</tr>
<tr>
<td>COLOR4 [02C8]</td>
<td>COLBK [D01A]</td>
<td></td>
</tr>
<tr>
<td>PCOLR0 [02C0]</td>
<td>COLPM0 [D012]</td>
<td></td>
</tr>
<tr>
<td>PCOLR1 [02C1]</td>
<td>COLPM1 [D013]</td>
<td></td>
</tr>
<tr>
<td>PCOLR2 [02C2]</td>
<td>COLPM2 [D014]</td>
<td></td>
</tr>
<tr>
<td>PCOLR3 [02C3]</td>
<td>COLPM3 [D015]</td>
<td></td>
</tr>
<tr>
<td>Constant = 8</td>
<td>CONSOL [D01F]</td>
<td>Console speaker off.</td>
</tr>
</tbody>
</table>

The stage 2 VBLANK process decrements the System Timer 2 CDTMV2 [021A,2] if it is nonzero; if the timer goes from nonzero to zero, then an indirect JSR is performed through CDTMA2 [0228,2].

The stage 2 VBLANK process decrements System Timers 3, 4 and 5 if they are nonzero; the corresponding flags are set to zero for each timer that changes from nonzero to zero.
A character is read from the POKEY keyboard register and stored in CH [02FC], if auto repeat is active.

The stage 2 VBLANK process decrements the keyboard debounce counter if it is not equal to zero, and if no key is pressed.

The stage 2 VBLANK process processes the keyboard auto repeat (see Appendix L, EB).

The stage 2 VBLANK process reads game controller data from the hardware to the RAM data base, as shown below:

<table>
<thead>
<tr>
<th>Hardware Register</th>
<th>Data Base Item</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA [D300]</td>
<td>STICK0 [0278]</td>
<td>Joysticks and</td>
</tr>
<tr>
<td></td>
<td>STICK1 [0279]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIGO [027C]</td>
<td>Paddle Controllers</td>
</tr>
<tr>
<td></td>
<td>PTRIG1 [027D]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIG2 [027E]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIG3 [027F]</td>
<td></td>
</tr>
<tr>
<td>PORTB [D301]</td>
<td>STICK2 [027A]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STICK3 [027B]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIG4 [0280]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIG5 [0281]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIG6 [0282]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTRIG7 [0283]</td>
<td></td>
</tr>
<tr>
<td>POT 0 [D200]</td>
<td>PADDL0 [0270]</td>
<td>Paddle Controllers</td>
</tr>
<tr>
<td>POT 1 [D201]</td>
<td>PADDL1 [0271]</td>
<td></td>
</tr>
<tr>
<td>POT 2 [D202]</td>
<td>PADDL2 [0272]</td>
<td></td>
</tr>
<tr>
<td>POT 3 [D203]</td>
<td>PADDL3 [0273]</td>
<td></td>
</tr>
<tr>
<td>POT 4 [D204]</td>
<td>PADDL4 [0274]</td>
<td></td>
</tr>
<tr>
<td>POT 5 [D205]</td>
<td>PADDL5 [0275]</td>
<td></td>
</tr>
<tr>
<td>POT 6 [D206]</td>
<td>PADDL6 [0276]</td>
<td></td>
</tr>
<tr>
<td>POT 7 [D207]</td>
<td>PADDL7 [0277]</td>
<td></td>
</tr>
<tr>
<td>TRIG0 [D001]</td>
<td>STRING0 [0284]</td>
<td>Joystick triggers.</td>
</tr>
<tr>
<td>TRIG1 [D002]</td>
<td>STRING1 [0285]</td>
<td></td>
</tr>
<tr>
<td>TRIG2 [D003]</td>
<td>STRING2 [0286]</td>
<td></td>
</tr>
<tr>
<td>TRIG3 [D004]</td>
<td>STRING3 [0287]</td>
<td></td>
</tr>
</tbody>
</table>
MASKABLE INTERRUPTS

An IRQ interrupt causes control to be transferred through the immediate IRQ global RAM vector VIMIRQ [0216]. Ordinarily this vector points to the system IRQ Handler. The Handler performs these following actions:

The IRQ Handler determines a cause for the interrupt by examining the IRQST [D20E] register and the PIA status registers PACTL [D302] and PBCTL [D303]. The interrupt status bit is cleared when it is found. One interrupt event is cleared and processed for each interrupt-service entry. If multiple IRQs are pending, then a separate interrupt will be generated for each pending IRQ, until all are serviced.

The system IRQ interrupt service routine deals with each of the possible IRQ causing events, in the following ways:

- The 6502 A register is pushed to the stack.
- If the interrupt is due to serial I/O bus output ready, then clear the interrupt and jump through global RAM vector VSEROR [020C].
- If the interrupt is due to serial I/O bus input ready, then clear the interrupt and jump through global RAM vector VSERIN [020A].
- If the interrupt is due to serial I/O bus output complete, then clear the interrupt and jump through global RAM vector VSEROC [020E].
- If the interrupt is due to POKEY timer #1, then clear the interrupt and jump through global RAM vector VTIMR1 [0210].
- If the interrupt is due to POKEY timer #2, then clear the interrupt and jump through global RAM vector VTIMR2 [0212].
- If the interrupt is due to POKEY timer #4, then clear the interrupt. The service routine contains a bug, and falls into the following test.
- If pressing a keyboard key caused the interrupt (other than [BREAK], [START], [OPTION], or [SELECT]); then clear the interrupt and jump through global RAM vector VKEYBD [0208].
- If pressing the [BREAK] key caused the interrupt; then clear the interrupt. Set the BREAK flag BRKKEY [0011] to zero, proceed to clear the following:
  - Start/stop flag SSFLAG [02FF]
  - Cursor inhibit flag CRSINH [02F0]
  - Attract mode flag ATRACT [004D]

OPERATING SYSTEM CO16555 -- Section 6 107
Return from the interrupt after restoring the 6502 A register from the stack.

- If the interrupt is due to the serial I/O bus proceed line; then clear the interrupt, and jump through global RAM vector VPRCED [0202].

- If the interrupt is due to the serial I/O bus interrupt line, then clear the interrupt and jump through global RAM vector VINTER [0204].

- If the interrupt is due to a 6502 BRK instruction, then jump through global RAM vector VBREAK [0206].

- If none of the above, restore the 6502 A register and return from the interrupt (RTI).

**INTERRUPT INITIALIZATION**

The interrupt subsystem completely reinitializes itself whenever the system is powered up or the [SYSTEM. RESET] key is pressed. The OS clears the hardware registers, and sets the interrupt global RAM vectors to the following configurations:

<table>
<thead>
<tr>
<th>Vector</th>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDSLST</td>
<td>NMI</td>
<td>RTI -- ignore interrupt.</td>
</tr>
<tr>
<td>VVBLKI</td>
<td>&quot;</td>
<td>System stage 1 VBLANK.</td>
</tr>
<tr>
<td>CDTMA1</td>
<td>&quot;</td>
<td>SIO timeout timer.</td>
</tr>
<tr>
<td>CDTMA2</td>
<td>&quot;</td>
<td>No system function.</td>
</tr>
<tr>
<td>VVBLKD</td>
<td>&quot;</td>
<td>System return from interrupt.</td>
</tr>
<tr>
<td>VIMIRQ</td>
<td>IRQ</td>
<td>System IRQ processor.</td>
</tr>
<tr>
<td>VSEROR</td>
<td>&quot;</td>
<td>SIO.</td>
</tr>
<tr>
<td>VSERIN</td>
<td>&quot;</td>
<td>SIO.</td>
</tr>
<tr>
<td>VSEROC</td>
<td>&quot;</td>
<td>SIO.</td>
</tr>
<tr>
<td>VTIMR1</td>
<td>&quot;</td>
<td>PLA,RTI -- ignore interrupt.</td>
</tr>
<tr>
<td>VTIMR2</td>
<td>&quot;</td>
<td>PLA,RTI -- ignore interrupt.</td>
</tr>
<tr>
<td>VTIMR4</td>
<td>&quot;</td>
<td>*** doesn't matter ***</td>
</tr>
<tr>
<td>VKEYBD</td>
<td>&quot;</td>
<td>System keyboard interrupt handler.</td>
</tr>
<tr>
<td>VPRCED</td>
<td>&quot;</td>
<td>PLA,RTI -- ignore interrupt.</td>
</tr>
<tr>
<td>VINTER</td>
<td>&quot;</td>
<td>PLA,RTI -- ignore interrupt.</td>
</tr>
<tr>
<td>VBREAK</td>
<td>BRK</td>
<td>PLA,RTI -- ignore interrupt.</td>
</tr>
</tbody>
</table>

Figure 6-2  Interrupt RAM Vector Initialization
System initialization sets the interrupt enable status as follows:

- **NMI**: VBLANK enabled, display list disabled.
- **IRQ**: [BREAK] key and data key interrupts enabled, all others disabled.

**SYSTEM TIMERS**

The OS contains five general purpose software timers, plus an OS-supported frame counter. The timers are 2 bytes in length (lo,hi) and the frame counter RTCLOK [0012] is three bytes in length (hi,mid,lo). The timers count downward from any nonzero value to zero. Upon reaching zero, they either clear an associated flag, or JSR through a RAM vector. The frame counter counts upward, wrapping to zero when it overflows.

The following table shows the timers and the frame counter characteristics:

<table>
<thead>
<tr>
<th>Timer Name</th>
<th>Flag/Vector</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDTMV2 [021A]</td>
<td>CDTMA2 [0228]</td>
<td>2-byte vector</td>
</tr>
<tr>
<td>CDTMV3 [021C]</td>
<td>CDTMF3 [022A]</td>
<td>1-byte flag</td>
</tr>
<tr>
<td>CDTMV4 [021E]</td>
<td>CDTMF4 [022C]</td>
<td>1-byte flag</td>
</tr>
<tr>
<td>CDTMV5 [0220]</td>
<td>CDTMF5 [022E]</td>
<td>1-byte flag</td>
</tr>
<tr>
<td>* RTCLOK [0012]</td>
<td></td>
<td>3-byte frame counter.</td>
</tr>
</tbody>
</table>

* These two timers are maintained as part of every VBLANK interrupt (stage 1 process). The other timers are subject to the critical section test (stage-2 process), that can defer their updating to a later VBLANK interrupt.

**USAGE NOTES**

This subsection describes the techniques you need to know in order to utilize interrupts in conjunction with the operating system.
POKEY Interrupt Mask

ANTIC (display-list and vertical-blank) and PIA (interrupt and proceed lines) interrupts can be masked directly (see the Hardware Manual). However, eight bits of a single byte IRGEN [D20E] mask the POKEY interrupts ([BREAK] key, data key, serial input ready, serial output ready, serial output done and timers 1, 2 and 4).

IRGEN is a write-only register. Thus, we must maintain a current value of that register in RAM in order to update individual mask bits selectively, while not changing other bits. The name of the variable used is POKMSK [0010], and it is used as shown in the examples below:

; EXAMPLE OF INTERRUPT ENABLE
SEI ; TO AVOID CONFLICT WITH IRQ ...
LDA POKMSK ; ... PROCESSOR WHICH ALTERS VAR.
ORA #$xx ; ENABLE BIT(S).
STA POKMSK
STA IRGEN ; TO HARDWARE REG TOO.
CLI

; EXAMPLE OF INTERRUPT DISABLE
SEI ; TO AVOID CONFLICT WITH IRQ ...
LDA POKMSK ; ... PROCESSOR WHICH ALTERS VAR.
AND #$FF-xx ; DISABLE BIT(S).
STA POKMSK
STA IRGEN ; TO HARDWARE REGISTER TOO.
CLI

Figure 6-3 POKEY Interrupt Mask Example

Note that the OS IRQ service routine uses and alters POKMSK, so alterations to the variable must be done with interrupts inhibited. If done at the interrupt level there is no problem, as the I bit is already set; if done at a background level then the SEI and CLI instructions should be used as shown in the examples.

Setting Interrupt and Timer Vectors

Because vertical-blank interrupts are generally kept enabled so that the frame counter RTCLOK is maintained accurately, there is a problem with setting the VBLANK vectors (VVBLKI and VVBLKD) or the timer values (CDTMVI through CDTMV5) directly. A VBLANK interrupt could occur when only one byte of the two-byte value had been updated, leading to undesired consequences. For this reason,
the SETBV [E45F] routine is provided to perform the desired update in safe manner. The calling sequence is shown below:

\[ \text{A} = \text{update item indicator} \]
\[ \text{1 - 5 for timers 1 - 5.} \]
\[ \text{6 for immediate VBLANK vector VVBLKI.} \]
\[ \text{7 for deferred VBLANK vector VVBLKD.} \]
\[ \text{X = MSB of value to store.} \]
\[ \text{Y = LSB of value to store.} \]

\[ \text{JSR SETBV} \]

The A, X and Y registers can be altered. The display list interrupt will always be disabled on return, even if enabled upon entry.

It is possible to fully process a vertical-blank interrupt during a call to this routine.

When working with the System Timers, the vectors for timers 1 and 2 and the flags for timers 3, 4 and 5 should be set while the associated timer is equal to zero, then the timer should be set to its (nonzero) value.

Stack Content at Interrupt Vector Points

The following table shows the stack content at every one of the RAM interrupt vector points:
RAM STACK CONTENT

<table>
<thead>
<tr>
<th>INTERRUPT VECTOR</th>
<th>DESCRIPTION</th>
<th>OS RETURN CONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDSLST [0200]</td>
<td>Display list</td>
<td>return, P</td>
</tr>
<tr>
<td>VVBLKI [0222]</td>
<td>VBLANK immediate</td>
<td>return, P, A, X, Y</td>
</tr>
<tr>
<td>CDTMA1 [0226]</td>
<td>System Timer 1</td>
<td>return, P, A, X, Y, return</td>
</tr>
<tr>
<td>CDTMA2 [0228]</td>
<td>System Timer 2</td>
<td>return, P, A, X, Y, return</td>
</tr>
<tr>
<td>VVBLKD [0224]</td>
<td>VBLANK defer.</td>
<td>return, P, A, X, Y</td>
</tr>
<tr>
<td>VIMIRQ [0216]</td>
<td>IRQ immediate</td>
<td>return, P</td>
</tr>
<tr>
<td>VSEROR [020C]</td>
<td>Serial out ready</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VSERIN [020A]</td>
<td>Serial in ready</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VSEROC [020E]</td>
<td>Serial out compare</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VTIMR1 [0210]</td>
<td>POKEY timer 1</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VTIMR2 [0212]</td>
<td>POKEY timer 2</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VTIMR4 [0214]</td>
<td>POKEY timer 4</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VKKEYBD [0208]</td>
<td>Keyboard data</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VPRSED [0202]</td>
<td>Serial proceed</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VINTER [0204]</td>
<td>Serial interrupt</td>
<td>return, P, A</td>
</tr>
<tr>
<td>VBREAK [0206]</td>
<td>BRK instruction</td>
<td>return, P, A</td>
</tr>
</tbody>
</table>

Figure 6-4  Interrupt and Timer Vector RAM Stack Content Table

* The OS initializes these entries at power-up. Improperly changing these vectors will alter system performance.

Miscellaneous Considerations

The following paragraphs list a set of miscellaneous considerations for the writer of an interrupt service routine.

Restrictions on Clearing of "I" Bit

Display list, immediate vertical-blank and System Timer #1 routines should not clear the 6502 I bit. If the NMI leading to one of these routines occurred while an IRQ was being processed, then clearing the I bit will cause the IRQ to re-interrupt with an unknown result.

The OS VBLANK processor carefully checks this condition after the stage 1 process and before the stage 2 process.

Interrupt Process Time Restrictions

You should not write an interrupt routine that exceeds 400 msec. when added to the stage 1 VBLANK, if the serial I/O is being used. The SIO sets the CRITIC flag while serial bus I/O is in progress.
Interrupt Delay Due to "WAIT FOR SYNC"

Whenever a key is read from the keyboard, the Keyboard Handler sets WSYNC [D40A] repeatedly while generating the audible click on the console speaker. A problem occurs when interrupts are generated during the wait-for-sync period; the processing of such interrupts will be delayed by one horizontal scan line. This condition cannot be prevented. You can work around the condition by examining the line count VCOUNT [D40B] and delaying interrupt processing by one line when no WSYNC delay has occurred.

FLOWCHARTS

The following pages contain process flowcharts showing the main events that occur in the NMI and IRQ interrupt processes.
NMI INTERRUPT PROCESS

1. ENTER
2. DISPLAY LIST?
   - Y: VDSLST → RTI
   - N: PUSH REG A TO STACK
3. VERTICAL BLANK?
   - N: RESET
   - Y: PUSH X & Y, CLEAR STATUS
     → VVBLK!
     → STAGE 1
4. CRITICAL SECTION?
   - Y: CLEAR I BIT
     → STAGE 2
     → VVBLKD
   - N: RESTORE REGISTERS
     → RTI
Section 7 discusses the details of the power-up and system reset processes. The power-up process will be explained first, and then the system reset process will be explained in terms of its differences from the power-up process.

Both power-up (also called coldstart) and pressing [SYSTEM. RESET] (warmstart) will cause system initialization: In addition, there are vectors for these processes at E474 (system reset) and E477 (power-up) so that they can be user-initiated.

The power-up initialization process is a superset of the system reset initialization process. Power-up initializes both the OS and user RAM regions, whereas system reset initializes only the OS RAM region. In both cases, the OS calls the outer level software initialization entry points allowing the application to initialize its own variables.

Pressing the [SYSTEM. RESET] key produces an NMI interrupt. It does not perform a 6502 chip-reset. If the processor is locked up, the [SYSTEM. RESET] key cannot be sufficient to unlock it, and the system must have power cycled to clear the problem.

POWER-UP INITIALIZATION (COLDSTART) PROCEDURE

The OS performs the following functions in the order shown, as part of the power-up initialization process:

1. The following 6502 processor states are set:
   - IRQ interrupts are disabled using the SEI instruction.
   - The decimal flag is cleared using the CLD instruction.
   - The stack pointer is set to $FF.

2. The OS sets the warmstart flag WARMST [0008] to 0 (false).
3. The OS tests to see if a diagnostic cartridge is in the A slot:
   o Cartridge address BFFC = 00?
   o The memory at BFFC is not RAM?
   o Bit 7 of the byte at BFFD = 1?

   If all of the above tests are true, then control is passed to the diagnostic cartridge via the vector at BFFE. No return is expected.

4. The OS determines the lowest memory address containing non-RAM, by testing the first byte of every 4K "block" to see if the content can be complemented. If it can be complemented, then the original value is restored and testing continues. If it can't be complemented; then the content is assumed to be the first non-RAM address in the system. The MSB of the address is stored temporarily in TRAMSZ [0006].

5. Zero is stored to all of the hardware register addresses shown below (most of that aren't decoded by the hardware):
   
   D000 through DOFF
   D200 through D2FF
   D300 through D3FF
   D400 through D4FF

6. The OS clears RAM from location 0008, to the address determined in step 4, above.

7. The default value for the "noncartridge" control vector DOSVEC [000A] is set to point to the blackboard routine. At the end of initialization, control is passed through this vector if a cartridge does not take control.

8. The coldstart flag COLDST [0244] is set to -1 (local use).

9. The screen margins are set: left margin = 2, right margin = 39, for a 38 character physical line. The maximum line size of 40 characters can be obtained by setting the margins to 0 and 39. The OS insets the left margin because the two leftmost columns of the video picture on many television sets are not entirely visible on the screen.

10. The interrupt RAM vectors VDSLST [0200] through VVBLKD [0224] are initialized. See Section 6 for the initialization values.

11. Portions of the OS RAM are set to their required nonzero values as shown below:

The top of memory pointer MEMTOP [02E5] = the lowest non-RAM address (from step 4); MEMTOP will be altered later when the Screen Editor is opened in step 15.

The bottom of memory pointer MEMLO [02E7] = 0700; MEMLO can be changed later if there is either a diskette- or cassette-boot operation.

The following resident routines are called for initialization:

- Screen Editor
- Display Handler
- Keyboard Handler
- Printer Handler
- Cassette Handler
- Central I/O Monitor (CID)
- Serial I/O Monitor (SIO)
- Interrupt processor

The [START] key is checked, and if pressed, the cassette-boot request flag CKEY [004A] is set.

12. 6502 IRQ interrupts are enabled using the CLI instruction.

13. The device table HATABS [031A] is initialized to point to the resident handlers. See Section 9 for information relating to the Device Handler table.

14. The cartridge slot addresses for cartridges B and A are examined to determine if cartridges are inserted, if RAM does not extend into the cartridge address space.

   If the content of location 9FFC is zero, then a JSR is executed through the vector at 9FFE, thus initializing cartridge "B". The cartridge is expected to return.

   If the content of location BFFC is zero, then a JSR is executed through the vector at BFFE, thus initializing cartridge "A". The cartridge is expected to return.

15. IOCB #0 is set up for an OPEN of the Screen Editor (E) and the OPEN is performed. The Screen Editor will use the highest portion of RAM for the screen and will adjust MEMTOP accordingly. If this operation should fail, the entire initialization process is repeated.

16. A delay is effected to assure that a VBLANK interrupt has occurred. This is done so that the screen will be established before continuing.

17. If the cassette-boot request flag is set (see step 11 above), then a cassette-boot operation is attempted. See Section 10
for details of the cassette-boot operation.

18. If any of the three conditions stated below exists, an attempt is made to boot from the disk.

   There are no cartridges in the slots.
   
   Cartridge B is inserted and bit 0 of 9FFD is 1.
   
   Cartridge A is inserted and bit 0 of BFFD is 1.

   See Section 10 for details of the diskette-boot operation.

19. The coldstart flag COLDST is reset to indicate that the coldstart process went to completion.

20. The initialization process is now complete, and the controlling application is now determined via the remaining steps.

   If there is an A cartridge inserted and bit-2 of BFFD is 1, then a JMP is executed through the vector at BFFA.

   Or, if there is a B cartridge inserted and bit-2 of 9FFD is 1, then a JMP is executed through the vector at 9FFA.

   Or, a jump is executed through the vector DOSVEC that can point to the blackboard routine (default case), cassette booted software or diskette booted software. DOSVEC can be altered by the booted software as explained in Section 10.

SYSTEM RESET INITIALIZATION (WARMSTART) PROCEDURE

The functions listed below are performed, in the order shown, as part of the system reset initialization process:

A. Same as power-up step 1.

B. The warmstart flag WARMST [0008] is set to -1 (true).

C. Same as power-up steps 3 through 5.

D. OS RAM is zeroed from locations 0200-03FF and 0010-007F.

E. Same as power-up steps 9 through 16.

F. If a cassette-boot was successfully completed during the power-up initialization, then a JSR is executed through the vector CASINI [0002]. See Section 10 for details of the cassette-boot process.
G. Same as power-up step 18, except instead of booting the diskette software, a JSR is executed through the vector DOSINI [000C] if the diskette-boot was successfully completed during the Power-up initialization. See Section 10 for details of the diskette-boot process.

H. Same as power-up steps 19 and 20.

Note that the initialization procedures and main entries for all software entities are executed at every system reset as well as at power up (see steps 14, 17, 18, 20, F and G). If the user-supplied initialization/startup code must behave differently in response to system reset than it does to power-up, then the warmstart flag WARMST [0008] should be interrogated; WARMST = 0 means power-up entry, else system reset entry.
8 FLOATING POINT ARITHMETIC PACKAGE

This section describes the BCD floating point (FP) package that is resident in the OS ROM in both the models 400 and 800.

The floating point package maintains numbers internally as 6-byte quantities: a 5-byte (10 BCD digit) mantissa with a 1-byte exponent. BCD internal representation was chosen so that decimal division would not lead to the rounding errors typically found in binary representation implementations.

The package provides the following operations:

- ASCII to FP conversion.
- FP to ASCII conversion.
- Integer to FP conversion.
- FP to integer conversion.
- FP add, subtract, multiply, and divide.
- FP logarithm, exponentiation, and polynomial evaluation.
- FP zero, load, store, and move.

A floating point operation is performed by calling one of the provided routines (each at a fixed address in ROM) after having set one or more floating point pseudo registers in RAM. The result of the desired operation will also involve floating point pseudo registers. The primary pseudo registers are described below and their addresses given within the square brackets:
FRO [00D4] = 6-byte internal form of FP number.
FR1 [00E0] = 6-byte internal form of FP number.
FLPTR [00FC] = 2-byte pointer (lo, hi) to a FP.
INBUFF [00F3] = 2-byte pointer (lo, hi) to an ASCII text buffer.
CIX [00F2] = 1-byte index, used as offset to buffer pointed to by INBUFF.
LBUFF [0580] = result buffer for the FASC routine.

FUNCTIONS/CALLING SEQUENCES

Descriptions of these floating point routines assume that a pseudo register is not altered by a given routine. The numbers in square brackets [xxxx] are the ROM addresses of the routines.

ASCII to Floating Point Conversion (AFP)

Function: This routine takes an ASCII string as input and produces a floating point number in internal form.

Calling sequence:

INBUFF = pointer to buffer containing the ASCII representation of the number.
CIX = the buffer offset to the first byte of the ASCII number.

JSR AFP [D800]
BCS first byte of ASCII number is invalid

FRO = floating point number.
CIX = the buffer offset to the first byte after the ASCII number.

Algorithm: The routine takes bytes from the buffer until it encounters a byte that cannot be part of the number. The bytes scanned to that point are then converted to a floating point number. If the first byte encountered is invalid, the carry bit is set as a flag.

Floating Point to ASCII Conversion (FASC)

Function: This routine converts a floating point number from internal form to its ASCII representation.
Calling sequence:

FRO = floating point number.

JSR FASC [D8E6]

INBUFF = pointer to the first byte of the ASCII number. The last byte of the ASCII representation has the most significant bit (sign bit) set; no EOL follows.

Algorithm: The routine converts the number from its internal floating point representation to a printable form (ATASCII). The pointer INBUFF will point to part of LBUFF, where the result is stored.

Integer to Floating Point Conversion (IFP)

Function: This routine converts a 2-byte unsigned integer (0 to 65535) to floating point internal representation.

Calling sequence:

FRO = integer (FRO+0 = LSB, FRO+1 = MSB).

JSR IFP [D9AA]

FRO = floating point representation of integer.

Floating Point to Integer Conversion (FPI)

Function: This routine converts a positive floating point number from its internal representation to the nearest 2-byte integer.

Calling sequence:

FRO = floating point number.

JSR FPI [D9D2]

BCS FP number is negative or >= 65535.5

FRO = 2-byte integer (FRO+0 = LSB, FRO+1 = MSB).

Algorithm: The routine performs true rounding, not truncation, during the conversion process.
Floating Point Addition (FADD)

Function: This routine adds two floating point numbers and checks the result for out-of-range.

Calling sequence:

FRO = floating point number.
FR1 = floating point number.

JSR FADD [DA66]
BCS out-of-range result.

FRO = result of FRO + FR1.
FR1 is altered.

Floating Point Subtraction (FSUB)

Function: This routine subtracts two floating point numbers and checks the result for out-of-range.

Calling sequence:

FRO = floating point minuend.
FR1 = floating point subtrahend.

JSR FSUB [DA60]
BCS out-of-range result.

FRO = result of FRO - FR1.
FR1 is altered.

Floating Point Multiplication (FMUL)

Function: This routine multiplies two floating point numbers and checks the result for out-of-range.

Calling sequence:

FRO = floating point multiplier.
FR1 = floating point multiplicand.

JSR FMUL [DADB]
BCS out-of-range result.

FRO = result of FRO * FR1.
FR1 is altered.
Floating Point Division (FDIV)

Function: This routine divides two floating point numbers and checks for division by zero and for result out-of-range.

Calling sequence:

FRO = floating point dividend.
FR1 = floating point divisor.
JSR FDIV [DB28]
BCS out-of-range result or divisor is zero.

FRO = result of FRO / FR1.
FR1 is altered.

Floating Point Logarithms (LOG and LOG10)

Function: These routines take the natural or base 10 logarithms of a floating point number.

Calling sequence:

FRO = floating point number.
JSR LOG [DECD] for natural logarithm
or
JSR LOG10 [DED1] for base 10 logarithm
BCS negative number or overflow.

FRO = floating point logarithm.
FR1 is altered.

Algorithm: Both logarithms are first computed as base 10 logarithms using a 10 term polynomial approximation; the natural logarithm is computed by dividing the base 10 result by the constant LOG10(e).

The logarithm of a number Z is computed as follows:

F * (10 ** Y) = Z where 1 <= F < 10 (normalization).
L = LOG10(F) by 10 term polynomial approximation.
LOG10(Z) = Y + L.  LOG(Z) = LOG10(Z) / LOG10(e).

NOTE: This routine does not return an error if the number input is zero; the LOG10 result in this case is approximately -129.5, which is not useful.
Floating Point Exponentiation (EXP and EXP10)

Function: This routine exponentiates.

Calling sequence:

FRO = floating point exponent (Z).

or

JSR       EXP [DDCO] for e ** Z

JSR       EXP10 [DDCC] for 10 ** Z

BCS       overflow.

FRO = floating point result.
FR1 is altered.

Algorithm: Both exponentials are computed internally as base 10, with the base e exponential using the identity:

\[ e^x = 10^{x \cdot \log_{10}(e)} \]

The base 10 exponential is evaluated in two parts using the identity:

\[ 10^x = 10^{(I + F)} = (10^I) \times (10^F) \]  -- where I is the integer portion of x and F is the fraction.

The term \(10^F\) is evaluated using a polynomial approximation, and \(10^I\) is a straightforward modification to the floating point exponent.

Floating Point Polynomial Evaluation (PLYEVL)

Function: This routine performs an n degree polynomial evaluation.

Calling sequence:

\(X, Y = \text{pointer } (X = \text{LSB}) \text{ to list of FP coefficients } (A(i))\)
\(\text{ordered from high order to low order (six bytes per coefficient)}\).
\(A = \text{number of coefficients in list}\).
\(\text{FRO = floating point independent variable } (Z)\).

JSR       PLYEVL [DD40]

BCS       overflow or other error.

FRO = result of \(A(n) \times Z^n + A(n-1) \times Z^{n-1} + \ldots + A(1) \times Z + A(0)\).
FR1 is altered.

Algorithm: The polynomial \(P(Z) = \sum_{i=0}^{n} (A(i) \times Z^i)\) is computed using the standard method shown below:

\[ P(Z) = (\ldots (A(n) \times Z + A(n-1)) \times Z + \ldots + A(1)) \times Z + A(0) \]
Clear FRO (ZFRO)

Function: This routine sets the contents of pseudo register FRO to all zeros.

Calling sequence:

```
JSR ZFRO [DA44]
```

FRO = zero.

Clear Page Zero Floating Point Number (ZF1)

Function: This routine sets the contents of a zero-page floating point number to all zeroes.

Calling sequence:

```
X = Zero-page address of FP number to clear.
JSR ZF1 [DA46]
Zero-page FP number(X) = zero.
```

Load Floating Point Number to FRO (FLDOR and FLDO P)

Function: These routines load pseudo register FRO with the floating point number specified by the calling sequence.

Calling sequences:

```
X, Y = pointer (X = LSB) to FP number.
JSR FLDOR [DD89]
```

or

```
FLPTR = pointer to FP number.
JSR FLDO P [DD8D]
```

FRO = floating point number (in either case).
FLPTR = pointer to FP number (in either case).
Load Floating Point Number to FR1 (FLD1R and FLD1P)

Function: These routines load pseudo register FR1 with the floating point number specified by the calling sequence.

Calling sequences:

As in prior description, except the result goes to FR1 instead of FRO. FLD1R [DD98] and FLD1P [DD9C].

Store Floating Point Number From FRO (FSTOR and FSTOP)

Function: These routines store the contents of pseudo register FRO to the address specified by the calling sequence:

Calling sequence:

As in prior descriptions, except the floating point number is stored from FRO rather than loaded to FRO. FSTOR [DDA7] and FSTOP [DDAB].

Move Floating Point Number From FRO to FR1 (FMOVE)

Function: This routine moves the floating point number in FRO to pseudo register FR1.

Calling sequence:

```
JSR        FMOVE [DDB6]
```

FR1 = FRO  (FRO remains unchanged).

RESOURCE UTILIZATION

The floating point package uses the following RAM locations in the course of performing the functions described in this section:

- 00D4 through 00FF
- 057E through 05FF

All of these locations are available for program coding if your program does not call the floating point package.
IMPLEMENTATION DETAILS

Floating point numbers are maintained internally as 6-byte quantities, with 5 bytes (10 BCD digits) of mantissa and 1 byte of exponent. The mantissa is always normalized such that the most significant byte is nonzero (note "byte" and not "BCD digit").

The most significant bit of the exponent byte provides the sign for the mantissa; 0 for positive and 1 for negative. The remaining 7 bits of the exponent byte provide the exponent in excess 64 notation. The resulting number represents powers of 100 decimal (not powers of 10). This storage format allows the mantissa to hold 10 BCD digits when the value of the exponent is an even power of 10, and 9 BCD digits when the value of the exponent is an odd power of 10.

The implied decimal point is always to the immediate right of the first byte. An exponent less than 64 indicates a number less than 1. An exponent equal to or greater than 64 represents a number equal to or greater than 1.

Zero is represented by a zero mantissa and a zero exponent. To test for a result from any of the standard routines; test either the exponent or the first mantissa byte for zero.

The absolute value of floating point numbers must be greater than 10**-98, and less than 10**+98, or be equal to zero. There is perfect symmetry between positive and negative numbers with the exception that negative zero is never generated.

The precision of all computations is maintained at 9 or 10 decimal digits, but accuracy is somewhat less for those functions involving polynomial approximations (logarithm and exponentiation). Also, the problems inherent in all floating point systems are present here; for example: subtracting two very nearly equal numbers, adding numbers of disparate magnitude, or successions of any operation, will all result in a loss of significant digits. An analysis of the data range and the order of evaluation of expressions may be required for some types of applications.

The examples below compare floating point numbers with their internal representations, as an aid to understanding storage format. All numbers prior to this point have been expressed in decimal notation, but these examples will use hexadecimal notation. Note that 64 decimal (the excess number of the exponent) is 40 when expressed in hexadecimal:

Number: +0.02 = 2 * 10**-2 = 2 * 100**-1
Stored: 3F 02 00 00 00 00 (FP exponent = 40 - 1)

Number: -0.02 = -2 * 10**-2 = -2 * 100**-1
Stored: BF 02 00 00 00 00 (FP exponent = 80 + 40 - 1)
Number: +37.0 = 3.7 * 10**1 = 37 * 100**0
Stored: 40 37 00 00 00 00 (FP exponent = 40 + 0)

Number: -4.60312486 * 10**11 = -46.03... * 100**5
Stored: C5 46 03 01 24 86 (FP exponent = 80 + 40 + 5)

Number: 0.0
Stored: 00 00 00 00 00 00 (special case)
This section describes the interface requirements for a nonresident Device Handler that is to be accessed via the Central I/O utility (CIO). The Serial bus I/O utility (SIO) interface is defined for those handlers that utilize the Serial I/O bus.

The I/O subsystem is organized with three levels of software between you and your hardware: The CIO, the individual device handlers, and the SIO.

The CIO performs the following functions:

- Logical device name to Device Handler mapping (on OPEN).
- I/O Control Block (IOCB) maintenance.
- Logical record handling.
- User buffer handling.

The device handlers are below CIO. They perform the following functions:

- Device initialization on power-up and system reset.
- Device-dependent support of OPEN and CLOSE commands.
- Byte-at-a-time data input and output.
- Device-dependent special operations.
- Device-dependent command support.
- Device data buffer management.

The SIO is at the bottom level (for Serial I/O bus peripheral handlers). It performs the following functions:

- Control of all Serial bus I/O, conforming to the bus protocol.
- Bus operation retries on errors.
- Return of unified error statuses on error conditions.
A separate control structure is used for communication at each interface, as follows:

<table>
<thead>
<tr>
<th>Interface</th>
<th>Control Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>User/CIO</td>
<td>I/O Control Block (IOCB)</td>
</tr>
<tr>
<td>CIO/Handler</td>
<td>Zero-page IOCB (ZIOCB)</td>
</tr>
<tr>
<td>Handler/SIO</td>
<td>Device Control Block (DCB)</td>
</tr>
</tbody>
</table>
Where: ---- shows a control path.
***** shows the data structure required for a path.

Note the following:

1. The Keyboard/Display/Screen Editor handlers don't use SIO.
2. The Diskette Handler cannot be called directly from CIO.
3. The DCB is shown twice in the diagram.

Figure 9-1 I/O Subsystem Flow Diagram
DEVICE TABLE

The device table is a RAM-resident table that contains the single-character device name (e.g. K, D, C, etc.) and the handler address for each of the handlers known to CIO. The table is initialized to contain entries for the following resident handlers: Keyboard (K), Display (S), Screen Editor (E), Cassette (C), and Printer (P) at power-up and system reset. To install a new handler, some procedure must insert a device table entry after the table is initialized.

The table format is shown below:

```
+----------------+ -+
| device name    | +- one entry
+----------------+ -+
| handler vector |    +
+----------------+    +
| table address  |    +
+----------------+    +
| more            |
+----------------+    =
| entries         |
+----------------+    =
| zero fill to    |
+----------------+    =
| end of table    |
```

Figure 9-2 Device Table Format

This 38-byte table will hold a maximum of 12 entries, with the last 2 bytes being zero. CIO scans the table from the end to the beginning (high to low address); so the entry nearest the end of the table will take precedence in case of multiple occurrences of a device name.

The device name for each entry is a single ATASCII character, and the handler address points to the handler’s vector table, that will be described in the following section.

CIO/HANDLER INTERFACE

This section describes the interface between the Central I/O utility and the individual device handlers that are represented in the Device Table (as described in the preceding section).
Calling Mechanism

Each handler has a vector table as shown below:

```
+----------------+  +----------------+
 + OPEN vector   +  + OPEN vector   + (low address)
 +----------------+  +----------------+
 + CLOSE vector  +  + CLOSE vector  +
 +----------------+  +----------------+
 + GETBYTE vector+  + GETBYTE vector+
 +----------------+  +----------------+
 + PUTBYTE vector+  + PUTBYTE vector+
 +----------------+  +----------------+
 + GETSTAT vector+  + GETSTAT vector+
 +----------------+  +----------------+
 + SPECIAL vector +  + SPECIAL vector +
 +----------------+  +----------------+
 + JMP init code  +  + JMP init code  + (high address)
 +----------------+  +----------------+
```

Figure 9-3 Handler Vector Table

The device table entry for the handler points to the first byte of the vector table.

The first six entries in the table are vectors (lo, hi) that contain the address - 1 of the handler routine that handles the indicated function. The seventh entry is a 6502 JMP instruction to the handler initialization routine. CIO uses only the addresses contained in this table for handler entry. Each user/CIO command translates to one or more calls to one of the handler entries defined in the vector table.

The vector table provides the handler addresses for certain fixed functions to be performed to CIO. In addition, operation parameters also must be passed for most functions. Parameter passing is accomplished using the 6502 A, X, and Y registers and an IOCB in page 0 named ZIOCB [0020]. In general, register A is used to pass data, register X contains the index to the originating IOCB, and register Y is used to pass status information to CIO. The zero-page IOCB, is a copy of the originating IOCB; but in the course of processing some commands, CIO can alter the buffer address and buffer length parameters in ZIOCB, but not in the originating IOCB (see Section 5 for information relating to the originating IOCB).

See Appendix B for the standard status byte values to be returned to CIO in register Y.
The following sections describe the CIO/handler interface for each of the vectors in the handler vector table.

Handler Initialization

NOTE: This entry doesn't appear to have any function for nonresident handlers due to a bug in the current OS -- the device table is cleared in response to system reset as well as power-up. This prevents this entry point from ever being called. The rest of this section discusses the intended use of this entry point. Conformation would be in order to allow compatibility with possible corrected versions of the OS in the future.

The entry was to have been called on all occurrences of power-up and system reset; the handler is to perform initialization of its hardware and RAM data using a routine that assures proper processing of all CIO commands that follow.

Functions Supported

This section describes the functions associated with the first six vectors from the handler vector table. This section also presents a brief, device-independent description of the CIO/handler interface and recommended actions for each function vector.

OPEN

This entry is called in response to an OPEN command to CIO. The handler is expected to validate the OPEN parameters and perform any required device initialization associated with a device OPEN.

At handler entry, the following parameters can be of interest:

- \( X \) = index to originating IDCB.
- \( Y \) = $92 (status = function not implemented by handler).
- \( \text{ICDNOZ [0021]} \) = device number (1-4, for multiple device handlers).
- \( \text{ICBALZ/ICBAHZ [0024/0025]} \) = address of device/filename specification.
- \( \text{ICAX1Z/ICAX2Z [002A/002B]} \) = device-specific information.

The handler attempts to perform the indicated OPEN and indicates the status of the operation by the value of the \( Y \) register. The responsibility for checking for multiple OPENS to
the same device or file, where it is illegal, lies with the handler.

CLOSE

This vector table entry is called in response to a CLOSE command to CIO. The handler is expected to release any held resources that relate specifically to that device/filename, and for output files to:

1) send any data remaining in handler buffers to the device,
2) mark the end of file
3) update any associated directories, allocation maps, etc.

At handler entry, the following parameters can be of interest:

X = index to originating IOCTL.
Y = $92 (status = function not implemented by handler).
ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler attempts to perform the indicated CLOSE and indicates the status of the operation by the value of the Y register.

CIO releases the associated IOCTL after the handler returns, regardless of the operation status value.

GETBYTE

This vector table entry is called in response to a GET CHARACTERS or GET RECORD command to CIO. The handler is expected to return a single byte in the A register, or return an error status in the Y register.

At handler entry, the following parameters can be of interest:

X = index to originating IOCTL.
Y = $92 (status = function not implemented by handler).
ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler will obtain a data byte directly from the device or from a handler-maintained buffer and return to CIO with the byte in the A register and the operation status in the Y register.
Handlers that do not have short timeouts associated with the reading of data (such as the Keyboard and Cassette Handlers), must monitor the [BREAK] key flag BRKKEY [0011] and return with a status of $80 when a [BREAK] condition occurs. See Appendix L, E5; and Section 12 for a discussion of [BREAK] key monitoring.

CIO checks for reads from device/files that have not been opened or have been opened for output only; the handler will not be called in those cases.

PUTBYTE

This entry is called in response to a PUT CHARACTERS or PUT RECORD command to CIO. The handler is expected to accept a single byte in the A register or return an error status in the Y register.

At handler entry, the following parameters can be of interest:

- X = index to originating IOCB.
- Y = $92 (status = function not implemented by handler).
- A = data byte.

ICDNOZ [0021] = device number (1-4, for multiple device handlers).
ICAX1Z/ICAX2Z [002A/002B] = device-specific information.

The handler sends the data byte directly to the device, or to a handler-maintained buffer, and returns to CIO with the operation status in the Y register. If a handler-maintained buffer fills, the handler will send the buffered data to the device before returning to CIO.

CIO checks for WRITEs to device/files that have not been opened, or have been opened for input only. The handler will not be called in those cases.

Now that the normal operation of PUTBYTE has been defined, a special case must be added. Any handler that will operate within the environment of the ATARI 8K BASIC language interpreter has a different set of rules. Because BASIC can call the handler PUTBYTE entry directly, without going through CIO, the zero-page IOCB (ZIOCB) can or may not have a relation to the PUTBYTE call. Therefore, the handler must use the outer level IOCB to obtain any information that would normally be obtained from ZIOCB. Note also that the OPEN protection normally provided by CIO is bypassed (i.e. PUTBYTE to a non-OPEN device/file and PUTBYTE to a read-only OPEN).
GETSTAT

This entry is called in response to a GET STATUS command to CIO. The handler is expected to return four bytes of status to memory or return an error status in the Y register.

At handler entry, the following parameters can be of interest:

- X = index to originating IOCB. Y = $92 (status = function not implemented by handler).
- ICDNOZ [0021] = device number (1-4, for multiple device handlers).
- ICBALZ/ICBAHZ [0024/0025] = address of device/filename specification.

The handler gets device status information from the device controller and puts the status bytes in DVSTAT [02EA] through DVSTAT+3, and finally returns to CIO with the operation status in register Y.

The IOCB need not be opened nor closed in order for you to request CIO to perform a GET STATUS operation; the handler must check where there are restrictions. See Section 5 for a discussion of the CIO actions involved with a GET STATUS operation using both open and closed IOCB's, and note the impact of this operation on the use of the buffer address parameter.

SPECIAL

This handler entry is used to support all functions not handled by the other entry points, such as diskette file RENAME, display DRAW, etc. Specifically, if the IOCB command byte value is greater than $0D, then CIO will use the SPECIAL entry point. The handler must interrogate the command byte to determine if the requested operation is supported.

At handler entry, the following parameters can be of interest:

- X = index to originating IOCB.
- Y = $92 (status = function not implemented by handler).
- ICDNOZ [0021] = device number (1-4, for multiple device handlers).
- ICCMZ [0022] = command byte.
- ICBALZ/ICBALH [0024/0025] = buffer address.
The handler will perform the indicated operation, if possible, and return to CID with the operation status in register Y.

The IOC B need not be opened nor closed in order for you to request CID to perform a SPECIAL operation; the handler must check where there are restrictions. See Section 5 for a discussion of the CID actions involved with a SPECIAL operation using both open and closed IOC B's, and note the impact of this on the use of the buffer address parameter.

Error Handling

Error handling has been simplified somewhat by having CID handle outer level errors and having SID handle Serial bus errors, leaving the handler to process the remaining errors. These errors include:

- out-of-range parameters.
- [BREAK] key abort.
- Invalid command.
- Read after end of file.

The current handlers respond to errors using the following guidelines:

They keep the recovery simple (and therefore predictable and repeatable).

They Do not interact directly with you for recovery instructions.

They lose as little data as possible.

They make all attempts to maintain the integrity of file oriented device storage -- this involves the initial design of the structural elements as well as error recovery techniques.

Resource Allocation

Nonresident handlers needing code and/or data space in RAM should use the techniques listed below, to assure nonconflict with other parts of the OS, including other nonresident handlers.
Zero-Page RAM

Zero-page RAM has no spare bytes, and even if there were, there is no allocation scheme to support multiple program assignment of the spares. Therefore, the nonresident handler must save and restore the bytes of zero-page RAM it is going to use. The bytes to use must be chosen carefully, according to the following criteria:

The bytes cannot be accessed by an interrupt routine.

The bytes cannot be accessed by any noninterrupt code between the time the handler modifies the bytes and then restores the original values.

A simple save/restore technique would utilize the stack in a manner similar to that shown below:

```assembly
LDA COLCRS ; FOR EXAMPLE
PHA ; SAVE ON STACK.
LDA COLCRS+1
PHA

LDA HPOINT ; HANDLER'S POINTER.
STA COLCRS
LDA HPOINT+1
STA COLCRS+1

XXX (COLCRS),Y ; DO YOUR POINTER THING.

PLA ; RESTORE OLD DATA.
STA COLCRS+1
PLA
STA COLCRS
```

Note that the Display Handler or Screen Editor should not be called before restoring the original value of COLCRS, because COLCRS is a variable used by those routines.

Nonzero-Page RAM

There is no allocation scheme to support the assignment of fixed regions of nonzero-page RAM to any specific process, so the handler has three choices:

1. Make a dynamic allocation at initialization time by altering MEMLO [02E7].

2. Include the variables with the handler for RAM-resident handlers. This still involves altering MEMLO at the time the handler is booted.

3. If the handler replaces one of the resident handlers (by removing the resident handler's entry in the device table), then the new handler can use any RAM that the
formerly resident handler would have used.

Stack Space

In most cases, there are no restrictions on the use of the stack by a handler. However, if the handler plans to push more than a couple dozen bytes to the stack; then it should do a stack overflow test, and always leave stack space for interrupt processing.

HANDLER/SID INTERFACE

This section describes the interface between serial bus device handlers and the serial bus I/O utility (SID). SIO completely handles all bus transactions following the device-independent bus protocol. SIO is responsible for the following functions:

- Bus data format and timing from computer end.
- Error detection, retries and statuses.
- Bus timeout.
- Transfer of data between the bus and the caller's buffer.

Calling Mechanism

SIO has a single entry point SIDV [E459] for all operations. The device control block (DCB) [0300] contains all parameters passed to SIO. The DCB contains the following bytes:

**DEVICE BUS ID -- DDEVIC [0300]**

The bus ID of the device is set by the handler prior to calling SIO (see Appendix I).

**DEVICE UNIT # -- DUNIT [0301]**

This byte indicates that of n units of a given device type to access, and is set by the handler prior to calling SIO. This value usually comes from ICDNOZ. SIO accesses the bus device whose address is equal to the value of DDEVIC plus DUNIT minus 1 (the lowest unit number is normally equal to 1).

**DEVICE COMMAND -- DCOMND [0302]**

The handler sets this byte prior to calling SIO. It will be sent to the bus device as part of the command frame. See Appendix I for device command byte values.
DEVICE STATUS -- DSTATS [0303]

This byte is bidirectional. The handler will use DSTATS to indicate to SIO what to do after the command frame is sent and acknowledged. SIO will use it to indicate to the handler the status of the requested operation.

Prior to an SIO call:

\[
\begin{array}{c}
7 \\
+-------+ \\
!W!R! unused! \\
+-------+
\end{array}
\]

Where: W,R = 0,0 indicates no data transfer is associated with the operation.
0,1 indicates a data frame is expected from the device.
1,0 indicates a data frame is to be sent to the device.
1,1 is invalid.

After an SIO call:

\[
\begin{array}{c}
7 \\
+-------+ \\
! status code! \\
+-------+
\end{array}
\]

See Appendix C for the possible SIO operation status codes.

HANDLER BUFFER ADDRESS -- DBUFL/DBUFHI [0304/0305]

The handler sets this 2-byte pointer. It indicates the source or destination buffer for device data or status information.

DEVICE TIMEOUT -- DTIMLO [0306]

The handler sets this byte. It specifies the device timeout time in units of 64/60 of a second. For example, a count of 6 specifies a timeout of 6.4 seconds.

BUFFER LENGTH/BYTE COUNT -- DBYTL/DBYTHI [0308/0309]

The handler sets this 2-byte count for the current operation, and indicates the number of data bytes to be transferred into or out of the buffer. This parameter is not required if the STATUS byte W and R bits are both zero. These values indicate that no data transfer is to take place.

WARNING: There is a bug in SIO that causes incorrect actions when the last byte of a buffer is in a memory address ending in $FF, such as 13FF, 42FF, etc.
The handler sets these 2-bytes. The SIO includes them in the bus command frame; they have device-specific meanings.

Functions Supported

SIO does not examine the COMMAND byte it sends to the device, because all bus transactions are expected to conform to a universal protocol. The protocol includes three forms, stated below (as seen from the computer):

Send command frame.
Send command frame and send data frame.
Send command frame and receive data frame.

The values of the W and R bits in the status byte select the command form.

Error Handling

SIO handles most of the serial bus errors for the handler, as indicated below:

Bus timeout -- SIO provides a uniform command frame and data frame ACK byte timeout of 1/60 of a second — 0 / + 1/60. The handler specifies the maximum COMPLETE byte timeout value in DTIMLO.

Bus errors -- SIO detects and reports UART overrun and framing errors. The sensing of these errors in any received byte will cause the entire associated frame to be considered bad.

Data frame checksum error -- SIO validates the checksum on all received data frames and generates a checksum for all transmitted frames.

Invalid response from device -- In addition to the error conditions stated above, SIO checks that the ACK and COMPLETE responses are proper (ACK = $41 and COMPLETE = $43). ACK stands for acknowledge.

Bus operation retries -- SIO will attempt one complete command retry if the first attempt is not error free, where a complete command try consists of up to 14 attempts to send (and acknowledge) a command frame, followed by a single attempt to
receive the COMPLETE code and possibly a data frame.

NOTE: There is a bug in the retry logic for data writes, such that if the command frame is acknowledged by the controller, but the data frame is not acknowledged, then SIO will retry indefinitely.

Unified error status codes -- SIO provides device-independent error codes (see Appendix C).

SERIAL I/O BUS CHARACTERISTICS AND PROTOCOL

This section describes:

- The electrical characteristics of the ATARI 400 and ATARI 800 Home Computers serial bus
- The use of the bus to send bytes of data,
- The organization of the bytes as "frames" (records),
- The overall command sequences that utilize frames and response bytes to provide computer/peripheral communication.

Hardware/Electrical Characteristics

The ATARI 400 and the ATARI 800 Home Computers communicate with peripheral devices over a 19,200 baud asynchronous serial port. The serial port consists of a serial DATA OUT (transmission) line, a serial DATA IN (receiver) line and other miscellaneous control lines.

Data is transmitted and received as 8 bits of serial data (LSB sent first) preceded by a logic zero start bit and succeeded by a logic one stop bit. The serial DATA OUT is transmitted as positive logic (+4v = one/true/high, Ov = zero/false/low). The serial DATA OUT line always assumes its new state when the serial CLOCK OUT line goes high; CLOCK OUT then goes low in the center of the DATA OUT bit time.

An end view of the Serial bus connector at the computer or peripheral is shown below (the cable connectors would of course be a mirror image):
where: 1 = computer CLOCK IN.
2 = computer CLOCK OUT.
3 = computer DATA IN.
4 = GND.
5 = computer DATA OUT.
6 = GND.
7 = COMMAND-.
8 = MOTOR CONTROL.
9 = PROCEED-.
10 = +5v/READY.
11 = computer AUDIO IN.
12 = +12v.
13 = INTERRUPT-.

Figure 9-4  Serial Bus Connector Pin Descriptions

CLOCK IN is not used by the present OS and peripherals. This line can be used in future synchronous communications schemes.

CLOCK OUT is the serial bus clock. CLOCK OUT goes high at the start of each DATA OUT bit and returns to low in the middle of each bit.

DATA IN is the serial bus data line to the computer.

Pin 4 GND is the signal/shield ground line.

DATA OUT is the serial bus data line from the computer.

Pin 6 GND is the signal/shield ground line.

COMMAND- is normally high and goes low when a command frame is being sent from the computer.

MOTOR CONTROL is the cassette motor control line (high=on, low=off).

PROCEED- is not used by the present OS and peripherals; this line is pulled high.

+5v/READY indicates that the computer is turned on and ready. This line can also be used as a +5 volt supply of 50ma current rating for ATARI peripherals only.

AUDIO IN accepts an audio signal from the cassette.
+12V is a +12 volt supply of unknown current rating for ATARI peripherals only.

INTERRUPT- is not used by the present OS and peripherals; this line is pulled high.

There are no pin reassignments made in the Serial bus cable, so pin 3, the computer’s DATA IN line, is the peripheral’s data output line; and similarly for pin 5.

Serial Port Electrical Specifications

Peripheral input:

\[
\begin{align*}
V_{IH} &= 2.0 \text{v min.} \\
V_{IL} &= 0.4 \text{v max.}
\end{align*}
\]

\[
\begin{align*}
I_{IH} &= 20\text{ua. max.} @ V_{IH} = 2.0 \text{v} \\
I_{IL} &= 5\text{ua. max.} @ V_{IL} = 0.4 \text{v}
\end{align*}
\]

Peripheral output (open collector bipolar):

\[
\begin{align*}
V_{OL} &= 0.4 \text{v max.} @ 1.6 \text{ ma.} \\
V_{OH} &= 4.5 \text{v min. with external 100Kohm pull-up.}
\end{align*}
\]

Vcc/READY input:

\[
\begin{align*}
V_{IH} &= 2.0 \text{v min.} @ I_{IH} = 1\text{ma. max.} \\
V_{IL} &= 0.4 \text{v max.} \\
\text{Input goes to logic zero when open.}
\end{align*}
\]

Bus Commands

The bus protocol specifies that all commands must originate from the computer, and that peripherals will present data on the bus only when commanded to. Every bus operation will go to completion before another bus operation is initiated (no overlap). An error detected at any point in the command sequence will abort the entire sequence.

A bus operation consists of the following elements:

Command frame from the computer.

Acknowledgement (ACK) from the peripheral.

Optional data frame to or from the computer.

Operation complete (COMPLETE) from the peripheral.
Command Frame

The serial bus protocol provides for three types of commands: 1) data send, 2) data receive and 3) immediate (no data -- command only). There is a common element in all three types, a command frame consisting of five bytes of information sent from the computer while the COMMAND- line is held low. The format of the command frame is shown below:

```
+----------------+  
| device ID      |  
+----------------+  
| command        |  
+----------------+  
| auxiliary #1    |  
+----------------+  
| auxiliary #2    |  
+----------------+  
| checksum       |  
+----------------+  
```

Figure 9-5  Serial Bus Command Frame Format

The device ID specifies that of the serial bus devices is being addressed (see Appendix I for a list of device IDs).

The command byte contains a device-dependent command (see Appendix I for a list of device commands).

The auxiliary bytes contain more device-dependent information.

The checksum byte contains the arithmetic sum of the first four bytes (with the carry added back after every addition).

Command Frame Acknowledge

The peripheral being addressed would normally respond to a command frame by sending an ACK byte ($41) to the computer; if there is a problem with the command frame, the peripheral should not respond.

Data Frame
Following the command frame (and ACK) can be an optional data frame that is formatted as shown below:

```
+----------------+
|               |
|               |
|    data      |
|   =   bytes  |
|               |
+----------------+
|               |
+----------------+

This data frame can originate at the computer or at the device controller, depending upon the command. Current device controllers expect fixed-length data frames as does the computer, where the data frame length is a fixed function of the device type and command.

The checksum value in the data frame is the arithmetic sum of all of the frame data preceding the checksum, with the carry from each addition being added back (the same as for the command frame).

In the case of the computer sending a data frame to a peripheral, the peripheral is expected to send an ACK if the data frame is acceptable, and send a NAK ($4E), or do nothing if the data frame is unacceptable. See the first flowchart in Section 9.

Operation Complete

A peripheral is also expected to send an operation-COMPLETE byte ($43) at the time the commanded operation is complete. The location of this byte in the command sequence for each command type is shown in the timing diagrams in Section 9. If the operation cannot go to normal, error-free completion, the peripheral should respond with an ERROR byte ($45) instead of COMPLETE.
Bus Timing

This section provides timing diagrams for the three types of command sequences: data send, data receive, and immediate.

**DATA SEND sequence:**

```
<table>
<thead>
<tr>
<th>COMMAND-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+---------</td>
</tr>
</tbody>
</table>

DATA OUT

```

```
<table>
<thead>
<tr>
<th>cmnd</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
<td>frame</td>
</tr>
</tbody>
</table>
```

```
DATA IN

```

```
<table>
<thead>
<tr>
<th>cmnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
</tr>
</tbody>
</table>
```

**DATA RECEIVE sequence:**

```
<table>
<thead>
<tr>
<th>COMMAND-</th>
</tr>
</thead>
<tbody>
<tr>
<td>+---------</td>
</tr>
</tbody>
</table>

DATA OUT

```

```
<table>
<thead>
<tr>
<th>cmnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
</tr>
</tbody>
</table>
```

```
DATA IN

```

```
<table>
<thead>
<tr>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>ACK</th>
<th>CMPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>t0</td>
<td>t1 t2 t3 t4 t5</td>
</tr>
</tbody>
</table>
```

**OPERATING SYSTEM CO16555 -- Section 9**
IMMEDIATE sequence:

---+ +--------+
| cmd |
+----+-+

---+----+-+
| cmd |
---+frame +----+-+

Figure 9-6 Serial Bus Timing Diagram

The computer generates a delay \( t_0 \) between the lowering of COMMAND- and the transmission of the first byte of the command frame.

\[
\text{computer } t_0 \text{ (min)} = 750 \text{ microsec.} \\
\text{computer } t_0 \text{ (max)} = 1600 \text{ microsec.}
\]

\[
\text{peripheral } t_0 \text{ (min)} = ?? \\
\text{peripheral } t_0 \text{ (max)} = ??
\]

The computer generates a delay \( t_1 \) between the transmission of the last bit of the command frame and the raising of the COMMAND-line.

\[
\text{computer } t_1 \text{ (min)} = 650 \text{ microsec.} \\
\text{computer } t_1 \text{ (max)} = 950 \text{ microsec.}
\]

\[
\text{peripheral } t_1 \text{ (min)} = ?? \\
\text{peripheral } t_1 \text{ (max)} = ??
\]

The peripheral generates a delay \( t_2 \) between the raising of COMMAND- and the transmission of the ACK byte by the peripheral.

\[
\text{computer } t_2 \text{ (min)} = 0 \text{ microsec.} \\
\text{computer } t_2 \text{ (max)} = 16 \text{ msec.}
\]

\[
\text{peripheral } t_2 \text{ (min)} = ?? \\
\text{peripheral } t_2 \text{ (max)} = ??
\]
The computer generates a delay (t3) between the receipt of the last bit of the ACK byte and the transmission of the first bit of the data frame by the computer.

computer t3 (min) = 1000 microsec.
computer t3 (max) = 1800 microsec.

peripheral t3 (min) = ??
peripheral t3 (max) = ??

The peripheral generates a delay (t4) between the transmission of the last bit of the data frame and the receipt of the first bit of the ACK byte by the computer.

computer t4 (min) = 850 microsec.
computer t4 (max) = 16 msec.

peripheral t4 (min) = ??
peripheral t4 (max) = ??

The Peripheral generates a delay (t5) between the the receipt of the last bit of the ACK byte and the first bit of the COMPLETE byte by the computer.

computer t5 (min) = 250 microsec.
computer t5 (max) = 255 sec. (handler-dependent)

peripheral t5 (min) = ??
peripheral t5 (max) = N/A

HANDLER ENVIRONMENT

Nonresident handlers can be installed in at least three different manners:

1. As booted software from diskette or cassette.
2. Resident in a cartridge (A or B).
3. Downloaded from a serial bus device.

This section will discuss the basic mechanisms for handler installation for these environments. In order to fully utilize the information in this section, you must have read and understood the following sections:

Program environments . . . Section 3
RAM region . . . . . . . . . . . . . . . . . . . . Section 4
Memory dynamics. . . . . . . . . . . . . . . . . . . Section 4
System initialization . . . . . . . . . . . . . . . Section 7
Adding new device handlers/peripherals . . . . . . . . Section 9
Program environment and initialization . . . . . . . . Section 10
Bootable Handler

The diskette- or cassette-booted software will insert the handler's vector table pointer and name to the device table whenever the booted software's initialization entry point is entered (on power-up and system reset). Remember that both power-up and system reset clear the device table of all but the resident handler entries.

Cartridge Resident Handler

The cartridge software will insert the handler's vector table pointer and name to the device table whenever the cartridge's initialization entry point is entered (on power-up and system reset). Remember that both power-up and system reset clear the device table of all but the resident handler entries; therefore the device table must be reestablished by the handler-initialization procedure upon every entry.

FLOWCHARTS

The following pages contain process flowcharts showing the SIO and peripheral actions for the Serial bus command forms.
PERIPHERAL'S COMMAND FRAME PROCESSING

IDLE

WAIT FOR HIGH TO LOW TRANSITION ON COMMAND-

GET NEXT 5 BYTES ON THE BUS

WAIT FOR COMMAND- TO GO HIGH

IS CHECKSUM OK?

NOT

YES

THIS DEVICE?

NOT

YES

VALID COMMAND?

NOT

SEND NAK

YES

VALID AUX DATA

NOT

SEND ACK

YES

A/B/C
This section discusses possible alternative software environments using OS Configurations. Environments other than those discussed here are also possible. A thorough understanding of the power-up and system reset processes (see Section 7) will be necessary to evaluate all alternative environments.

CARTRIDGE

Most games (and some language processors) are supported via the cartridge environment. The cartridge resident software is in control of the system, sometimes using the OS and sometimes not. A cartridge can specify whether the diskette is to be booted at power-up time, whether the cartridge is to provide the controlling software, or whether the cartridge is a special diagnostic cartridge. These options are specified by bits in the cartridge header, as shown below:

```
+----------------+
| cartridge     |
BFFA (9FFA for cartridge B)
+-- --+
| start address |
+-----------+
| 00         |
+-----------+
| option byte |
+-----------+
| cartridge  |
+-- --+
| init address |
BFFF (9FFF for cartridge B)
+-----------+
```

Figure 10-1  Cartridge Header Format

The byte of "00" is used to allow the OS to determine when a cartridge is inserted; locations BFFC and 9FFC will not read zero when there is neither RAM at those locations nor a cartridge inserted. RAM is differentiated from a cartridge by its ability to be altered.
The option byte has the following option bits:

- bit 0 = 0, then do not boot the diskette.
  1, then boot the diskette.

- Bit 2 = 0, then init but do not start the cartridge.
  1, then init and start the cartridge.

- bit 7 = 0, then cartridge is not a diagnostic cartridge.
  1, then cartridge is a diagnostic cartridge and control will be given to the cartridge before any of the OS is initialized (JMP (BFFE)).

The cartridge init address specifies the location to which the OS will JSR during all power-up and system reset operations. As a minimum, this vector should point to an RTS instruction.

The cartridge start address specifies the location to which the OS will JMP during all power-up and system reset operations, if bit 1 of the option byte is = 1. The application should examine the variable WARMST [0008] if system reset action is to be different than power-up (WARMST will be zero on power-up and nonzero thereafter).

Cartridge Without Booted Support Package

A cartridge that does not specify the diskette-boot option and does not support the cassette-boot possibility can use lower memory (from 0480 to the address in MEMTOP [02E5]) in any way it sees fit.

Cartridge With Booted Support Package

A cartridge that does specify the diskette-boot option or does support the cassette-boot possibility must use some care in its use of lower memory. The following regions are defined:

- 0480-06FF is always available to the cartridge.
- MEMLO/MEMTOP region is always available to the cartridge.

DISKETTE-BOOTED SOFTWARE

Software can be booted from the disk drive at power-up time in response to one of the following conditions:
Neither Cartridge A nor B is inserted.

Cartridge A is inserted and has bit 0 of its option byte \([BFFD] = 1\).

Cartridge B is inserted and has bit 0 of its option byte \([9FFD] = 1\).

If any of these conditions are met, the OS will attempt to read the boot record from sector #1 of disk drive 1 and then transfer control to the software that was read in. The exact sequence of operations will be explained later in this section.

**Diskette-Boot File Format**

The key region of a diskette-boot file is the first six bytes, which are formatted as shown below:

```
+---------------+ first byte
| flags        |
+---------------+ # of sectors
+---------------+ memory address
| to start load |
+---------------+ init
| address       | sixth byte
+---------------+ boot
| continuation  |
| code          |
```

**Figure 10-2 Diskette-Boot File Format**

The first byte is stored in DFLAGS [0240], but is otherwise unused. It should equal zero.

The second byte contains the number of 128-byte diskette sectors to be read as part of the boot process (including the record containing this information). This number can range from 1 to 255, with 0 meaning 256.
The third and fourth bytes contain the address \((lo, hi)\) at which to start loading the first byte of the file.

The fifth and sixth bytes contain the address \((lo, hi)\) to which the booter will transfer control after the boot process is complete and whenever the [SYSTEM.RESET] key is pressed.

The Diskette File Management System (FMS) has extra bytes assigned to its boot record, but this is a special case of the generalized diskette-boot and is discussed in Section 5.

Diskette-Boot Process

If no cartridge is installed, then the diskette will follow these steps to boot up:

1. Read the first diskette record to the cassette buffer [0400].
2. Extract information from the first six bytes:
   - Save the flags byte to DFLAGS [0240, 1].
   - Save the # of sectors to boot to DBSECT [0241, 1].
   - Save the load address to BOOTAD [0242, 2].
   - Save the initialization address in DOSINI [000C, 2].
3. Move the record just read to the load address specified.
4. Read the remaining records directly to the load area.
5. JSR to the load address+6 where a multistage boot process can continue. The carry bit indicates the success of this operation (carry set = error, carry reset = success).

NOTE: During step 5, after the initial boot process is complete, the booter will transfer control to the seventh byte of the first record. The software should continue the boot process at this point, if it is a multistage boot. The value of MEMLO [02E7] should point to the first free RAM location beyond the software just booted. It should be established by the booted software as shown below:

```
LDA #END+1 ; SET UP LSB.
STA MEMLO
STA APPMHI
LDA #END+1/256 ; SET UP MSB.
STA MEMLO+1
STA APPMHI+1
```

If the booted software is to take control of the system at the end of the boot operation, the vector DOSVEC [000A] must be set up by the application at this time; DOSVEC points to the...
restart entry for the booted application. If the booted software is not to take control, then DOSVEC should remain unchanged.

```
LDA #RESTART ; RESTART LSB.
STA DOSVEC
LDA #RESTART/256
STA DOSVEC+1
```

6. JSR indirectly through DOSINI for initialization of the application; the application will initialize and return.

NOTE: The OS enters the initialization point on every system reset and power-up. Internal initialization can take place during system reset and power-up as well. Initialization can also be deferred until Step 7 for controlling applications.

7. JMP indirectly through DOSVEC to transfer control to the application.

NOTE: Pressing the [SYSTEM.RESET] key after the application is fully booted will cause steps 6 and 7 to be repeated.

Sample Diskette-Bootable Program Listing

This skeletal program can be booted from the diskette. It retains control when it is entered.

```
; THIS IS THE START OF THE PROGRAM FILE.
PST= $0700 ; (OR SOME OTHER LOCATION).
** PST ; (.ORG).

; THIS IS THE diskette-boot CONTROL INFORMATION.

.BYTE 0 ;
.BYTE PND-PST+127/128 ; NUMBER OF RECORDS.
.WORD PST ; MEMORY ADDRESS TO START LOAD.
.WORD PINIT ; PROGRAM INIT.
```
; THIS IS THE START OF THE BOOT CONTINUATION.

LDA #PND           ; ESTABLISH LOW MEMORY LIMITS.
STA MEMLO
STA APPMHI
LDA #PND/256
STA MEMLO+1
STA APPMHI+1

LDA #RESTR        ; ESTABLISH RESTART VECTOR.
STA DOSVEC
LDA #RESTR/256
STA DOSVEC+1

CLC               ; SET FLAG FOR SUCCESSFUL BOOT.
RTS

; APPLICATION INITIALIZATION ENTRY POINT.

PINIT RTS         ; NOTHING TO DO HERE FOR ...
; ... CONTROLLING APPLICATION.

; THE MAIN BODY OF THE PROGRAM follows.

RESTR=*           ; THE MAIN BODY OF THE PROGRAM ENDS HERE.

PND= *            ; 'PND' = NEXT FREE LOCATION.
. END

Figure 10-3 Diskette-Bootable Program Listing Example

Program to Create Diskette-Boot Files

This section provides a program that can be used to make bootable files on diskettes. The program given is not the only one possible, and no claims are made as to its elegance.
Shown below is a listing of the program to create diskette-boot files.

; THIS PROGRAM WRITES A SINGLE "FILE" TO THE DISKETTE AND IS
; USED IN CONJUNCTION WITH A PROCEDURE TO MAKE DISKETTE-
; BOOTABLE FILES. THE FOLLOWING TWO SYMBOLS MUST BE EQUATED
; USING THE MEMORY LIMITS OF THE PROGRAM TO BE COPIED:
; 'PST' = PROGRAM START ADDRESS (SEE SAMPLE PROGRAM).
; 'PND' = PROGRAM END ADDRESS (SEE SAMPLE PROGRAM).

SECSIZ=128 ; DISKETTE SECTOR SIZE.
PST= $0700
PND= $1324
FLEN= PND-PST+SECSIZ-1/SECSIZ ; # OF SECTORS IN FILE.
**= $8000 ; THIS PROGRAM'S ORIGIN.

BOOTB BRK ; *** LOAD APPLICATION ***

; SET UP DEVICE CONTROL BLOCK FOR DISKETTE HANDLER CALL

LDA #FLEN ; # OF SECTORS TO WRITE.
STA COUNT

LDA #1 ; DISK DRIVE #1.
STA DUNIT

LDA #'W ; SET UP FOR WRITE WITH CHECK.
STA DCOMND

LDA #PST ; POINT TO START OF APPLIC. PROG.
STA DBUFLO
LDA #PST/256
STA DBUFHI

LDA #01 ; SET UP STARTING SECTOR # = 0001.
STA DAUX1
LDA #00
STA DAUX2
; NOW WRITE THE FILE ONE SECTOR AT A TIME.

BOTO10 JSR DSKINV ; WRITE ONE SECTOR.
    BMI DERR ; ERROR.
    LDA DBUFLO ; INCREMENT MEMORY ADDRESS.
    CLC
    ADC #SECSIZ
    STA DBUFLO
    LDA DBUFHI
    ADC #0
    STA DBUFHI
    INC DAUX1 ; INCREMENT SECTOR #.
    BNE BOTO20
    INC DAUX2

BOTO20 DEC COUNT ; MORE SECTORS TO WRITE?
    BNE BOTO10 ; YES.
    BRK ; STOP WHEN DONE.
DERR BRK ; STOP ON ERROR.
COUNT +=1 ; SECTOR COUNT.

; THIS IS THE CARTRIDGE HEADER

*= $BFF9 ; "A" CARTRIDGE.

INIT RTS
    .WORD BOOTB
    .BYTE 0,4
    .WORD INIT
    .END

CASSETTE-BOOTED SOFTWARE

You can boot software from the cassette as well as from the diskette, at power-up. The following requirements must be met in order to boot from the cassette:

- You must be pressing the [START] key as power is applied to the system.

- A cassette tape with a proper boot format file must be installed in the cassette drive, and the PLAY button must be pressed.
When you are given the audio prompt by the cassette handler you must press the [RETURN] key.

If all of these conditions are met, the OS will read the boot file from the cassette and then transfer control to the software that was read in. The exact sequence of operations will be explained later in this section.

Cassette-Boot File Format

The key region of a cassette-boot file is the first six bytes, that are formatted as shown below:

```
+----------------+
|                |
+----------------+
| # of Records   |
+----------------+
| Memory Address |
+-+                |
| To Start Load  |
+----------------+
| Init           |
+-+                |
| address        |
+----------------+
```

The first byte is not used by the cassette-boot process.

The second byte contains the number of 128-byte cassette records to be read as part of the boot process (including the record containing this information). This number can range from 1 to 255, with 0 meaning 256.

The third and fourth bytes contain the address (lo, hi) to which the booter will transfer control after the boot process is complete and whenever the [SYSTEM.RESET] key is pressed.

Cassette-Boot Process

The cassette-boot process is described step-by-step for a configuration in that no cartridge is installed and no diskettes are attached. For the general case see Section 7.

1. Read the first cassette record to the cassette buffer.

2. Extract information from the first six bytes:
Save the # of records to boot. Save the load address. Save the initialization address in CASINI [0002]

3. Move the record just read to the load address specified.

4. Read the remaining records directly to the load area.

5. JSR to the load address+6 where a multistage boot process can continue; the carry bit will indicate the success of this operation (carry set=error, carry reset=success).

6. JSR indirectly through CASINI for initialization of the application; the application will initialize and return.

7. JMP indirectly through DOSVEC to transfer control to the application.

Pressing the [SYSTEM.RESET] key after the application is fully booted will cause steps 6 and 7 to be repeated.

NOTE: After the initial boot process is complete, the booter will transfer control to the seventh byte of the first record; at this point the software should continue the boot process (if it is a multistage boot) and then stop the cassette drive, which due to a system bug will still be running, using the following instruction sequence:

```
  LDA #$3C
  STA PACTL [D302]
```

The application should then set a value in MEMLO [0237] that points to the first free RAM location beyond the software just booted, as shown below:

```
  LDA #END+1
  STA MEMLO
  STA APPMHI
  LDA #END+1/256
  STA MEMLO+1
  STA APPMHI+1
```

If the booted software is to take control of the system at the end of the boot operation, the vector DOSVEC [000A] must be set up by the application at this time; DOSVEC points to the restart entry for the booted application. If the booted software is not to take control, then DOSVEC should remain unchanged.

```
  LDA #RESTRT ; RESTART LSB
  STA DOSVEC
  LDA #RESTRT/256
  STA DOSVEC+1
```

NOTE: The initialization point is entered on every system reset and power-up; internal initialization can take place here.
For controlling applications initialization can also be deferred until step 7.

Sample Cassette-Bootable Program Listing

Shown below is a skeletal program that can be booted from the cassette and that retains control when it is entered.

; THIS IS THE START OF THE PROGRAM FILE.

PST= $0700 ; (OR SOME OTHER LOCATION).
*= PST ; (.ORG).

; THIS IS THE cassette-boot CONTROL INFORMATION.

. BYTE 0 ; (DOESN'T MATTER).
. BYTE PND-PST+127/128 ; NUMBER OF RECORDS.
. WORD PST ; MEMORY ADDRESS TO START LOAD.
. WORD PINIT ; PROGRAM INIT.

; THIS IS THE START OF THE BOOT CONTINUATION.

LDA #**3C ; STOP THE CASSETTE.
STA PACTL

LDA #PND ; ESTABLISH LOW MEMORY LIMITS.
STA MEMLO
STA APPMHI
LDA #PND/256
STA MEMLO+1
STA APPMHI+1

LDA #RESTR ; ESTABLISH RESTART VECTOR.
STA DOSVEC
LDA #RESTR/256
STA DOSVEC+1

CLC ; SET FLAG FOR SUCCESSFUL BOOT.
RTS

; APPLICATION INITIALIZATION ENTRY POINT.

PINIT RTS ; NOTHING TO DO HERE FOR ...
; ... CONTROLLING APPLICATION.

; THE MAIN BODY OF THE PROGRAM Follows.

RESTR=* ;

; THE MAIN BODY OF THE PROGRAM ENDS HERE.

OPERATING SYSTEM CD16555 -- Section 10
PND= * ; 'PND' = NEXT FREE LOCATION.

.END

Figure 10-4 Sample Cassette-Bootable Program

Program to Create Cassette-Boot Files

This section provides a program listing that can be used to make bootable files on cassette tapes. The program given is not the only one possible, and no claims are made as to its elegance.

Shown below is a listing of the program to create a cassette-boot file:

; THIS PROGRAM WRITES A SINGLE FILE TO THE CASSETTE AND IS
; USED IN CONJUNCTION WITH A PROCEDURE TO MAKE CASSETTE-
; BOOTABLE FILES. THE FOLLOWING TWO SYMBOLS MUST BE EQUATED
; USING THE MEMORY LIMITS OF THE PROGRAM TO BE COPIED:

; 'PST' = PROGRAM START ADDRESS (SEE SAMPLE PROGRAM).
; 'PND' = PROGRAM END ADDRESS (SEE SAMPLE PROGRAM).

PST= $0700
PND= $1324
FLEN= PND-PST+127/128*128 ; ROUND UP TO MULTIPLE OF 128.

== $8000 ; THIS PROGRAM'S ORIGIN.

BOOTB LDX #$10 ; USE IOCB #1.

; FIRST OPEN THE CASSETTE FILE FOR WRITING.
LDA #OPEN
STA ICCOM, X ; SET UP FOR DEVICE "OPEN."
LDA #OPNOT
STA ICAX1, X ; DIRECTION IS "OUTPUT."
LDA #$80
STA ICAX2, X ; SELECT SHORT IRG.
LDA #CFILE
STA ICBAL, X
LDA #CFILE/256
STA ICBAH, X
JSR CIOV ; ATTEMPT TO OPEN FILE.
BMI CERR ; ERROR.

; NOW WRITE THE ENTIRE FILE AS ONE OPERATION.

OPERATING SYSTEM CO16555 -- Section 10
LDA #PUTCHR ; SET UP FOR "PUT CHARACTERS."
STA ICCOM, X

LDA #PST ; POINT TO START OF APPLIC. PROG.
STA ICBAL, X
LDA #PST/256
STA ICBAH, X

LDA #FLEN ; SET UP # OF BYTES TO WRITE.
STA ICBLL, X
LDA #FLEN/256
STA ICBLH, X

JSR CIOV ; WRITE ENTIRE FILE.
BMI CERR ; ERROR.

; NOW CLOSE THE FILE AFTER SUCCESSFUL WRITE.
LDA #CLOSE ; SET UP FOR "CLOSE."
STA ICCOM, X

JSR CIOV ; CLOSE THE FILE.
BMI CERR ; ERROR.

BRK ; STOP WHEN DONE.
CERR BRK ; STOP ON ERROR.

CFILE .BYTE "C:", CR ; FILE NAME.

; THIS IS THE CARTRIDGE HEADER
*= $BFF9 ; "A" CARTRIDGE.

INIT RTS
.WORD BOOTB
.BYTE 0, 4
.WORD INIT
.END
This section presents information to use the capabilities of the OS and some of the hardware capabilities that aren't directly available through the OS, and in fact, can be in direct conflict with parts of the OS.

**SOUND GENERATION**

The OS uses the POKEY sound generation capabilities only in the I/O subsystem, for cassette FSK tone generation, and for the "noisy bus" option in SIO.

**Capabilities**

The hardware provides four independently programmable audio channels that are mixed and sent to the television set as part of the composite video signal. The POKEY registers shown below are all concerned with sound control (as described in the ATARI Home Computer Hardware Manual).

- **AUDCTL [D208]** Audio control.
- **AUDC1 [D201] and AUDF1 [D200]** Channel 1 control.
- **AUDC2 [D203] and AUDF2 [D202]** Channel 2 control.
- **AUDC3 [D205] and AUDF3 [D204]** Channel 3 control.
- **AUDC4 [D207] and AUDF4 [D206]** Channel 4 control.

**Conflicts With OS**

There are two potential conflicts with the OS involving sound generation:

- The OS can generate its own sounds and then turn off all sounds as part of I/O operations to the cassette and the serial bus peripherals.

- The OS does not turn off sounds when you press [SYSTEM.RESET] or [BREAK]. If the sounds are to be turned off under those conditions, the controlling program must provide that capability.
SCREEN GRAPHICS

Hardware Capabilities

The hardware capabilities for screen presentations are quite versatile; the OS uses a very small amount of the capability provided. The means of extension, however, are non-trivial; and making changes to a screen format while still utilizing the resident Display Handler will be difficult. See the ATARI Home Computer Hardware Manual for information regarding screen presentations.

OS Capabilities

The resident Display Handler arbitrarily supports 8 of the 11 possible full screen modes (11 of 14 modes if the GTIA chip is used in place of the CTIA). The resident Display Handler allows for an optional "split-screen" text window of fixed size. The hardware allows for many more options than the Display Handler supports, as will be seen by reading the ATARI Home Computer Hardware Manual.

Cursor Control

You can control the Display Handler text and graphics cursors directly (see Section 5 and Appendix L, B1-4).

Color Control

You can alter the color register assignments that the Display Handler makes upon all OPEN commands (see Appendix L B7-8 and elsewhere). Note that every system reset or Display Handler OPEN will reset the values back to the system default.
Alternate Character Sets

Two character sets are available in screen text modes 1 and 2. The value stored in the database variable CHBAS [02F4] selects the character set of interest to you. The default value of $E0 provides capital (uppercase) letters, numbers and the punctuation characters corresponding to display codes $20 through $5F in Appendix E). The alternate value of $E2 provides lowercase letters and the special character graphics set (corresponding to display codes $60 through $7F and $00 through $1F in Appendix E).

User-defined character sets can also be obtained for text modes 0, 1, and 2 by providing the character matrix definitions in RAM and setting CHBAS to point to those definitions. CHBAS always contains the most significant bits of the memory address of the start of the character definitions, as shown below:

```
<table>
<thead>
<tr>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>000101</td>
</tr>
<tr>
<td>x</td>
</tr>
</tbody>
</table>
```

Text mode 0

```
<table>
<thead>
<tr>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>000110</td>
</tr>
<tr>
<td>x</td>
</tr>
</tbody>
</table>
```

Text modes 1 and 2

Figure 11-1 User-Defined Character Set Bit Memory Addresses

(X indicates an ignored address bit assumed to be 0.)

172
Each character is defined by an 8 x 8 bit matrix; the character 'a' is defined as shown below:

```
<table>
<thead>
<tr>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>++-+-+-+-+-+-+-+-+-+-+</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>++-+-+-+-+-+-+-+-+-+-+</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>++-+-+-+-+-+-+-+-+-+-+</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>++-+-+-+-+-+-+-+-+-+-+</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Figure 11-2 User Defined 8 x 8 Character Matrix Bit Table

The storage for the character set involves eight consecutive bytes for each character with characters ordered consecutively by their internal code value (see the discussion in Appendix L relating to B55).

```
Character base |
| code $00 | 8 bytes |
++-+-+-+-+-+-+-+-+-+-+ |
| code $01 | increasing addresses |
++-+-+-+-+-+-+-+-+-+-+ |
| code $7E |
++-+-+-+-+-+-+-+-+-+-+ |
| code $7F |
++-+-+-+-+-+-+-+-+-+-+ |
```

Figure 11-3 Character Base Diagram

PLAYER/MISSILE GRAPHICS

The OS makes no use of the player/missile generation capability of the hardware. It can be used independently of the OS with no conflict.
Hardware Capabilities

The hardware allows a number of independently moveable screen objects of limited width to be positioned and moved about the screen without affecting the "playfield" (bit-mapped graphics or character) data. Priority control allows the various objects to have a display precedence in case of conflict (overlap).

Conflicts With OS

You must assure that the player/missile data is address-aligned as required by PMBASE [D407]. You also must find a suitable free area that the OS guarantees to be free under all environments.

READING GAME CONTROLLERS

The OS reads the game controllers (shown below) as part of the stage 2 VBLANK process (see Appendix L): 1-9:

- Joysticks/triggers 1-4.
- Paddle controllers/triggers 1-8.
- Driving controllers/triggers 1-4.
- Light pen/trigger

In addition to these controllers, other information can be sensed or sent using the PIA chip to that the console connectors are interfaced.

Keyboard Controller Sensing

Data can be read from an ATARI keyboard controller connected to the first port. This program alters registers on a chip called a PIA. To set these back to the default values to do further I/O, hit [SYSTEM.RESET] or POKE PACTL, 60. If this program is to be loaded from diskette, use LOAD, not RUN and wait for the busy light on the disk drive to go out. Do not execute the program before this light goes out, otherwise the diskette continues to spin.

```
1 GRAPHICS 0
5 PRINT :PRINT " KEYBOARD CONTROLLER DEMO"
10 DIM ROW(3), I$(13), BUTTON$(1)
30 GOSUB 6000
40 FOR CNT=1 TO 4
60 POSITION 2, CNT*2+5: PRINT "CONTROLLER # "; CNT; ": ";
```

OPERATING SYSTEM CO16555 -- Section 11
70 NEXT CNT
80 FOR CNT=1 TO 4: GOSUB 7000: POSITION 19, CNT+CNT+5: PRINT BUTTON$: NEXT CNT
120 GOTO 80
6000 REM ** SET UP FOR CONTROLLERS **
6010 PORTA=54016: PORTB=54017: PACTL=54018: PBCTL=54019
6020 POKE PACTL, 48: POKE PORTA, 255: POKE PACTL, 52: POKE PORTA, 221
6025 POKE PBCTL, 48: POKE PORTB, 255: POKE PBCTL, 52: POKE PORTB, 221
6030 ROW(0)=238: ROW(1)=221: ROW(2)=187: ROW(3)=119
6040 I$="123456789*0*"
6050 RETURN
7000 REM ** RETURN BUTTON$ WITH CHARACTER FOR BUTTON WHICH HAS BEEN PRESSED ON CONTROLLER CNT (1-4). **
7001 REM ** NOTE: A 1 WILL BE RETURNED IF NO CONTROLLER IS CONNECTED. **
7002 REM ** A SPACE WILL BE RETURNED IF THE CONTROLLER IS CONNECTED BUT NO KEY HAS BEEN PRESSED. **
7003 PORT=PORTA: IF CNT>2 THEN PORT=PORTB
7005 P=1
7008 PAO=CNT+CNT-2
7010 FOR J=0 TO 3
7020 POKE PORT, ROW(J)
7030 FOR I=1 TO 10: NEXT I
7050 IF PADDLE(PAO+1)>10 THEN P=J+J+J+2: GOTO 7090
7060 IF PADDLE(PAO)>10 THEN P=J+J+J+3: GOTO 7090
7070 IF STRING(CNT-1)=0 THEN P=J+J+J+4: GOTO 7090
7080 NEXT J
7090 BUTTON$=I$(P, P)
7095 RETURN

Figure 11-4 Reading Data From an ATARI Keyboard Controller

OPERATING SYSTEM CO16555 -- Section 11
The table below shows the variable/register values used for reading a keyboard controller from each of the four controller ports.

<table>
<thead>
<tr>
<th>Port 1</th>
<th>Port 2</th>
<th>Port 3</th>
<th>Port 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORT A</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td>direction</td>
<td>OF</td>
<td>FO</td>
<td>-</td>
</tr>
<tr>
<td>bits</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td>PORT B</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td>direction</td>
<td>-</td>
<td>-</td>
<td>OF</td>
</tr>
<tr>
<td>bits</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td><img src="#" alt="Port A values" /></td>
<td><img src="#" alt="Port B values" /></td>
<td><img src="#" alt="Port A values" /></td>
<td><img src="#" alt="Port B values" /></td>
</tr>
<tr>
<td>row sel</td>
<td>FB,F7</td>
<td>BF,7F</td>
<td>-</td>
</tr>
<tr>
<td>ect</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td><img src="#" alt="Column 1 values" /></td>
<td><img src="#" alt="Column 2 values" /></td>
<td><img src="#" alt="Column 3 values" /></td>
<td></td>
</tr>
<tr>
<td>Column 1</td>
<td>PADDL1</td>
<td>PADDL3</td>
<td>PADDL5</td>
</tr>
<tr>
<td>Sense</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td><img src="#" alt="Column 2 values" /></td>
<td><img src="#" alt="Column 3 values" /></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column 2</td>
<td>PADDLO</td>
<td>PADDL2</td>
<td>PADDL4</td>
</tr>
<tr>
<td>Sense</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
<tr>
<td><img src="#" alt="Column 3 values" /></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column 3</td>
<td>STRIGO</td>
<td>STRIG1</td>
<td>STRIG2</td>
</tr>
<tr>
<td>Sense</td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
<td><img src="#" alt="Table values" /></td>
</tr>
</tbody>
</table>

**Figure 11-5** ATARI Keyboard Controller Variable/Register Value Table

Front Panel Connectors as I/O Ports

The three pages that follow show how some of the pins in the front panel (game controller) connectors can be used as general I/O pins.

**Hardware Information**

PIA (6520 / 6820)

Out: TTL levels, 1 load
In: TTL levels, 1 load

For more information refer to 6520 chip manual.

--------

OPERATING SYSTEM COD555 -- Section 11
Port A Circuit (typical):

---

Port B Circuit (typical):

"Trigger" Port Circuit (typical):

Software Information

6520 PIA: (This also pertains to all of the following: **)

Port A control (address D302)

Port A data direction (address D300)

Write this into this register.

Data direction control for Port A

Note: 50mA maximum total external drain on power supply allowed. 

OPERATING SYSTEM CO16555 -- Section 11
Port A data (address D300)

7 6 5 4 3 2 1 0

Read or Write this register

Jack 2       Jack 1
Pin Numbers

Port B Control (address D303)

0 0 1 1 1 X 0 0

6520 PIA:
Port B Control (address D303)

7 6 5 4 3 2 1 0

write this into this register

Port B Data/Data direction
addressing control
0 = D301 contains data
direction
1 = $D301 contains

Port B data direction (address D301)

7 6 5 4 3 2 1

write this into this register

data direction control for Port B
1 = Out
0 = In

Port B data (address D301)

7 6 5 4 3 2 1 0

Jack 4       Jack 3
Pin Numbers

Four "Trigger" ports: D010, D011, D012, D013

7 6 5 4 3 2 1 0

Read this port

Trigger Value
D010 = Port 1 Pin 6
D013 = Port 4 Pin 6

178 OPERATING SYSTEM CO16555 -- Section 11
Other Miscellaneous Software Information

1). The OS sets up all PIA ports as inputs during initialization.
2). The OS usually reads the above once per television frame (during vertical-blank) into RAM as follows:

<table>
<thead>
<tr>
<th>Data Base Name</th>
<th>Address</th>
<th>Data</th>
<th>Pins S</th>
</tr>
</thead>
<tbody>
<tr>
<td>STICK0</td>
<td>0278</td>
<td>7 6 5 4 3 2 1 0</td>
<td>Jack 1, pins 4,3,2, if 10053,7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0 X X X X</td>
<td></td>
</tr>
<tr>
<td>STICK1</td>
<td>0729</td>
<td></td>
<td>Jack 2, Pins 4,3,2,1</td>
</tr>
<tr>
<td>STICK2</td>
<td>027A</td>
<td></td>
<td>Jack 3, Pins 4,3,2,1</td>
</tr>
<tr>
<td>STICK3</td>
<td>027B</td>
<td></td>
<td>Jack 4, Pins 4,3,2,1</td>
</tr>
<tr>
<td>STRIGO</td>
<td>0284</td>
<td></td>
<td>Jack 1, Pin 6</td>
</tr>
<tr>
<td>STRIG1</td>
<td>0285</td>
<td>7 6 5 4 3 2 1 0</td>
<td>Jack 2, Pin 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
</tr>
<tr>
<td>STRIG2</td>
<td>0286</td>
<td></td>
<td>Jack 3, Pin 6</td>
</tr>
<tr>
<td>STRIG3</td>
<td>0287</td>
<td></td>
<td>Jack 4, Pin 6</td>
</tr>
<tr>
<td>PADDL1</td>
<td>0270</td>
<td>7 6 5 4 3 2 1 0</td>
<td>Jack 1, Pin 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X X X X X X X X</td>
<td></td>
</tr>
<tr>
<td>PADDL3</td>
<td>0272</td>
<td></td>
<td>Jack 3, Pin 5</td>
</tr>
<tr>
<td>PADDL5</td>
<td>0274</td>
<td></td>
<td>Jack 4, Pin 5</td>
</tr>
<tr>
<td>PADDL7</td>
<td>0276</td>
<td></td>
<td>Jack 1, Pin 9</td>
</tr>
<tr>
<td>PADDL0</td>
<td>0271</td>
<td></td>
<td>Jack 2, Pin 9</td>
</tr>
<tr>
<td>PADDL2</td>
<td>0273</td>
<td></td>
<td>Jack 3, Pin 9</td>
</tr>
<tr>
<td>PADDL4</td>
<td>0275</td>
<td></td>
<td>Jack 4, Pin 9</td>
</tr>
<tr>
<td>PADDL 6</td>
<td>0277</td>
<td></td>
<td>Jack 4, Pin 9</td>
</tr>
</tbody>
</table>

Figure 11-6 Using Front Panel Connectors As I/O Ports: Pin Function Tables

* Pins 5 and 9 are read through the paddle controller circuitry. A nominal value of 7 indicates that the pin is high (or floating) and a nominal value of 228 indicates that the pin is pulled low.
Appendix A -- CIO COMMAND BYTE VALUES

The following hex values are known to be legitimate CIO commands.

Most handlers:

03 -- OPEN
05 -- GET RECORD
07 -- GET CHARACTERS
09 -- PUT RECORD
08 -- PUT CHARACTERS
0C -- CLOSE
0D -- GET STATUS

Display Handler only:

11 -- FILL
12 -- DRAW

Diskette File Manager only:

20 -- RENAME
21 -- DELETE
22 -- FORMAT
23 -- LOCK
24 -- UNLOCK
25 -- POINT
26 -- NOTE
Appendix B -- CIO STATUS BYTE VALUES

Shown below are the known CIO STATUS BYTE values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 (001)</td>
<td>OPERATION COMPLETE (NO ERRORS)</td>
</tr>
<tr>
<td>80 (128)</td>
<td>[BREAK] KEY ABORT</td>
</tr>
<tr>
<td>81 (129)</td>
<td>IOCB ALREADY IN USE (OPEN)</td>
</tr>
<tr>
<td>82 (130)</td>
<td>NON-EXISTENT DEVICE</td>
</tr>
<tr>
<td>83 (131)</td>
<td>OPENED FOR WRITE ONLY</td>
</tr>
<tr>
<td>84 (132)</td>
<td>INVALID COMMAND</td>
</tr>
<tr>
<td>85 (133)</td>
<td>DEVICE OR FILE NOT OPEN</td>
</tr>
<tr>
<td>86 (134)</td>
<td>INVALID IOCB NUMBER (Y reg only)</td>
</tr>
<tr>
<td>87 (135)</td>
<td>OPENED FOR READ ONLY</td>
</tr>
<tr>
<td>88 (136)</td>
<td>END OF FILE</td>
</tr>
<tr>
<td>89 (137)</td>
<td>TRUNCATED RECORD</td>
</tr>
<tr>
<td>8A (138)</td>
<td>DEVICE TIMEOUT (DOESN'T RESPOND)</td>
</tr>
<tr>
<td>8B (139)</td>
<td>DEVICE NAK</td>
</tr>
<tr>
<td>8C (140)</td>
<td>SERIAL BUS INPUT FRAMING ERROR</td>
</tr>
<tr>
<td>8D (141)</td>
<td>CURSOR out-of-range</td>
</tr>
<tr>
<td>8E (142)</td>
<td>SERIAL BUS DATA FRAME OVERRUN ERROR</td>
</tr>
<tr>
<td>8F (143)</td>
<td>SERIAL BUS DATA FRAME CHECKSUM ERROR</td>
</tr>
<tr>
<td>90 (144)</td>
<td>DEVICE DONE ERROR</td>
</tr>
<tr>
<td>91 (145)</td>
<td>BAD SCREEN MODE</td>
</tr>
<tr>
<td>92 (146)</td>
<td>FUNCTION NOT SUPPORTED BY HANDLER</td>
</tr>
<tr>
<td>93 (147)</td>
<td>INSUFFICIENT MEMORY FOR SCREEN MODE</td>
</tr>
<tr>
<td>A0 (160)</td>
<td>DISK DRIVE # ERROR</td>
</tr>
<tr>
<td>A1 (161)</td>
<td>TOO MANY OPEN DISK FILES</td>
</tr>
<tr>
<td>A2 (162)</td>
<td>DISK FULL</td>
</tr>
<tr>
<td>A3 (163)</td>
<td>FATAL DISK I/O ERROR</td>
</tr>
<tr>
<td>A4 (164)</td>
<td>INTERNAL FILE # MISMATCH</td>
</tr>
<tr>
<td>A5 (165)</td>
<td>FILE NAME ERROR</td>
</tr>
<tr>
<td>A6 (166)</td>
<td>POINT DATA LENGTH ERROR</td>
</tr>
<tr>
<td>A7 (167)</td>
<td>FILE LOCKED</td>
</tr>
<tr>
<td>A8 (168)</td>
<td>COMMAND INVALID FOR DISK</td>
</tr>
<tr>
<td>A9 (169)</td>
<td>DIRECTORY FULL (64 FILES)</td>
</tr>
<tr>
<td>AA (170)</td>
<td>FILE NOT FOUND</td>
</tr>
<tr>
<td>AB (171)</td>
<td>POINT INVALID</td>
</tr>
</tbody>
</table>
Appendix C -- SIO STATUS BYTE VALUES

Shown below are the known SIO STATUS BYTE hexadecimal values.

01 (001) -- OPERATION COMPLETE (NO ERRORS)

8A (138) -- DEVICE TIMEOUT (DOESN'T RESPOND)
8B (139) -- DEVICE NAK
8C (140) -- SERIAL BUS INPUT FRAMING ERROR
8E (142) -- SERIAL BUS DATA FRAME OVERRUN ERROR
8F (143) -- SERIAL BUS DATA FRAME CHECKSUM ERROR
90 (144) -- DEVICE DONE ERROR
## Appendix D -- ATASCII CODES

<table>
<thead>
<tr>
<th>0X</th>
<th>2X</th>
<th>4X</th>
<th>6X</th>
<th>8X</th>
<th>AX</th>
<th>CX</th>
<th>EX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ØØ</td>
<td>Space</td>
<td>@</td>
<td>a</td>
<td>ØØ</td>
<td>ØØ</td>
<td>ØØ</td>
<td>ØØ</td>
</tr>
<tr>
<td>Ø1</td>
<td>!</td>
<td>A</td>
<td>a</td>
<td>Ø1</td>
<td>Ø1</td>
<td>Ø1</td>
<td>Ø1</td>
</tr>
<tr>
<td>Ø2</td>
<td>&quot;</td>
<td>B</td>
<td>b</td>
<td>Ø2</td>
<td>Ø2</td>
<td>Ø2</td>
<td>Ø2</td>
</tr>
<tr>
<td>Ø3</td>
<td>#</td>
<td>C</td>
<td>c</td>
<td>Ø3</td>
<td>Ø3</td>
<td>Ø3</td>
<td>Ø3</td>
</tr>
<tr>
<td>Ø4</td>
<td>$</td>
<td>D</td>
<td>d</td>
<td>Ø4</td>
<td>Ø4</td>
<td>Ø4</td>
<td>Ø4</td>
</tr>
<tr>
<td>Ø5</td>
<td>%</td>
<td>E</td>
<td>e</td>
<td>Ø5</td>
<td>Ø5</td>
<td>Ø5</td>
<td>Ø5</td>
</tr>
<tr>
<td>Ø6</td>
<td>&amp;</td>
<td>F</td>
<td>f</td>
<td>Ø6</td>
<td>Ø6</td>
<td>Ø6</td>
<td>Ø6</td>
</tr>
<tr>
<td>Ø7</td>
<td>'</td>
<td>G</td>
<td>g</td>
<td>Ø7</td>
<td>Ø7</td>
<td>Ø7</td>
<td>Ø7</td>
</tr>
<tr>
<td>Ø8</td>
<td>(</td>
<td>H</td>
<td>h</td>
<td>Ø8</td>
<td>Ø8</td>
<td>Ø8</td>
<td>Ø8</td>
</tr>
<tr>
<td>Ø9</td>
<td>)</td>
<td>I</td>
<td>i</td>
<td>Ø9</td>
<td>Ø9</td>
<td>Ø9</td>
<td>Ø9</td>
</tr>
<tr>
<td>ØA</td>
<td>*</td>
<td>J</td>
<td>j</td>
<td>ØA</td>
<td>ØA</td>
<td>ØA</td>
<td>ØA</td>
</tr>
<tr>
<td>ØB</td>
<td>+</td>
<td>K</td>
<td>k</td>
<td>ØB</td>
<td>ØB</td>
<td>ØB</td>
<td>ØB</td>
</tr>
<tr>
<td>ØC</td>
<td>,</td>
<td>L</td>
<td>l</td>
<td>ØC</td>
<td>ØC</td>
<td>ØC</td>
<td>ØC</td>
</tr>
<tr>
<td>ØD</td>
<td>-</td>
<td>M</td>
<td>m</td>
<td>ØD</td>
<td>ØD</td>
<td>ØD</td>
<td>ØD</td>
</tr>
<tr>
<td>ØE</td>
<td>.</td>
<td>N</td>
<td>n</td>
<td>ØE</td>
<td>ØE</td>
<td>ØE</td>
<td>ØE</td>
</tr>
<tr>
<td>ØF</td>
<td>/</td>
<td>O</td>
<td>o</td>
<td>ØF</td>
<td>ØF</td>
<td>ØF</td>
<td>ØF</td>
</tr>
<tr>
<td>1Ø</td>
<td>Ø</td>
<td>P</td>
<td>p</td>
<td>1Ø</td>
<td>1Ø</td>
<td>1Ø</td>
<td>1Ø</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>Q</td>
<td>q</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>R</td>
<td>r</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>S</td>
<td>s</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>T</td>
<td>t</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>U</td>
<td>u</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>V</td>
<td>v</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>7</td>
<td>W</td>
<td>w</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>18</td>
<td>8</td>
<td>X</td>
<td>x</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>9</td>
<td>Y</td>
<td>y</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>1A</td>
<td>:</td>
<td>Z</td>
<td>z</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
</tr>
<tr>
<td>1B</td>
<td>;</td>
<td>[</td>
<td></td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
</tr>
<tr>
<td>1C</td>
<td>&lt;</td>
<td>\</td>
<td></td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
</tr>
<tr>
<td>1D</td>
<td>=</td>
<td>]</td>
<td></td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
</tr>
<tr>
<td>1E</td>
<td>&gt;</td>
<td>^</td>
<td></td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
</tr>
<tr>
<td>1F</td>
<td>?</td>
<td>-</td>
<td></td>
<td>1F</td>
<td>1F</td>
<td>1F</td>
<td>1F</td>
</tr>
</tbody>
</table>

**Operating System CO16555 -- Appendix D**
## Appendix E -- DISPLAY CODES (ATASCII)

<table>
<thead>
<tr>
<th>( 0 \times 4 )</th>
<th>2X</th>
<th>4X</th>
<th>6X</th>
<th>8X</th>
<th>AX</th>
<th>CX</th>
<th>EX</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 3 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 6 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 7 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 8 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 9 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( A )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( B )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( C )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( D )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( E )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( F )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 10 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 11 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 12 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 13 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 14 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 15 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 16 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 17 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 18 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 19 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 20 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 21 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 22 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 23 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 24 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CODES 80-FF SHOW AS THE INVERSE VIDEO OF CODES 00-7F**
Appendix F -- KEYBOARD CODES (ATASCII)

<table>
<thead>
<tr>
<th>CTRL</th>
<th>SHIFT &amp; LOWER</th>
<th>SHIFT</th>
<th>LOWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>, 20</td>
<td>20 &lt;space&gt; 21</td>
<td>40 @ 35</td>
</tr>
<tr>
<td>01</td>
<td>A 3F</td>
<td>21 ! 1F</td>
<td>41 A 3F</td>
</tr>
<tr>
<td>02</td>
<td>B 15</td>
<td>22 &quot; 1E</td>
<td>42 B 15</td>
</tr>
<tr>
<td>03</td>
<td>C 12</td>
<td>23 # 1A</td>
<td>43 C 12</td>
</tr>
<tr>
<td>04</td>
<td>D 3A</td>
<td>24 $ 18</td>
<td>44 D 3A</td>
</tr>
<tr>
<td>05</td>
<td>E 2A</td>
<td>25 % 1D</td>
<td>45 E 2A</td>
</tr>
<tr>
<td>06</td>
<td>F 3B</td>
<td>26 &amp; 1B</td>
<td>46 F 3B</td>
</tr>
<tr>
<td>07</td>
<td>G 3D</td>
<td>27 ' 33</td>
<td>47 G 3D</td>
</tr>
<tr>
<td>08</td>
<td>H 39</td>
<td>28 ( 30</td>
<td>48 H 39</td>
</tr>
<tr>
<td>09</td>
<td>I 0D</td>
<td>29 ) 32</td>
<td>49 I 0D</td>
</tr>
<tr>
<td>0A</td>
<td>J 01</td>
<td>2A * 07</td>
<td>4A J 01</td>
</tr>
<tr>
<td>0B</td>
<td>K 05</td>
<td>2B + 06</td>
<td>4B K 05</td>
</tr>
<tr>
<td>0C</td>
<td>L 00</td>
<td>2C , 20</td>
<td>4C L 00</td>
</tr>
<tr>
<td>0D</td>
<td>M 25</td>
<td>2D - 0E</td>
<td>4D M 25</td>
</tr>
<tr>
<td>0E</td>
<td>N 23</td>
<td>2E . 22</td>
<td>4E N 23</td>
</tr>
<tr>
<td>0F</td>
<td>O 08</td>
<td>2F / 26</td>
<td>4F O 08</td>
</tr>
<tr>
<td>10</td>
<td>P 0A</td>
<td>30 0 32</td>
<td>50 P 0A</td>
</tr>
<tr>
<td>11</td>
<td>Q 2F</td>
<td>31 1 1F</td>
<td>51 Q 2F</td>
</tr>
<tr>
<td>12</td>
<td>R 2B</td>
<td>32 2 1E</td>
<td>52 R 2B</td>
</tr>
<tr>
<td>13</td>
<td>S 3E</td>
<td>33 3 1A</td>
<td>53 S 3E</td>
</tr>
<tr>
<td>14</td>
<td>T 2D</td>
<td>34 4 1B</td>
<td>54 T 2D</td>
</tr>
<tr>
<td>15</td>
<td>U 0B</td>
<td>35 5 1D</td>
<td>55 U 0B</td>
</tr>
<tr>
<td>16</td>
<td>V 10</td>
<td>36 6 1B</td>
<td>56 V 10</td>
</tr>
<tr>
<td>17</td>
<td>W 2E</td>
<td>37 7 33</td>
<td>57 W 2E</td>
</tr>
<tr>
<td>18</td>
<td>X 16</td>
<td>38 8 35</td>
<td>58 X 16</td>
</tr>
<tr>
<td>19</td>
<td>Y 2B</td>
<td>39 9 30</td>
<td>59 Y 2B</td>
</tr>
<tr>
<td>1A</td>
<td>Z 17</td>
<td>3A : 02</td>
<td>5A Z 17</td>
</tr>
<tr>
<td>1B</td>
<td>&lt;esc&gt; 1C</td>
<td>3B ; 0D</td>
<td>5B C 20</td>
</tr>
<tr>
<td>1C</td>
<td>^&lt;up&gt; 0E</td>
<td>3C &lt; 36</td>
<td>5C \ 06</td>
</tr>
<tr>
<td>1D</td>
<td>^&lt;down&gt; 0F</td>
<td>3D = 0F</td>
<td>5D J 22</td>
</tr>
<tr>
<td>1E</td>
<td>^&lt;left&gt; 06</td>
<td>3E &gt; 37</td>
<td>5E ^ 07</td>
</tr>
<tr>
<td>1F</td>
<td>^&lt;right&gt; 07</td>
<td>3F ? 26</td>
<td>5F _ 0E</td>
</tr>
</tbody>
</table>

80-9A / : / 00-1A 9F s<tab> 2C
9B <return> and ^3 OC, 1A AO-FC / : / 20-7C
9C s<del> 34 FD ^2 1E
9D s<insert>37 FE ^<del> 34
9E ^<tab> 2C FF ^<insert>37

<c><clear> ::= s< or ^<
<c><return> ::= <return> or s<return> or ^<return>
<esc> ::= <esc> or s<esc> or ^<esc>
<space> ::= <space> or s<space> or ^<space>

Where: s as a prefix indicates [SHIFT].
^ as a prefix indicates [CTRL].
/: as a prefix indicates ATARI key inverse active.
Appendix G -- PRINTER CODES (ATASCII)

Character set for "normal" mode printing:

<table>
<thead>
<tr>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
<th>Character</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;space&gt;</td>
<td>20</td>
<td>@</td>
<td>40</td>
<td>'</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>21</td>
<td>A</td>
<td>41</td>
<td>a</td>
<td>61</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;</td>
<td>22</td>
<td>B</td>
<td>42</td>
<td>b</td>
<td>62</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>23</td>
<td>C</td>
<td>43</td>
<td>c</td>
<td>63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>24</td>
<td>D</td>
<td>44</td>
<td>d</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>%</td>
<td>25</td>
<td>E</td>
<td>45</td>
<td>e</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;</td>
<td>26</td>
<td>F</td>
<td>46</td>
<td>f</td>
<td>66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'</td>
<td>27</td>
<td>G</td>
<td>47</td>
<td>g</td>
<td>67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>28</td>
<td>H</td>
<td>48</td>
<td>h</td>
<td>68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>)</td>
<td>29</td>
<td>I</td>
<td>49</td>
<td>i</td>
<td>69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>2A</td>
<td>J</td>
<td>4A</td>
<td>j</td>
<td>6A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>2B</td>
<td>K</td>
<td>4B</td>
<td>k</td>
<td>6B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>,</td>
<td>2C</td>
<td>L</td>
<td>4C</td>
<td>l</td>
<td>6C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>2D</td>
<td>M</td>
<td>4D</td>
<td>m</td>
<td>6D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>2E</td>
<td>N</td>
<td>4E</td>
<td>n</td>
<td>6E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/</td>
<td>2F</td>
<td>O</td>
<td>4F</td>
<td>o</td>
<td>6F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>P</td>
<td>50</td>
<td>p</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>Q</td>
<td>51</td>
<td>q</td>
<td>71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>32</td>
<td>R</td>
<td>52</td>
<td>r</td>
<td>72</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>S</td>
<td>53</td>
<td>s</td>
<td>73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>T</td>
<td>54</td>
<td>t</td>
<td>74</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>U</td>
<td>55</td>
<td>u</td>
<td>75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>V</td>
<td>56</td>
<td>v</td>
<td>76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>37</td>
<td>W</td>
<td>57</td>
<td>w</td>
<td>77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>X</td>
<td>58</td>
<td>x</td>
<td>78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>39</td>
<td>Y</td>
<td>59</td>
<td>y</td>
<td>79</td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>3A</td>
<td>Z</td>
<td>5A</td>
<td>z</td>
<td>7A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>;</td>
<td>3B</td>
<td>[</td>
<td>5B</td>
<td>{</td>
<td>7B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td>3C</td>
<td>\</td>
<td>5C</td>
<td></td>
<td></td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>=</td>
<td>3D</td>
<td>]</td>
<td>5D</td>
<td>}</td>
<td>7D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>3E</td>
<td>^</td>
<td>5E</td>
<td>~</td>
<td>7E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>?</td>
<td>3F</td>
<td>_</td>
<td>5F</td>
<td></td>
<td>7F</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The following codes print differently than defined by the ATASCII definition.

- 00 through 1F print blank.
- 60 prints ' instead of "diamond".
- 7B prints { instead of "spade".
- 7D prints } instead of "clear".
- 7E prints ~ instead of "backspace".
- 7F prints blank instead of "tab".
Character set for "sideways" mode printing:

| 40 @ | 60 @ |
| 41 A | 61 A |
| 42 B | 62 B |
| 43 C | 63 C |
| 44 D | 64 D |
| 45 E | 65 E |
| 46 F | 66 F |
| 47 G | 67 G |
| 48 H | 68 H |
| 49 I | 69 I |
| 4A J | 6A J |
| 4B K | 6B K |
| 4C L | 6C L |
| 4D M | 6D M |
| 4E N | 6E N |
| 4F O | 6F O |
| 30 0 | 50 P |
| 31 1 | 51 Q |
| 32 2 | 52 R |
| 33 3 | 53 S |
| 34 4 | 54 T |
| 35 5 | 55 U |
| 36 6 | 56 V |
| 37 7 | 57 W |
| 38 8 | 58 X |
| 39 9 | 59 Y |
| 3A : | 5A Z |
| 3B ; | 5B [ |
| 3C < | 5C \ |
| 3D = | 5D J |
| 3E > | 5E <up> |
| 3F ? | 5F <left> |

Note: the following codes print differently than defined by the ATASCII definition.

- 00 through 2F print blank.
- 5E prints "up arrow" instead of .
- 5F prints "left arrow" instead of _
- 60 through 7F repeats 40 through 5F instead of proper set.
## Appendix H -- SCREEN MODE CHARACTERISTICS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
<td>24</td>
<td>--</td>
<td>2</td>
<td>00-FF</td>
<td>BAK</td>
<td>992</td>
<td>992</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PF 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&quot;</td>
<td>PF 1*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>24</td>
<td>20</td>
<td>5</td>
<td>bgcolor. BAK</td>
<td>674</td>
<td>672</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00-3F</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40-7F</td>
<td>PF 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80-BF</td>
<td>PF 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CO-FF</td>
<td>PF 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>12</td>
<td>10</td>
<td>5</td>
<td>bgcolor. BAK</td>
<td>424</td>
<td>420</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00-3F</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>40-7F</td>
<td>PF 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>80-BF</td>
<td>PF 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CO-FF</td>
<td>PF 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>24</td>
<td>20</td>
<td>4</td>
<td>0</td>
<td>BAK</td>
<td>434</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>PF 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>PF 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>48</td>
<td>40</td>
<td>2</td>
<td>0</td>
<td>BAK</td>
<td>694</td>
<td>696</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>48</td>
<td>40</td>
<td>4</td>
<td>0</td>
<td>BAK</td>
<td>1174</td>
<td>1176</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>PF 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>PF 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>160</td>
<td>96</td>
<td>80</td>
<td>2</td>
<td>0</td>
<td>BAK</td>
<td>2174</td>
<td>2184</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>160</td>
<td>96</td>
<td>80</td>
<td>4</td>
<td>0</td>
<td>BAK</td>
<td>4190</td>
<td>4200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>PF 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>PF 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>320</td>
<td>192</td>
<td>160</td>
<td>2</td>
<td>0</td>
<td>PF 2</td>
<td>8112</td>
<td>8138</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF 1*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>80</td>
<td>192</td>
<td>--</td>
<td>1</td>
<td>Note 2</td>
<td></td>
<td>8138</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>80</td>
<td>192</td>
<td>--</td>
<td>9</td>
<td>0</td>
<td>PM 0</td>
<td>8138</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PM 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>PM 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>PM 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>PF 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OPERATING SYSTEM C016555 -- Appendix H**
Notes:

1. Uses color of PF 2, lum of PF 1.
2. Uses color of BAK, lum of data value ($0-F$).
3. Uses color of data value ($0-F$), lum of BAK.

PF x ::= Playfield color register x.
PM x ::= Player/Missile Graphics color register x.
BAK ::= Background color register (also known as PF 4).

The default values for the color registers are shown below:

BAK = $00$
PF0 = $28$
PF1 = $CA$
PF2 = $94$
PF3 = $46$
The form of a color register byte is shown below:

```
7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
| color | lum |
+-+-+-+-+-+-+-+-+
```

Where: color (hex values)  lum

<table>
<thead>
<tr>
<th>0</th>
<th>gray</th>
<th>0 = minimum luminance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>light orange</td>
<td>1 =</td>
</tr>
<tr>
<td>2</td>
<td>orange</td>
<td>2 =</td>
</tr>
<tr>
<td>3</td>
<td>red orange</td>
<td>3 = (increasing luminance)</td>
</tr>
<tr>
<td>4</td>
<td>pink</td>
<td>4 =</td>
</tr>
<tr>
<td>5</td>
<td>purple</td>
<td>5 =</td>
</tr>
<tr>
<td>6</td>
<td>purple-blue</td>
<td>6 =</td>
</tr>
<tr>
<td>7</td>
<td>blue</td>
<td>7 = maximum luminance</td>
</tr>
<tr>
<td>8</td>
<td>blue</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>light blue</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>turquoise</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>green-blue</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>green</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>yellow-green</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>orange-green</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>light orange</td>
<td></td>
</tr>
</tbody>
</table>
Appendix I -- SERIAL BUS ID AND COMMAND SUMMARY

Serial bus device IDs

Floppy diskettes    D1-D4    $31-34
Printer             P1       $40
RS-232-C            R1-R4    $50-53

Serial bus control codes

ACK - $41 ('A')
NAK - $4E ('N')
COMPLETE - $43 ('C')
ERR - $45 ('E')

Serial bus command codes

READ - $52 ('R')  Disk
WRITE - $57 ('W') Printer/Disk
STATUS - $53 ('S') Printer/Disk
PUT(no check) - $50 ('P') Disk
FORMAT - $21 ('!') Disk
READ ADDRESS - $54 ('T') Disk
READ SPIN - $51 ('Q') Disk
MOTOR ON - $55 ('U') Disk
VERIFY SECTOR - $56 ('V') Disk

OPERATING SYSTEM CO16555 -- Appendix I
Appendix J -- ROM VECTORS

The fixed address OS ROM JMP vectors are shown below; at each address is a JMP instruction to the indicated routine.

<table>
<thead>
<tr>
<th>Name</th>
<th>Addr</th>
<th>Reference Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISKIV</td>
<td>E450</td>
<td>Diskette Handler initialization</td>
</tr>
<tr>
<td>DSKINV</td>
<td>E453</td>
<td>Diskette Handler entry.</td>
</tr>
<tr>
<td>CIOV</td>
<td>E456</td>
<td>CIO utility entry.</td>
</tr>
<tr>
<td>SIOV</td>
<td>E459</td>
<td>SIO utility entry.</td>
</tr>
<tr>
<td>SETVBV</td>
<td>E45C</td>
<td>Set System Timers routine.</td>
</tr>
<tr>
<td>SYSVBV</td>
<td>E45F</td>
<td>Stage 1 VBLANK entry.</td>
</tr>
<tr>
<td>XITVBV</td>
<td>E462</td>
<td>Exit VBLANK entry.</td>
</tr>
<tr>
<td>SIOINV</td>
<td>E465</td>
<td>SIO utility initialization.</td>
</tr>
<tr>
<td>SENDEV</td>
<td>E468</td>
<td>Send enable routine.</td>
</tr>
<tr>
<td>INTINV</td>
<td>E46B</td>
<td>Interrupt Handler initialization.</td>
</tr>
<tr>
<td>CIOINV</td>
<td>E46E</td>
<td>CIO utility initialization.</td>
</tr>
<tr>
<td>BLKBDV</td>
<td>E471</td>
<td>Blackboard mode entry.</td>
</tr>
<tr>
<td>WARMSSV</td>
<td>E474</td>
<td>Warmstart ([SYSTEM. RESET]) entry.</td>
</tr>
<tr>
<td>COLDSV</td>
<td>E477</td>
<td>Coldstart (power-up) entry.</td>
</tr>
<tr>
<td>RBDKLV</td>
<td>E47A</td>
<td>Cassette-read block entry.</td>
</tr>
<tr>
<td>CSPIV</td>
<td>E47D</td>
<td>Cassette-OPEN input entry.</td>
</tr>
</tbody>
</table>

* These vectors are for OS internal use only.

The fixed address Floating Point Package ROM routine entry point addresses are shown below; complete descriptions of the corresponding routines are provided in Section 8.

<table>
<thead>
<tr>
<th>Name</th>
<th>Addr</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFP</td>
<td>D800</td>
<td>ASCII to FP convert.</td>
</tr>
<tr>
<td>FASC</td>
<td>D8E6</td>
<td>FP to ASCII convert.</td>
</tr>
<tr>
<td>IFP</td>
<td>D9AA</td>
<td>Integer to FP convert.</td>
</tr>
<tr>
<td>FPI</td>
<td>D9D2</td>
<td>FP to integer convert.</td>
</tr>
<tr>
<td>FADD</td>
<td>DA66</td>
<td>FP add.</td>
</tr>
<tr>
<td>FSUB</td>
<td>DA60</td>
<td>FP subtract.</td>
</tr>
<tr>
<td>FMUL</td>
<td>DADB</td>
<td>FP multiply</td>
</tr>
<tr>
<td>FDIV</td>
<td>DB28</td>
<td>FP divide.</td>
</tr>
<tr>
<td>LOG</td>
<td>DECD</td>
<td>FP base e logarithm.</td>
</tr>
<tr>
<td>LOG10</td>
<td>DED1</td>
<td>FP base 10 logarithm.</td>
</tr>
<tr>
<td>EXP</td>
<td>DDC0</td>
<td>FP base e exponentiation.</td>
</tr>
<tr>
<td>EXP10</td>
<td>DDCC</td>
<td>FP base 10 exponentiation.</td>
</tr>
<tr>
<td>PLYEV</td>
<td>DD40</td>
<td>FP polynomial evaluation.</td>
</tr>
<tr>
<td>ZFRO</td>
<td>DA44</td>
<td>Clear FRO.</td>
</tr>
<tr>
<td>ZF1</td>
<td>DA46</td>
<td>Clear FP number.</td>
</tr>
<tr>
<td>FLDOR</td>
<td>DD89</td>
<td>Load FP number.</td>
</tr>
<tr>
<td>FLDOP</td>
<td>DD8D</td>
<td>Load FP number.</td>
</tr>
<tr>
<td>FLD1R</td>
<td>DD98</td>
<td>Load FP number.</td>
</tr>
<tr>
<td>FLD1P</td>
<td>DD9C</td>
<td>Load FP number.</td>
</tr>
<tr>
<td>FSTOR</td>
<td>DDA7</td>
<td>Store FP number.</td>
</tr>
<tr>
<td>FSTOP</td>
<td>DDA8</td>
<td>Store FP number.</td>
</tr>
<tr>
<td>FMOVE</td>
<td>DDB6</td>
<td>Move FP number.</td>
</tr>
</tbody>
</table>
The base addresses of the Handler vectors for the resident handlers are shown below:

- Screen Editor (E)  E400
- Display Handler (S) E410
- Keyboard Handler (K) E420
- Printer Handler (P) E430
- Cassette Handler (C) E440

See Section 5 for the format of the entry for each Handler.

The 6502 Computer interrupt vector values are shown below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>FFFA</td>
<td>E7B4</td>
</tr>
<tr>
<td>RESET</td>
<td>FFAC</td>
<td>E477</td>
</tr>
<tr>
<td>IRQ</td>
<td>FFFE</td>
<td>E6FE</td>
</tr>
</tbody>
</table>
Appendix K -- DEVICE CHARACTERISTICS

This appendix describes the physical characteristics of the devices that interface to the ATARI 400 and ATARI 800 Home Computers. Where applicable, data capacity, data transfer rate, storage format, SIO interface, and cabling will be detailed.

KEYBOARD

The keyboard input rate is limited by the OS keyboard reading procedure to be 60 characters per second. The code for each key is shown in Table 5-4. The keyboard hardware has no buffering and is rate-limited by the debounce algorithm used.

DISPLAY

The television screen display generator has many capabilities that are not used by the Display Handler (as described in Section 5 and shown in Appendix H). There are additional display modes, object generators, hardware display scrolling, and many other features that are described in the ATARI Home Computer Hardware Manual.

Since all display data is stored in RAM, the display data update rate is limited primarily by the software routines that generate and format the data and access the RAM. The generation of the display from the RAM is accomplished by the ANTIc and CTIA or GTIA chips using Direct Memory Access (DMA) to access the RAM data.

The internal storage formats for display data for the various modes are detailed in the ATARI Home Computer Hardware Manual.

ATARI 410 PROGRAM RECORDER

The ATARI 410 Program Recorder has the following characteristics:

DATA CAPACITY:

100 characters per C-60 tape (unformatted).

DATA TRANSFER RATES:

* 600 Baud (60 characters per second)

* Note: The OS has the ability to adjust to different tape speeds (447 - 895 Baud).
STORAGE FORMAT:

Tapes are recorded in 1/4 track stereo format at 1 7/8 inches per second. The tape can be recorded in both directions, where tracks 1 and 2 are side A left and right; and tracks 3 and 4 are side B right and left (industry standard). On each side, the left channel (1 or 4) is used for audio and the right channel (2 and 3) is used for digital information.

The audio channel is recorded the normal way. The digital channel is recorded using the POKEY two-tone mode producing FSK data at up to 600 baud. The MARK frequency is 5327 Hz and the SPACE frequency is 3995 Hz. The transmission of data is asynchronous byte serial as seen from the computer; POKEY reads or writes a bit serial FSK sequence for each byte, in the following order:

1 start bit (SPACE)
   data bit 0 -->
   data bit 1 !
   .           0 = SPACE, 1 = MARK.
   data bit 6 !
   data bit 7 -->
1 stop bit (MARK)

The only control the computer has over tape motion is motor start/stop; and this only if the PLAY button is pressed by the user. In order for recording to take place, the user must press both the REC and PLAY buttons on the cassette. The computer has no way to sense the position of these buttons, nor even if an ATARI 410 Program Recorder is cabled to the computer, so the user must be careful when using this device.

SIO INTERFACE

The cassette device utilizes portions of the serial bus hardware, but does not follow any of the protocol as defined in Section 9.

ATARI 820(TM) 40-COLUMN IMPACT PRINTER

The ATARI 820 Printer has the following characteristics:

DATA CAPACITY:

   40 characters per line (normal printing)
   29 characters per line (sideways printing)

DATA TRANSFER RATES:
Bus rate: xx characters per second.
Print time (burst): xx characters per second.
Print time (average): xx characters per second.

**STORAGE FORMAT:**

3 7/8 inch wide paper.
5X7 dot matrix, impact printing.

Normal format --
40 characters per line.
6 lines per inch (vertical).
12 characters per inch (horizontal).

Sideways format --
29 characters per line.
6 lines per inch (vertical).
9 characters per inch (horizontal).

**SIO INTERFACE**

The controller serial bus ID is $40.

The controller supports the following SIO commands (see Section 5 for more information regarding the Handler and Section 9 for a general discussion of bus commands):

**GET STATUS**

The computer sends a command frame of the format shown below:

Device ID = $40.
Command byte = $53.
auxiliary 1 = doesn’t matter.
auxiliary 2 = doesn’t matter.
Checksum = checksum of bytes above.

The printer controller responds with a data frame of the format shown earlier in this appendix as part of the GET STATUS discussion.

**PRINT LINE**

The computer sends a command frame of the format shown below:

Device ID = $40.
Command byte = $57.
auxiliary 1 = doesn’t matter.
auxiliary 2 = $4E for normal print or $53 for sideways.
Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:

Leftmost character of line (column 1).
Next character of line (column 2).
.
Rightmost character of line (column 40 or 29).
Checksum byte.

Note that the data frame size is variable, either 41 or 30 bytes
in length, depending upon the print mode specified in the command frame.

ATARI 810 DISK DRIVE

The ATARI 810[TM] Disk Drive has the following characteristics:

DATA CAPACITY:

720 sectors of 128 bytes each (Disk Handler format).
709 sectors of 125 data bytes each (Disk File Manager format).

DATA TRANSFER RATES:

Bus rate: 1920 characters per second.
Seek time: 5.25 msec. per track + 10 to 210 msec.
Rotational latency: 104 msec maximum (288 rpm).

STORAGE FORMAT:

5 1/4 inch diskette, soft sectored by the controller.
40 tracks per diskette.
18 sectors per track.
128 bytes per sector.
Controlled by National INS1771-1 formatter/controller chip.
Sector sequence per track is: 18, 1, 3, 5, 7, 9, 11, 13, 15,
17, 2, 4, 6, 8, 10, 12, 14, 16

SIO INTERFACE

The controller serial bus IDs range from $31 (for ‘D1’) to $34
(for ‘D4’).

OPERATING SYSTEM C016555 -- Appendix K
The controller supports the following SIO commands (see earlier in this Appendix for information about the Diskette Handler and Section 9 for a general discussion of bus commands):

**GET STATUS**

The computer sends a command frame of the format shown below:

- Device ID = $31-34.
- Command byte = $53.
- auxiliary 1 = doesn't matter.
- auxiliary 2 = doesn't matter.
- Checksum = checksum of bytes above.

The diskette controller responds with a data frame of the format shown earlier in this Appendix as part of the STATUS REQUEST discussion.

**PUT SECTOR (WITH VERIFY)**

The computer sends a command frame of the format shown below:

- Device ID = $31-34
- Command byte = $57.
- auxiliary 1 = low byte of sector number.
- auxiliary 2 = high byte of sector number (1-720).
- Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:

- 128 data bytes.
- Checksum byte.

The diskette controller writes the frame data to the specified sector, then reads the sector and compares the content with the frame data. The COMPLETE byte value indicates the status of the operation.

**PUT SECTOR (NO VERIFY)**

The computer sends a command frame of the format shown below:

- Device ID = $31-34
- Command byte = $50.
- auxiliary 1 = low byte of sector number.
- auxiliary 2 = high byte of sector number (1-720).
- Checksum = checksum of bytes above.

The computer sends a data frame of the format shown below:
128 data bytes.
Checksum byte.

The diskette controller writes the frame data to the specified sector, then sends a COMPLETE byte value that indicates the status of the operation.

GET SECTOR

The computer sends a command frame of the format shown below:

- Device ID = $31-34
- Command byte = $52.
- auxiliary 1 = low byte of sector number.
- auxiliary 2 = high byte of sector number (1-720).
- Checksum = checksum of bytes above.

The diskette controller sends a data frame of the format shown below:

- 128 data bytes.
- Checksum byte.

FORMAT DISKETTE

The computer sends a command frame of the format shown below:

- Device ID = $31-34
- Command byte = $21.
- auxiliary 1 = doesn’t matter.
- auxiliary 2 = doesn’t matter.
- Checksum = checksum of bytes above.

The diskette controller completely formats the diskette (generates 40 tracks of 18 soft sectors per track with the data portion of each sector equal to all zeros) and then reads each sector to verify its integrity. A data frame of 128 bytes plus checksum is returned in that the sector numbers of all bad sectors (up to a maximum of 63 sectors) are contained, followed by two consecutive bytes of $FF. If there are no bad sectors on the diskette the first 2 bytes of the data

OPERATING SYSTEM CO16555 -- Appendix K
199
Appendix L -- OS DATA BASE VARIABLE FUNCTIONAL DESCRIPTIONS

CENTRAL DATA BASE DESCRIPTION

This appendix provides detailed information for those variables in the OS data base that can be altered by the user. Remaining variables are provided narrative descriptions. Information on the variables is presented in a multiple access scheme: Lookup tables are referenced to a common set of narratives, that is itself ordered by function.

Variable descriptions are referenced by a label called a variable identifier (VID) number. The label comprises a single letter followed by a number. A different letter is assigned for each major functional area being described, and the numbers are assigned sequentially within each functional area. Those variables that are not considered to be of interest to any user are flagged with an asterisk (*) after their names. The data base lookup tables provided are:

1. Functional grouping -- index to the function narrative and descriptions of variables, giving VID and variable name.

2. Alphabetic list of names -- giving VID of description.

3. Address ordered list -- giving VID of description.

Item 1, the functional grouping index, starts on the next page; the other two lookup tables are at the end of Appendix L.
FUNCTIONAL INDEX TO DATA BASE VARIABLE DESCRIPTIONS

A. Memory configuration
   A1 MEMLO
   A2 MEMTOP
   A3 APPMHI
   A4 RAMTOP
   A5 RAMSIZ

B. Text/graphics screen
   Cursor control
      B1 CRSINH
      B2 ROWCRS, COLCRS
      B3 OLDROW, OLDCOL
      B4 TXTROW, TXTCOL
   Screen margins
      B5 LMARGN
      B6 RMARGN
   Color control
      B7 PCOLRO - PCOLR3
      B8 COLORO - COLOR4
   Text scrolling
      B9 SCRFLG*
   Attract mode
      B10 ATRACT
      B11 COLRSH*
      B12 DRKMSK*
   Tabbing
      B13 TABMAP
   Logical text lines
      B14 LOGMAP*
      B15 LOGCOL*
   Split screen
B16 BOTSCR*

FILL/DRAW function
B17 FILDAT
B18 FILFLG*
B19 NEW SROW*, NEWCOL*
B20 HOLD4*

B21 ROWINC*, COLINC*
B22 DELTAR*, DELTAC*
B23 COUNTR*
B24 ROWAC*, COLAC*
B25 ENDPT*

Displaying control characters

Escape (display following control char)
B26 ESCFLG*

Display control characters mode
B27 DSPFLG

Bit mapped graphics
B28 DMASK*
B29 SHFAMT*
Internal working variables
B30 HOLD1*
B31 HOLD2*
B32 HOLD3*
B33 TMPCHR*
B34 DSTAT*
B35 DINDEX
B36 SAVMSC
B37 OLDCHR*
B38 OLDADR*
B39 ADRESS*
B40 MLTTMP/OPNTMP/TOADR*
B41 SAVADR/FRMADR*
B42 BUFCNT*
B43 BUFSTR*
B44 SWPFLG*
B45 INSDAT*
B46 TMPROW*, TMPCOL*
B47 TMPLBT*
B48 SUBTMP*
B49 TINDEX*
B50 BITMSK*
B51 LINBUF*
B52 TXTMSC
B53 TXTOLD*
Internal character code conversion
854 ATACHR
B55 CHAR*

C. Disk Handler
C1 BUFADR*
C2 DSKTIM*

D. Cassette (part in SIO part in Handler)
Baud rate determination
D1 CBAUDL*, CBAUDH*
D2 TIMFLG*
D3 TIMER1*, TIMER2*
D4 ADDCOR*
D5 TEMP1*
D6 TEMP3*
D7 SAVIO*

Cassette mode
D8 CASFLG*

Cassette buffer
D9 CASBUF*
D10 BLIM*
D11 BPTR*

Internal working variables
D12 FEOF*
D13 FTYPE*
D14 WMODE*
D15 FREG*

E. Keyboard
Key reading and debouncing
E1 CH1*
E2 KEYDEL*
E3 CH
Special functions

Start/stop
E4 SSFLAG

[BREAK]
E5 BRKKEY

[SHIFT]/[CONTROL] lock
E6 SHFLOK
E7 HOLDCH*

Autorepeat
E8 SRTIMR*

Inverse video
E9 INVFLG

Console switches ([SELECT], [START], and [OPTION])

F. Printer

printer-buffer
F1 PRNBUF*
F2 PBUFSZ*
F3 PBPNST*

Internal working variables
F4 PTEMP*
F5 PTIMOT*

G. Central I/O routine (CIO)

User call parameters
G1 IOCBI
G2 ICHID
G3 ICDNO
G4 ICCOM
G5 ICSTA
G6 ICBAI, ICBAH
G7 ICPTL, ICPTH
G8 ICBLI, ICBLH
G9 ICAXI, ICAX2
G10 ICSPR

Device status
G11 DVSTAT

device table
G12 HATABS

OPERATING SYSTEM CO16555 -- Appendix L
CIO/Handler interface Parameters

G13 ZIOCB (IOCBAS)
  G14 ICHIDZ
  G15 ICDNOZ
  G16 ICCOMZ
  G17 ICSTAZ
  G18 ICBALZ,ICBALH
  G19 ICPTLZ,ICPTHZ
  G20 ICBLLZ,ICBLHZ
  G21 ICAX1Z,ICAX2Z
  G22 ICSPRZ (ICIDNO,CIOCHR)

Internal working variables
  G23 ICCOMT*
  G24 ICIDNO*
  G25 CIOCHR*

H. Serial I/O routine (SID)

User call parameters

H1 DCB control block
  H2 DDEVIC
  H3 DUNIT
  H4 DCOMND
  H5 DSTATS
  H6 DBUFL0, DBUFI
  H7 DTIMLO
  H8 DBYTL0, DBYTHI
  H9 DAUX1, DAUX2

Bus sound control
  H10 SOUND

Serial bus control

Retry logic
  H11 CRETRY*
  H12 DRETRY*

Checksum
  H13 CHKSUM*
  H14 CHKSNT*
  H15 NOCKSM*
Data buffering

General buffer control

H16 BUFRLO*, BUFRHI*
H17 BFENLO*, BFENHI*

Command frame output buffer

H18 CDEVIC*
H19 CCOMND*
H20 CAUX1*, CAUX2*

Receive/transmit data buffering

H21 BUFRFL*
H22 RECVDN*
H23 TEMP*
H24 XMTDDON*

S10 timeout

H25 TIMFLG*
H26 CDTMV1*
H27 CDTMA1*

Internal working variables

H28 STACKP*
H29 TSTAT*
H30 ERRFLG*
H31 STATUS*
H32 SSKCTL*

J. ATARI controllers

Joysticks

J1 STICKO - STICK3
J2 STRIGO - STRIG3

Paddles

J3 PADDLO - PADDL7
J4 PTRIGO - PTRIG7

Paddle controllers

J8 STICKO - STICK3
J9 STRIGO - STRIG3

K. Disk file manager

K1 FMSZPG*
K2 ZBUFZ*
K3 ZDRVA*
K4 ZSBA*
K5 ERRNO*

OPERATING SYSTEM CO16555 --- Appendix L
L. Disk utilities (DOS)
   L1 DSKUTL*

M. Floating point package
   M1 FRO
   M2 FRE*
   M3 FR1
   M4 FR2*
   M5 FRX*
   M6 EEXP*
   M7 NSIGN*
   M8 ESIGN*
   M9 FCHRFLG*
   M10 DIGRT*
   M11 CIX
   M12 INBUFF
   M13 ZTEMP1*
   M14 ZTEMP4*
   M15 ZTEMP3*
   M16 FLPTR
   M17 FPTR2*
   M18 LBPR1*
   M19 LBPR2*
   M20 LBUFF
   M21 PLYARG*
   M22 FPSCR/FSCR*
   M23 FPSCR1/FSCR1*
   M24 DEGFLG/RADFLG*

N. Power-Up and System Reset
   RAM sizing
      N1 RAMLD*, TRAMSZ*
      N2 TSTDAT*

   Diskette/cassette-boot
      N3 DOSINI
      N4 CKEY*
      N5 CASSBT*
      N6 CASINI
      N7 BOOT?*
      N8 DFLAGS*
      N9 DBSECT*
      N10 BOOTAD*

   Environmental control
      N11 COLDST
      N12 DOSVEC
P. Interrupts

P1 CRITIC
P2 POKMSK

System Timers

Real-time clock
P3 RTCLOK

System Timer 1
P4 CDTMV1
P5 CDTMA1

System Timer 2
P6 CDTMV2
P7 CDTMA2

System Timers 3-5
P8 CDTMV3, CDTMV4, CDTMV5
P9 CDTMF3, CDTMF4, CDTMF5

RAM-interrupt vectors

NMI-interrupt vectors
P10 VDSLST
P11 VVBLKI
P12 VVBLKD

IRQ-interrupt vectors
P13 VIMIRQ
P14 VPRCED
P15 VINTER
P16 VBREAK
P17 VKEYBD
P18 VSERIN
P19 VSEROR
P20 VSEROC
P21 VTIMR1, VTIMR2, VTIMR4

Hardware register updates
P22 SDMCTL
P23 SDLSTL, SDLSTH
P24 GPRIOR
P25 CHACT
P26 CHBAS
P27 PCOLRx, COLORx

OPERATING SYSTEM CO16555 -- Appendix L

209
Internal working variable
P28 INTEMP*

R. User areas
R1 (unlabeled)
R2 USAREA

This appendix contains descriptions of many of the data base variables; descriptions are included for all of the user-accessible variables and for some of the "internal" variables as well. Those variables that are not considered to be normally of interest to any user are flagged with an asterisk (*) after their names; the other variables can be of interest to one or more of the following classes of users:

- End user.
- Game developer.
- Applications programmer.
- System utility writer.
- Language processor developer.
- Device Handler Writer.

Each variable is specified by its system equate file name followed by its address (in hex) and the number of bytes reserved in the data base (in decimal), in the following form:

\(<name>\ [\langle address>,<size>]\)

For example:

MEMLO [02E7,2]

Note that most word (2 byte) variables are ordered with the least significant byte at the lower address.
A. MEMORY CONFIGURATION

See Section 4 for a general discussion of memory dynamics and section 7 for details of system initialization.

A1 MEMLO [02E7,2] -- User-free memory low address

MEMLO contains the address of the first location in the free memory region. The value is established by the OS during power-up and system reset initialization and is never altered by the OS thereafter.

A2 MEMTOP [02E5,2] -- User-free memory high address

MEMTOP contains the address of the first non-useable memory location above the free memory region. The value is established by the OS during power-up and system reset initialization; and then is re-established whenever the display is opened, based upon the requirements of the selected graphics mode.

A3 APPMHI [00OE,2] -- User-free memory screen lower limit

APPMHI is a user-controlled variable that contains the address within the free memory region below which the Display Handler cannot go in setting up a display screen. This variable is initialized to zero by the OS at power-up.

A4 RAMTOP* [006A,1] -- Display Handler top of RAM address (MSB)

RAMTOP permanently retains the RAM top address that was contained in TRAMSZ (as described in N1) for the Display Handler’s use. The value is set up as part of Handler initialization.

A5 RAMSIZ [02E4,1] -- Top of RAM address (MSB only)

RAMSIZ permanently retains the RAM top address that was contained in TRAMSZ (as described in N1).
B. TEXT/GRAPHICS SCREEN

See Section 5 for a discussion of the text and graphics screens and their Handlers.

Cursor Control

For the text screen and split-screen text window there is a visible cursor on the screen which shows the position of the next input or output operation. The cursor is represented by inverting the video of the character upon which it resides; but the cursor can be made invisible, at the user's option. The graphics screen always has an invisible cursor.

The cursor position is sensed by examining data base variables and can be moved by altering those same variables; in addition, when using the Screen Editor, there are cursor movement control codes that can be sent as data (as explained in Section 5).

B1 CRSINH [02FO,1] -- Cursor display inhibit flag

When CRSINH is zero, all outputs to the text screen will be followed by a visible cursor (inversed character); and when CRSINH is nonzero, no visible cursor will be generated.

CRSINH is set to zero by power-up, the [SYSTEM.RESET] or [BREAK] keys or an OPEN command to the Display Handler or Screen Editor.

Note that altering CRSINH does not cause the visible cursor to change states until the next output to the screen; if an immediate change to the cursor state is desired, without altering the screen data, follow the CRSINH change with the output of CURSOR UP, CURSOR DOWN, or some other innocuous sequence.

B2 ROWCRS [0054,1] and COLCRS [0055,2] -- Current cursor position

ROWCRS and COLCRS define the cursor location (row and column, respectively) for the next data element to be read from or written to the main screen segment. When in split-screen mode, the variables TXTROW and TXTCOL define the cursor for the text window at the bottom of the screen as explained in B4 below.

The row and column numbering start with the value zero, and increase in increments of one to the number of rows or columns minus 1; with the upper left corner of the screen being the origin (0,0).

ROWCRS is a single-byte variable with a maximum allowable value of 191 (screen modes 8-11); COLCRS is a 2-byte variable with a maximum allowable value of 319 (screen mode 8).
B3 OLDROW [005A,1] and OLDCOL [005B,2] -- Prior cursor position

OLDROW and OLDCOL are updated from ROWCRS and COLCRS before every operation. The variables are used only for the DRAW and FILL operations.

B4 TXTROW [0290,1] and TXTCOL [0291,2] -- Split-screen text cursor position

TXTROW and TXTCOL define the cursor location (row and column, respectively) for the next data element to be read from or written to the split-screen text window.

The row and column numbering start with the value zero, and increase in increments of one to 3 and 39, respectively, with the upper left corner of the split-screen text window being the origin (0,0).

Screen Margins

The text screen and split-screen text window have user-alterable left and right margins that define the normal domain of the text cursor.

B5 LMARGN [0052,1] -- Text column left margin

LMARGN contains the column number (0-39) of the text screen left margin; the text cursor will remain on or to the right of the left margin as a result of all operations, unless the cursor column variable is directly updated by the user (see B2 and B4 above). The default value for LMARGN is 2 and is established upon power-up or system reset.

B6 RMARGN [0053,1] -- Text column right margin

RMARGN contains the column number (0-39) of the text screen right margin; the text cursor will remain on or to the left of the right margin as a result of all operations, unless the cursor column variable is directly updated by the user (see B2 and B4 above). The default value for RMARGN is 39 and is established upon power-up or system reset.
Color Control

As part of the stage 2 VBLANK process (see Section 6), the values of nine database variables are stored in corresponding hardware color control registers. The color registers are divided into two groups: the player/missile colors and the playfield colors. The playfield color registers are utilized by the different screen modes as shown in Appendix H. The player/missile color registers are not used by the standard OS.

B7 PCOLRO - PCOLR3 [02C0,4] -- Player/missile graphics colors

Each color variable is stored in the corresponding hardware register as shown below:

PCOLRO [02C0]      COLPMO [D012]
PCOLR1 [02C1]      COLPM1 [D013]
PCOLR2 [02C2]      COLPM2 [D014]
PCOLR3 [02C3]      COLPM3 [D015]

Each color variable has the format shown below:

7 6 5 4 3 2 1 0
+-----+-----+-----+-----+
\color / \lum / \ix/ 
+-----+-----+-----+-----+

See Appendix H for information regarding the color and luminance field values.

B8 COLORO - COLOR4 [02C5,5] -- Playfield colors

Each color variable is stored in the corresponding hardware register as shown below:

COLORO [02C4]      COLPFO [D016]
COLOR1 [02C5]      COLPF1 [D017]
COLOR2 [02C6]      COLPF2 [D018]
COLOR3 [02C7]      COLPF3 [D019]
COLOR4 [02C8]      COLBK [D01A]

Each color variable has the format shown below:

7 6 5 4 3 2 1 0
+-----+-----+-----+-----+
\color / \lum / \ix/ 
+-----+-----+-----+-----+

See Appendix H for information regarding the color and luminance field values.
Text Scrolling

The text screen or split-screen text window "scrolls" upward whenever one of the two conditions shown below occurs:

- A text line at the bottom row of the screen extends past the right margin.
- A text line at the bottom row of the screen is terminated by an EOL.

Scrolling has the effect of removing the entire logical line that starts at the top of the screen and then moving all subsequent lines upward to fill in the void. The cursor will also move upward if the logical line deleted exceeds one physical line.

B9 SCRFLG* [02BB,1] -- Scroll flag

SCRFLG is a working variable that counts the number of physical lines minus 1 that were deleted from the top of the screen; since a logical line ranges in size from 1 to 3, SCRFLG ranges from 0 to 2.

Attract Mode

Attract mode is a mechanism that protects the television screen from having patterns "burned into" the phosphors due to a fixed display being left on the screen for extended periods of time. When the computer is left unattended for more than 9 minutes, the color intensities are limited to 50 percent of maximum and the hues are continually varied every 8.3 seconds. Pressing any keyboard data key will be sufficient to remove the attract mode for 9 more minutes.

As part of the stage 2 VBLANK process, the color registers from the database are sent to the corresponding hardware color registers; before they are sent, they undergo the following transformation:

\[ \text{hardware register} = \text{database variable XOR COLRSH AND DRKMSK} \]

Normally COLRSH = $00$ and DRKMSK = $FE$, thus making the above calculation a null operation; however, once attract mode becomes active, COLRSH = the content of RTCLOK+1 and DRKMSK = $F6$, that has the effect of modifying all of the colors and keeping their luminance always below the 50 percent level.

Since RTCLOK+1 is incremented every 256/60 of a second and since the least significant bit of COLRSH is of no consequence, a
color/lum change will be effected every 8.3 seconds (512/60).

B10 ATRACT [004D, 1] -- Attract mode timer and flag

ATRACT is the timer (and flag) that controls the initiation and termination of attract mode. Whenever a keyboard key is pressed, the keyboard IRQ service routine sets ATRACT to zero, thus terminating attract mode; the [BREAK] key logic behaves accordingly. As part of the stage 1 VBLANK process, ATRACT is incremented every 4 seconds; if the value exceeds 127 (after 9 minutes without keyboard activity), the value of ATRACT will be set to $FE and will retain that value until attract mode is terminated.

Since the attract mode is prevented and terminated by the OS based only upon keyboard activity, some users can want to reset ATRACT based upon Atari-controller event detection, user-controlled Serial I/O bus activity or any other signs of life.

B11 COLRSH* [004F, 1] -- Color shift mask

COLRSH has the value $00 when attract mode is inactive, thus effecting no change to the screen colors; when attract mode is active, COLRSH contains the current value of the timer variable middle digit (RTCLOK+1).

B12 DRKMSK* [004E, 1] -- Dark (luminance) mask

DRKMSK has the value $FE when attract mode is inactive, which does not alter the luminance; and has the value $F6 when attract mode is active, which forces the most significant bit of the luminance field to zero, thus guaranteeing that the luminance will never exceed 50 percent.

Tabbing

See Section 5 for a discussion of the use of tabs in conjunction with the Screen Editor.

B13 TABMAP [02A3, 15] -- Tab stop setting map

The tab settings are retained in a 15-byte (120 bit) map, where a bit value of 1 indicates a tab setting; the diagram below shows the mapping of the individual bits to tab positions.
Whenever the Display Handler or Screen Editor is opened, this map is initialized to contain the value of $01$ in every byte, thus providing the default tab stops at 7, 15, 23, etc.

Logical Text Lines

The text screen is invisibly divided into logical lines of text, each comprising from one to three physical lines of text. The screen is initialized to 24 logical lines of one physical line each; but data entry and/or data insertion can increase the size of a logical line to two or three physical lines.

B14 LOGMAP* [02B2,4] -- Logical line starting row map

The beginning physical line number for each logical line on the screen is retained in a four byte (32 bit) map, where a bit value of one indicates the start of a logical line; the diagram below shows the mapping of the individual bits to physical line (row) numbers.

The map bits are all set to 1 whenever the text screen is opened or cleared. From that point, the map is updated as logical lines are entered, edited and deleted from the screen.
LOGCOL* [0063,1] -- Cursor/logical line column number

LOGCOL contains the logical-line column number for the current cursor position; note that a logical line can comprise up to three physical lines. This variable is for the internal use of the Display Handler.

Split Screen

The Display Handler and Screen Editor together support the operation of a split-screen mode (see Section 5) in which the main portion of the screen is in one of the graphics modes and is controlled by the Display Handler, and there are 4 physical lines in the text window at the bottom of the screen which is controlled by the Screen Editor.

BOTSCR* [02BF,1] -- Text screen lines count

BOTSCR contains the number of lines of text for the current screen: 24 for mode 0 or 4 for a split-screen mode. The Handler also uses this variable as an indication of the split-screen status; tests are made for the specific values 4 and 24.

DRAW/FILL Function

The DRAW function line drawing algorithm is shown below translated to the PASCAL language from assembly language.

```
NEWROW := ROWCRS; NEWCOL := COLCRS;
DELTAR := ABS (NEWROW-OLDROW);
ROWINC := SIGN (NEWROW-OLDROW); { +1 or -1 }
DELTAC := ABS (NEWCOL-OLDCOL);
COLINC := SIGN (NEWCOL-OLDCOL); { +1 or -1 }
ROWAC := 0; COLAC := 0;
ROWCRS := OLDROW; COLCRS := OLDCOL;
COUNTR := MAX (DELTAC,DELTAR);
ENDPT := COUNTR;
IF COUNTR = DELTAC
  THEN ROWAC := ENDPT DIV 2 
  ELSE COLAC := ENDPT DIV 2;
WHILE COUNTR > 0 DO 
  BEGIN
```

OPERATING SYSTEM CO16555 -- Appendix L
ROWAC := ROWAC + DELTAR;
IF ROWAC >= ENDPT
  THEN
    BEGIN
      ROWAC := ROWAC - ENDPT;
      ROWCRS := ROWCRS + ROWINC
    END;

COLAC := COLAC + DELTAC;
IF COLAC >= ENDPT
  THEN
    BEGIN
      COLAC := COLAC - ENDPT;
      COLCRS := COLCRS + COLINC
    END;

PLOT_POINT; { point defined by ROWCRS and COLCRS }
IF FILFLG <> 0 THEN FILL_LINE;
COUNTR := COUNTR - 1
END;

The FILL function algorithm (FILL_LINE above) is described briefly in Section 5.

B17 FILDAT [02FD,1] -- Fill data
FILLDAT contains the fill region data value as part of the calling sequence for a FILL command as described in Section 5.

B18 FILFLG* [02B7,1] -- Fill flag
FILFLG indicates to the shared code within the Display Handler whether the current operation is FILL (FILFLG <> 0) or DRAW (FILFLG = 0).

B19 NEWROW* [0060,1] and NEWCOL* [0061,2] -- Destination point
NEWROW and NEWCOL are initialized to the values in ROWCRS and COLCRS, which represent the destination endpoint of the DRAW/FILL command. This is done so that ROWCRS and COLCRS can be altered during the performance of the command.

B20 HOLD4* [02BC,1] -- Temporary storage
HOLD4 is used to save and restore the value in ATACHR during the FILL process; ATACHR is temporarily set to the value in FILDAT to accomplish the filling portion of the command.

B21 ROWINC* [0079,1] and COLINC* [007A,1] -- Row/column increment/decrement

ROWINC and COLINC are the row and column increment values; they are each set to +1 or -1 to control the basic direction of line drawing. ROWINC and COLINC represent the signs of NEWROW - ROWCRS and NEWCOL - COLCRS, respectively.

B22 DELTAR* [0076,1] and DELTAC* [0077,2] -- Delta row and delta column

DELTAR and DELTAC contain the absolute values of NEWROW - ROWCRS and NEWCOL - COLCRS, respectively; together with ROWINC and COLINC, they define the slope of the line to be drawn.

B23 COUNTR* [007E,2] -- Draw iteration count

COUNTR initially contains the larger of DELTAR and DELTAC, that is the number of iterations required to generate the desired line. COUNTR is then decremented after every point on the line is plotted, until it reaches a value of zero.

B24 ROWAC* [0070,2] and COLAC* [0072,2] -- Accumulators

ROWAC and COLAC are working accumulators that control the row-and column-point plotting and increment (or decrement) function.

B25 ENDPT* [0074,2] -- Line length

ENDPT contains the larger of DELTAR and DELTAC, and is used in conjunction with ROWAC/COLAC and DELTAR/DELTAC to control the plotting of line points.

Displaying Control Characters

Often it is useful to have ATASCII control codes (such as CLEAR, CURSOR UP, etc.) displayed in their graphic forms instead of having them perform their control function. This display capability is provided in two forms when outputting to the Screen Editor: 1) a data content form in which a special character (ESC) precedes each control character to be displayed and 2) a mode control form.
Escape (Display Following Control Character)

Whenever an ESC character is detected by the Screen Editor, the next character following this code is displayed as data, even if it would normally be treated as a control code; the EOL code is the sole exception. It is always treated as a control code. The sequence ESC ESC will cause the second ESC character to be displayed.

B26 ESCFLG* [02A2,1] -- Escape flag

ESCFLG is used by the Screen Editor to control the escape sequence function; the flag is set (to $80) by the detection of an ESC character ($1B) in the data stream and is reset (to 0) following the output of the next character.

Display Control Characters Mode

When it is desired to display ATASCII control codes other than EOL in their graphics form, but not have an ESC character associated with each control code, a display mode can be established by setting a flag in the data base. This capability is used by language processors when displaying high-level language statements, that can contain control codes as data elements.

B27 DSPFLG [02FE,1] -- Display control characters flag

When DSPFLG is nonzero, ATASCII control codes other than EOL are treated as data and displayed on the screen when output to the Screen Editor. When DSPFLG is zero, ATASCII control codes are processed normally.

DSPFLG is set to zero by Power-up and [SYSTEM.RESET].

Bit-Mapped Graphics

A number of temporary variables are used by the Display Handler when handling data elements (pixels) going to or from the screen; of interest here are those variables that are used to control the packing and unpacking of graphics data, where a memory byte typically contains more than one data element (for example, screen mode 8 contains 8 pixels per memory byte).

B28 DMASK* [02A0,1] -- Pixel location mask
DMASK is a mask that contains zeros for all bits that do not correspond to the specific pixel to be operated upon, and 1’s for all bits that do correspond. DMASK can contain the values shown below in binary notation:

- 11111111 -- screen modes 1 and 2; one pixel per byte.
- 11110000 -- screen modes 9-11; two pixels per byte.
- 11000000 -- screen modes 3, 5 and 7; four pixels per byte.
- 01100000
- 01100000
- 00001100
- 00000011
- 00000000
- 01000000
- 00000010
- 00000001

B29 SHFAMT* [006F,1] -- Pixel justification

SHFAMT indicates the amount to shift the right-justified pixel data on output, or the amount to shift the input data to right justify it on input. The value is always the same as for DMASK prior to the justification process.

Internal Working Variables

B30 HOLD1* [0051,1] -- Temporary storage
B31 HOLD2* [029F,1] -- Temporary storage
B32 HOLD3* [029D,1] -- Temporary storage
B33 TMPCHR* [0050,1] -- Temporary storage
B34 DSTAT* [004C,1] -- Display status
B35 DINDEX [0057,1] -- Display mode

DINDEX contains the current screen mode obtained from the low order four bits of the most recent OPEN AUX1 byte.

B36 SAVMSC [0058,2] -- Screen Memory Address

SAVMSC contains the lowest address of the screen data region; the data at that address is displayed at the upper left corner of the screen.
B37  OLDCHR* [005D,1] -- Cursor character save/restore

OLDCHR retains the value of the character under the visible text cursor; this variable is used to restore the original character value when the cursor is moved.

B38  OLDADR* [005E,2] -- Cursor memory address

OLDADR retains the memory address of the current visible text cursor location; this variable is used in conjunction with OLDCHR (B37) to restore the original character value when the cursor is moved.

B39  ADRESS* [0064,2] -- Temporary storage

B40  MLTTMP/OPNTMP/TOADR* [0066,2] -- Temporary storage

B41  SAVADR/FRMADR* [0068,2] -- Temporary storage

B42  BUFCNT* [006B,1] -- Screen Editor current logical line size

B43  BUFSTR* [006C,2] -- Temporary storage

B44  SWPFLG* [007B,1] -- Split-screen cursor control

In split-screen mode, the graphics cursor data and the text window cursor data are frequently swapped as shown below in order to get the variables associated with the region being accessed into the ROWCRS-OLDADR variables.

ROWCRS B2 ------ TXTROW B4
COLCRS B2 ------ TXTCOL B4
DINDEX B35 ------ TINDEX B49
SAVMSC B36 ------ TXTMSC B52
OLDROW B3 ------ TXTOLD B53
OLDCOL B3 ------ " "
OLDCHR B37 ------ " "
OLDADR B38 ------ " "

SWPFLG is used to keep track of what data set is currently in the ROWCRS-OLDADR region; SWPFLG is equal to $FF when split-screen text window cursor data is in the main region, otherwise SWPFLG is equal to 0.

B45  INSDAT* [007D,1] -- Temporary storage
B46 TMPROW* [02BB,1] and TMPCOL* [02B9,2] -- Temporary storage

B47 TMPLBT* [02A1,1] -- Temporary storage

B48 SUBTMP* [029E,1] -- Temporary storage

B49 TINDEX* [0293,1] -- Split screen text window screen mode

TINDEX is the split-screen text window equivalent of DINDEX and is always equal to zero when SWPFLG is equal to zero (see B44).

B50 BITMSK* [006E,1] -- Temporary storage

B51 LINBUF* [0247,40] -- Physical line buffer

LINBUF is used to temporarily buffer one physical line of text when the Screen Editor is moving screen data.

B52 TXTMSC [0294,2] -- Split screen memory address

TXTMSC is the split-screen text window version of SAVMSC (B36).

See B44 for more information.

B53 TXTOLD* [0296,6] -- Split screen cursor data

See B44 for more information.

Internal Character Code Conversion

Two variables are used to retain the current character being processed (for both reading and writing); ATACHR contains the value passed to or from CIQ, and CHAR contains the internal code corresponding to the value in ATACHR. Because the hardware does not interpret ATASCII characters directly, the transformations shown below are applied to all text data read and written:

<table>
<thead>
<tr>
<th>ATASCII CODE</th>
<th>INTERNAL CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-1F</td>
<td>40-5F</td>
</tr>
<tr>
<td>20-3F</td>
<td>00-1F</td>
</tr>
<tr>
<td>40-5F</td>
<td>20-3F</td>
</tr>
<tr>
<td>60-7F</td>
<td>60-7F</td>
</tr>
<tr>
<td>80-9F</td>
<td>CO-DF</td>
</tr>
</tbody>
</table>

OPERATING SYSTEM CO16555 -- Appendix L
See P26 for more information.

B54 ATACHR [02FB,1] -- Last ATASCII character or plot point

ATACHR contains the ATASCII value for the most recent character read or written, or the value of the graphics point. This variable can also be considered to be a parameter of the FILL/DRAW commands, as the value in ATACHR will determine the line color when a DRAW or FILL is performed.

B55 CHAR* [02FA,1] -- Internal character code

CHAR contains the internal code value for the most recent character read or written.

C. DISKETTE HANDLER

See Section 5 for a discussion of the resident Diskette Handler.

C1 BUFADR* [0015,2] -- Data buffer pointer

BUFADR acts as temporary page zero pointer to the current diskette buffer.

C2 DSKTIM* [024B,1] -- Disk format operation timeout time

DSKTIM contains the timeout value for SIO calling sequence variable DTILO (see Section 9). DSKTIM is set to 160 (which represents a 171-second timeout) at initialization time, and is updated after each diskette status request operation. It contains the value returned in the third byte of the status frame (see Section 5). Note that all diskette operations other than format have a fixed (7) second timeout, established by the Diskette Handler.

D. CASSETTE

See Section 5 for a general description of the Cassette Handler. The cassette uses the Serial I/O bus hardware, but does not conform with the Serial I/O bus protocol as defined in Section 9. Hence, the Serial
I/O utility (SIO) has cassette specific code within it. Some variables in this subsection are utilized by SIO and some by the Cassette Handler.

Baud Rate Determination

The input baud rate is assumed to be a nominal 600 baud, but will be adjusted, if necessary, by the SIO routine to account for drive-motor variations, stretched tape, etc. The beginning of every cassette record contains a pattern of alternating 1's and zeros that is used solely for speed correction; by measuring the time to read a fixed number of bits, the true-receive baud rate is determined and the hardware adjusted accordingly. Input baud rates ranging from 318 to 1407 baud can theoretically be handled using this technique.

The input baud rate is adjusted by setting the POKEY counter that controls the bit sampling period.

D1 CBAUDL* [02EE,1] and CBAUDH* [02EF,1] -- Cassette baud rate

Initialized to 05CC hex, which represents a nominal 600 baud. After baud rate calculation, these variables will contain POKEY counter values for the corrected baud rate.

D2 TIMFLG* [0317,1] -- Baud rate determination timeout flag

TIMFLG is used by SIO to timeout an unsuccessful baud rate determination. The flag is initially set to 1, and if it attains a value of zero (after 2 seconds) before the first byte of the cassette record has been read, the operation will be aborted. See also H24.

D3 TIMER1* [030C,2] and TIMER2* [0310,2] -- Baud rate timers

These timers contain reference times for the beginning and end of the fixed bit pattern receive period. The first byte of each timer contains the then current vertical line counter value read from ANTIC, and the second byte of each timer contains the then current value of the least significant byte of the OS real time clock (RTCLOK+2).

The difference between the timers is converted to raster pair counts and is then used to perform a table lookup with interpolation to determine the new values for CBAUDL and CBAUDH.

D4 ADDCOR* [030E,1] -- Interpolation adjustment variable
AODCOR is a temporary variable used for the interpolation calculation of the above computation.

D5  TEMP1* [0312,2] -- Temporary storage

D6  TEMP3* [0315,1] -- Temporary storage

D7  SAVIO* [0316,1] -- Serial in data detect

SAVIO is used to retain the state of SKSTAT [D20F] bit 4 (serial data in); it is used to detect (and is updated after) every bit arrival.

Cassette Mode

D8  CASFLG* [030F,1] -- Cassette I/O flag

CASFLG is used internally by SIO to control the program flow through shared code. A value of zero indicates that the current operation is a standard Serial I/O bus operation, and a nonzero value indicates a cassette operation.

Cassette Buffer

D9  CASBUF* [03FD,131] -- Cassette record buffer

CASBUF is the buffer used by the Cassette Handler for the packing and unpacking of cassette-record data, and by the initialization cassette-boot logic. The format for the standard cassette record in the buffer is shown below:

```
|   7 6 5 4 3 2 1 0   |
|   ++++++++++++++++++ |
| 0 1 0 1 0 1 0 1!    |
|   ++++++++++++++++++ |
| 0 1 0 1 0 1 0 1!    |
|   ++++++++++++++++++ |
| control byte !      |
|   ++++++++++++++++++ |
| 12B !               |
| = data =           |
| ! bytes !           |
|   ++++++++++++++++++ |
```

CASBUF+0

CASBUF+1

CASBUF+2

CASBUF+3

CASBUF+130

See Section 5 for an explanation of the standard cassette-record format.
D10  **BLIM** [028A,1] -- Cassette record data size

BLIM contains the count of the number of data bytes in the current cassette record being read. BLIM will have a value ranging from 1 to 128, depending upon the record control byte as explained in Section 5.

D11  **BPTR** [003D,1] -- Cassette-record data index

BPTR contains an index into the data portion of the cassette record being read or written. The value will range from 0 to the then current value of BLIM. When BPTR equals BLIM then the buffer (CASBUF) is full if writing or empty if reading.

**Internal Working Variables**

D12  **FEOF** [003F,1] -- Cassette end-of-file flag

FEOF is used by the Cassette Handler to flag the detection of an end of file condition (control byte = $FE). FEOF equal to zero indicates that an EOF has not yet been detected, and a nonzero value indicates that an EOF has been detected. The flag is reset at every OPEN.

D13  **FTYPE** [003E,1] -- Interrecord gap type

FTYPE is a copy of ICAX2Z from the OPEN command and indicates the type of interrecord gap selected: a positive value indicates normal record gaps, and a negative value indicates continuous mode gaps.

D14  **WMODE** [0289,1] -- Cassette read/write mode flag

WMODE is used by the Cassette Handler to indicate whether the current operation is a read or write operation; a value of zero indicates read, and a value of $80 indicates write.

D15  **FREQ** [0040,1] -- Beep count

FREQ is used to retain and count the number of beeps requested of the BEEP routine by the Cassette Handler during the OPEN command process.
E. KEYBOARD

See Section 5 for a general description of the Keyboard Handler.

Key Reading and Debouncing

The console key code register is read in response to an IRG interrupt that is generated whenever a key stroke is detected by the hardware. The key code is compared with the prior key code accepted (CH1); if the codes are not identical, then the new code is accepted and stored in the key code FIFO (CH) and in the prior key code variable (CH1). If the codes are identical, then the new code is accepted only if a suitable key debounce delay has transpired since the prior value was accepted.

If the key code read and accepted is the code for \[\text{CTRL} \] 1, then the display start/stop flag (SSFLAG) is complemented and the value is not stored in the key code FIFO (CH).

In addition to the reading of the key data, SRTIMR is set to $30 for all interrupts received (see EB), and ATRACT is set to 0 whenever a new code is accepted (see B10).

The Keyboard Handler obtains all key data from CH; whenever a code is extracted from that 1-byte FIFO, the Handler stores a value of $FF to the FIFO to indicate that the code has been read. See Section 5 for further discussion of the Keyboard Handler’s processing of the key codes.

E1 CH1* [02F2,1] -- Prior keyboard character code.

CH1 contains the key code value of the key most recently read and accepted.

E2 KEYDEL* [02F1,1] -- Debounce delay timer.

KEYDEL is set to a value of 3 whenever a key code is accepted, and is decremented every 60th of a second by the stage 2 VBLANK process (until it reaches zero).

E3 CH [02FC,1] -- Keyboard character code FIFO.

CH is a 1-byte FIFO that contains either the value of the most recently read and accepted key code or the value $FF (which indicates that the FIFO is empty). The FIFO is normally read by the Keyboard Handler, but can be read by a user program.

Key data can also be stored into CH by the Autorepeat logic as explained in the discussion relating to EB.
Special Functions

Start/Stop

Display Handler and Screen Editor output to the text or graphics mode screen can be stopped and started (without losing any of the output data) through the use of the [CTRL] 1 key combination. Each key depression toggles a flag that is monitored by the above mentioned Handlers. When the flag is nonzero, the handlers wait for it to go to zero before continuing any output.

E4 SSFLAG [02FF,1] -- Start/stop flag

The flag is normally zero, indicating that screen output is not to be stopped. The flag is complemented by every occurrence of the [CTRL] 1 key combination by the keyboard IRQ service routine.

The flag is set to zero upon power-up, [SYSTEM.RESET] or [BREAK] key processing.

[BREAK] Key

E5 BRKKEY [0011,1] -- [BREAK] key flag

BRKKEY is used to indicate that the [BREAK] key has been pressed. The value is normally nonzero and is set to zero whenever the [BREAK] key is pressed. The code that detects and processes the [BREAK] condition (flag = 0) should set the flag nonzero again.

BRKKEY is monitored by the following OS routines: Keyboard Handler, Display Handler, Screen Editor, Cassette Handler, xx?

The detection of a [BREAK] condition during an I/O operation will cause the operation to be aborted and a status of $80 to be returned to the user.

The flag is set to nonzero upon Power-up, [SYSTEM.RESET] or upon aborting a pending I/O operation.

[SHIFT]/[CONTROL] Lock

The keyboard control has three different modes for code generation that apply to the alphabetic keys A through Z: 1) normal, 2) caps lock, and 3) control lock.
In normal mode, all unmodified alphabetic character keys generate the lowercase letter ATASCII code ($61-7A).

In caps lock mode, all unmodified alphabetic character keys generate the uppercase letter ATASCII code ($41-5A).

In control lock mode, all unmodified alphabetic character keys generate the control letter ATASCII code ($01-1A).

In all three modes, any alphabetic character key that is modified (by being pressed in conjunction with the [SHIFT] or [CTRL] key) will generate the desired modified code.

E6 SHFLOK [02BE,1] — Shift/control lock control flag

SHFLOK normally has one of three values:

- $00 = normal mode (no locks in effect).
- $40 = caps lock.
- $80 = control lock.

SHFLOK is set to $40 upon Power-up and [SYSTEM.RESET] and is modified thereafter by the OS only when the [CAPS.LOWER] key is pressed (either by itself or in conjunction with the [SHIFT] or [CTRL] key).

E7 HOLDCH* [007C,1] — Character holding variable

HOLDCH is used to retain the current character value prior to the [SHIFT]/[CONTROL] logic process.

Autorepeat

The Autorepeat feature responds to the continuous depression of a keyboard key by replicating the key code 10 times per second, after an initial 1/2 second delay. The timer variable SRTIMR is used to control both the initial delay and the repeat rate.

Whenever SRTIMR is equal to zero and a key is being held down, the value of the key code is stored in the key code FIFO (CH). This logic is part of the stage 2 VBLANK process.

E8 SRTIMR* [022B,1] — Autorepeat timer

SRTIMR is controlled by two independent processes: 1) the keyboard IRQ service routine, which establishes the initial delay value and 2) the stage 2 VBLANK routine that establishes the repeat rate, decrements the timer and implements the auto repeat logic.
Inverse Video Control

The Keyboard Handler allows the direct generation of more than half of the 256 ATASCII codes; but codes $80-9A and codes $A0-FC can be generated only with the "inverse video mode" active. The ATARI key acts as an on/off toggle for this mode, and all characters (except for screen editing control characters) will be subject to inversion when the mode is active.

E9 INVFLG [0286,1] -- Inverse video flag

INVFLG is normally zero, indicating that normal video ATASCII codes (bit 7 = 0) are to be generated from keystrokes; whenever INVFLG is nonzero, inverse video ATASCII codes (bit 7 = 1) will be generated. The special control codes are exempt from this bit manipulation.

INVFLG is set to zero by power-up and system reset.

The Keyboard Handler inverts bit 7 of INVFLG whenever the ATARI key is pressed; the lower order bits are not altered and are assumed to be zero.

The Keyboard Handler's "exclusive or's" (XOR's) the ATASCII key data with the value in INVFLG at all times; the normal values of $00 and $80 thus lead to control of the inverse video bit (bit 7).

Console Keys: [SELECT], [START], and [OPTION]

The console keys are sensed directly from the hardware register CONSOL [DOIF]; see the ATARI Home Computer Hardware Manual for details.

F. PRINTER

See Section 5 for a general description of the Printer Handler.

Printer-Buffer

F1 PRNBUF* [03CO,40] -- Printer-record buffer

PRNBUF is the buffer used by the Printer Handler for packing printer data to be sent to the device controller. The buffer is 40 bytes long.
and contains nothing but printer data.

**F2 PBUFSZ** [001E,1] -- Printer-record size

PBUFSZ contains the size of the Printer-record for the current mode selected; the modes and respective sizes (in decimal bytes) are shown below:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>40</td>
</tr>
<tr>
<td>Double width</td>
<td>20 (not currently supported by the device)</td>
</tr>
<tr>
<td>Sideways</td>
<td>29</td>
</tr>
</tbody>
</table>

Status request 4

**F3 PBPNT** [001D,1] -- Printer-buffer index

PBPNT contains the current index to the Printer-buffer. PBPNT ranges in value from zero to the value of PBUFSZ.

Internal Working Variables

**F4 PTEMP** [001F,1] -- Printer Handler temporary data save

PTEMP is used by the Printer Handler to temporarily save the value of a character to be output to the printer.

**F5 PTIMOT** [001C,1] -- Printer timeout value

PTIMOT contains the timeout value for SIO calling sequence variable DTIMLO (see Section 9); PTIMOT is set to 30 (which represents a 32 second timeout) at initialization time, and is updated after each printer status request operation to contain the value returned in the third byte of the status frame (see Section 5).

G. CENTRAL I/O ROUTINE (CIO)

See Section 5 for a description of the Central I/O Utility.

User Call Parameters
CIO call parameters are passed primarily through an I/O Control Block (IOCB); although additional device status information can be returned in DVSTAT, and Handler information is obtained from the device table (HATABS).

I/O Control Block

IOCB is the name applied collectively to the 16 bytes associated with each of the 8 provided control structures; see Section 5.

G1 IOCB [0340,16] -- I/O Control Block

The label IOCB is the location of the first byte of the first IOCB in the data base. For VID's G2 through G10, the addresses given are for IOCB #0 only, the addresses for all of the IOCB's are shown below:

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0340-034F</td>
<td>IOCB #0</td>
</tr>
<tr>
<td>0350-035F</td>
<td>IOCB #1</td>
</tr>
<tr>
<td>0360-036F</td>
<td>IOCB #2</td>
</tr>
<tr>
<td>0370-037F</td>
<td>IOCB #3</td>
</tr>
<tr>
<td>0380-038F</td>
<td>IOCB #4</td>
</tr>
<tr>
<td>0390-039F</td>
<td>IOCB #5</td>
</tr>
<tr>
<td>03A0-03AF</td>
<td>IOCB #6</td>
</tr>
<tr>
<td>03B0-03BF</td>
<td>IOCB #7</td>
</tr>
</tbody>
</table>

G2 ICHID [0340,1] -- Handler ID

See Section 5. Initialized to $FF at power-up and system reset.

G3 ICDNO [0341,1] -- Device number

See Section 5.

G4 ICCOM [0342,1] -- Command byte

See Section 5.

G5 ICSTA [0343,1] -- Status

See Section 5.

G6 ICBAL,ICBAH [0344,2] -- Buffer address

See Section 5.
G7 ICPTL,ICPTH [0346,2] -- PUT BYTE vector

See Section 5. Initialized to point to CIO's "IOCB not OPEN" routine at power-up and system reset.

G8 ICBLL,ICBLH [0348,2] -- Buffer length / byte count

See Section 5.

G9 ICAX1,ICAX2 [034A,2] -- Auxiliary information

See Section 5.

G10 ICSPR [034C,4] -- Spare bytes for Handler use

There is no fixed assignment of these four bytes; the Handler associated with an IOCB can or may not use these bytes.

Device Status

G11 DVSTAT [02EA,4] -- Device status

See Section 5 for a discussion of the GET STATUS command.

Device Table

G12 HATABS [031A,38] -- Device table

See Section 9 for a description of the device table.

CID/Handler Interface Parameters

Communication between CIO and a Handler is accomplished using the 6502 machine registers, and a data structure called the Zero-page IOCB (ZIOCB). The ZIOCB is essentially a copy of the particular IOCB being used for the current operation.
Zero-Page IOCB

G13 ZIOCB (IOCBAS) [0020, 16] -- Zero-page IOCB

The Zero-page IOCB is an exact copy (except as noted in the discussions that follow) of the IOCB specified by the 6502 X register upon entry to CIO; CIO copies the outer level IOCB to the Zero-page IOCB, performs the indicated function, moves the (possibly altered) Zero-page IOCB back to the outer level IOCB, and then returns to the caller.

Although both the outer level IOCB and the Zero-page IOCB are defined to be 16 bytes in size, only the first 12 bytes are moved by CIO.

G14 ICHIDZ [0020, 1] -- Handler index number
See Section 5. Set to $FF on CLOSE.

G15 ICNOZ [0021, 1] -- Device drive number
See Section 5.

G16 ICCOMZ [0022, 1] -- Command byte
See Section 5.

G17 ICSTAZ [0023, 1] -- Status byte
See Section 5.

G18 ICBALZ,ICBALH [0024, 2] -- Buffer address
See Section 5. This pointer variable is modified by CIO in the course of processing some commands; however, the original value is restored before returning to the caller.

G19 ICPTLZ, ICPTHZ
See Section 5. Set to point to CIO's "IOCB not OPEN" routine on CLOSE.

G20 ICBLZ, ICBLHZ [0028, 2] -- Buffer length / byte count
See Section 5. This double-byte variable, which starts out representing the buffer length, is modified by CIO in the course of processing some commands; however, the original value is restored before returning to the caller.
of processing some commands; then, before returning to the caller, the transaction byte count is stored therein.

G21 ICAX1Z, ICAX2Z [002A, 2] -- Auxiliary information
See Section 5.

G22 ICSPRZ (ICIDNO, CIOCHR) [002C, 4] -- CIO working variables
ICSPRZ and ICSPRZ+1 are used by CIO in obtaining the appropriate Handler entry point from the handler's vector table (see Section 9).

ICSPRZ+2 is also labeled ICIDNO and retains the value of the 6502 X register from CIO entry. The X register is loaded from ICIDNO as CIO returns to the caller.

ICSPRZ+3 is also labeled CIOCHR and retains the value of the 6502 A register from CIO entry, except for data reading type commands, in which case the most recent data byte read is stored in CIOCHR. The 6502 A register is loaded from CIOCHR as CIO returns to the caller.

Internal Working Variables

G23 ICCOMT* [0017, 1] -- Command table index
ICCOMT is used as an index to CIO's internal command table, which maps command byte values to Handler entry offsets (see Section 9 for more information). ICCOMT contains the value from ICCOMZ except when ICCOMZ is greater than $OE, in which case ICCOMT is set to $OE.

G24 ICIDNO* [002E, 1] -- CIO call X register save/restore
See G22.

G25 CIOCHR* [002F, 1] -- CIO call A register save/restore
See G22.

H. SERIAL I/O ROUTINE (SIO)

See Section 9 for discussions relating to SIO.
User Call Parameters

SIO call parameters are passed primarily through a Device Control Block; although an additional "noisy bus" option exists that is selectable through a separate variable.

Device Control Block

H1 DCB [0300,12] -- Device Control Block

DCB is the name applied collectively to the 12 bytes at locations 0300-030B. These bytes provide the parameter passing mechanism for SIO and are described individually below.

H2 DDEVIC [0300,1] -- Device bus ID
See Section 9.

H3 DUNIT [0301,1] -- Device unit number
See Section 9.

H4 DCOMND [0302,1] -- Device command
See Section 9.

H5 DSTATS [0303,1] -- Device status
See Section 9.

H6 DBUFL0, DBUFI [0304,2] -- Handler buffer address
See Section 9.

H7 DTIMLO [0306,1] -- Device timeout
See Section 9.

H8 DBYTL0, DBYTHI [0308,2] -- Buffer length / byte count
See Section 9.

OPERATING SYSTEM CO16555 -- Appendix L
Bus Sound Control

H10 SOUND [0041,1] -- Quiet/noisy I/O flag

SOUND is a flag used to indicate to SIO whether noise is to be generated on the television audio circuit when Serial I/O bus activity is in progress. SOUND equal to zero indicates that sound is to be inhibited, and nonzero indicates that sound is to be enabled. SIO sets SOUND to 3 at power-up and system reset.

Serial Bus Control

Retry Logic

SIO will attempt one complete command retry if the first attempt is not error free, where a complete command try consists of up to 14 attempts to send (and acknowledge) a command frame, followed by a single attempt to receive COMPLETE and possibly a data frame.

H11 CRETRY [0036,1] -- Command frame retry counter

CRETRY controls the inner loop of the retry logic, that associated with sending and receiving an acknowledgement of the command frame. CRETRY is set to 13 by SIO at the beginning of every command initiation, thus allowing for an initial attempt and up to 13 additional retries.

H12 DRETRY [0037,1] -- Device retry counter

DRETRY controls the outer loop of the retry logic, that associated with initiating a command retry after a failure subsequent to the command frame acknowledgement. DRETRY is set to 1 by SIO at entry, thus allowing for an initial attempt and 1 additional retry.
Checksum

The Serial I/O bus protocol specifies that all command and data frames must contain a checksum validation byte; this byte is the arithmetic sum (with end-around carry) of all of the other bytes in the frame.

H13 CHKSUM* [0031,1] -- Checksum value

CHKSUM contains the frame checksum as computed by SIO for all frame transfers.

H14 CHKSNT* [003B,1] -- Checksum sent flag

CHKSNT indicates to the serial bus transmit interrupt service routine whether the frame checksum byte has been sent yet. CHKSNT equal to zero indicates that the checksum byte has not yet been sent; after the checksum is sent, CHKSNT is then set nonzero.

H15 NOCKSM* [003C,1] -- No checksum follows data flag

NOCKSM is a flag used to communicate between the SIO top level code and the Serial bus receive interrupt service routine that the next input will not be followed by a checksum byte. A value of zero specifies that a checksum byte will follow, nonzero specifies that a checksum byte will not follow.

Data Buffering

General Buffer Control

H16 BUFRLO* [0032,1] and BUFRHI* [0033,1] -- Next byte address

BUFRLO and BUFRHI comprise a pointer to the next buffer location to be read from or written to. For a data frame transfer, the pointer is initially set to the value contained in the SIO call parameters DBUFLO and DBUFHI, and is then incremented by the interrupt service routines as a part of normal bus data transfer. For a command frame transfer, the pointer is set to point to the SIO-maintained command frame output buffer.

H17 BFENLO* [0034,1] and BFENHI* [0035,1] -- Buffer end address

BFENLO/BFENHI form a pointer to the byte following the last frame data byte (not including the checksum) to be sent or received.
BFENLO/BFENHI is the arithmetic sum of BUFRLO/BUFRHI plus the frame size plus -1.

Command Frame Output Buffer
See Section 9 for the command frame format and description.

H18 CDEVIC* [023A,1] -- Command frame device ID
CDEVIC is set to the value obtained by adding SIO call parameter DDEVIC to DUNIT and subtracting 1.

H19 CCOMND* [023B,1] -- Command frame command.
CCOMND is set to the value obtained from SIO call parameter DCOMND.

H20 CAUX1* [023C,1] and CAUX2* [023D,1] -- Auxiliary information
CAUX1 and CAUX2 are set to the values obtained from SIO call parameters DAUX1 and DAUX2, respectively.

Receive/Transmit Data Buffering

H21 BUFRFL* [0038,1] -- Buffer full flag
BUFRFL is a flag used by the serial bus receive interrupt service routine to indicate when the main portion of a bus frame has been received -- all but the checksum byte. BUFRFL equal to zero indicates that the main portion has not been completely received, a nonzero value indicates that the main portion has been received.

H22 RECVDN* [0039,1] -- Receive frame done flag
RECVDN is a flag used by SIO to communicate between the Serial bus receive interrupt service routine and the main SIO code. The flag is initially set to zero by SIO, and later set nonzero by the interrupt service routine after the last byte of a bus frame has been received.

H23 TEMP* [023E,1] -- SIO 1-byte I/O data

OPERATING SYSTEM CO16555 -- Appendix L
TEMP is used to receive 1-byte responses from serial bus controllers, such as ACK, NAK, COMPLETE or ERROR.

H24 XMTDON* [003A,1] -- Transmit frame done flag

XMTDON is a flag used by SIO to communicate between the Serial bus transmit interrupt service routine and the main SIO code. The flag is initially set to zero by SIO, and later set nonzero by the interrupt service routine after the last byte of a bus frame has been transmitted.

SIO Timeout

SIO uses System Timer 1 to provide the timeout capability for various operations initiated internally. See Section 6 for a discussion of the capabilities of the System Timers. TIMFLG is the flag used to communicate between SIO and the timer initiated code pointed to by CDTMA1.

H25 TIMFLG* [0317,1] -- SIO operation timeout flag

TIMFLG is used to indicate a timeout situation for a bus operation. The flag is initially set to 1, and if it attains a value of zero (after the timeout period) before the current operation is complete, the operation will be aborted. See also D2.

H26 CDTMV1* [0218,2] -- System Timer 1 value

This 2-byte count takes on various values depending upon the operation being timed. See also P4.

H27 CDTMA1* [0226,2] -- System Timer 1 address

This vector always points to the JTIMER routine, whose only function is to set TIMFLG to zero. This vector is initialized by SIO before every use, so that System Timer 1 can be used by any process that does not use SIO within a timing function. See also P5.
Internal Working Variables

H28 STACKP* [0318,1] -- Stack pointer save/restore

STACKP contains the value of the 6502 SP register at entry to SIO; this is retained to facilitate a direct error exit from an SIO subroutine.

H29 TSTAT* [0319,1] -- Temporary status

TSTAT is used to return the operation status from the WAIT routine and will contain one of the SIO status byte values as shown in Appendix B.

H30 ERRFLG* [023F,1] -- I/O error flag

ERRFLG is used for communication between the WAIT routine and the outer level SIO code. ERRFLG is normally zero, but is set to $FF when a device responds with an invalid response byte.

H31 STATUS* [0030,1] -- SIO operation status

STATUS is a zero-page variable that is used within SIO to contain the operation status that will be stored to the calling sequence parameter variable DSTATS when SIO returns to the caller.

H32 SSKCTL* [0232,1] -- SKCTL copy

SSKCTL is utilized by SIO to keep track of the content of the SKCTL [D20F] register, which is a write-only register.

J. ATARI CONTROLLERS

The ATARI controllers are read as part of the Stage 2 VBLANK process. The encoded data is partially decoded and processed as shown in the subsections that follow.

Joysticks

Up to four joystick controllers can be attached to the computer console, each with a 9-position joystick plus a trigger button.
J1  STICKO - STICK3 [0278,4] -- Joystick position sense

The 4 joystick position sense variables contain a bit-encoded position sense as shown below:

\[
\begin{array}{cccccccc}
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\hline
 & & & & & & & +-----+-----+-----+-----+-----+
\end{array}
\]

where:
\begin{align*}
R &= 0 \text{ indicates joystick RIGHT sensor true.} \\
L &= 0 \text{ indicates joystick LEFT sensor true.} \\
D &= 0 \text{ indicates joystick DOWN sensor true.} \\
U &= 0 \text{ indicates joystick UP sensor true.}
\end{align*}

Nine unique combinations are possible, indicating the possible joystick positions shown below:

- CENTER: $0F$
- UP: $0E$
- UP/RIGHT: $06$
- RIGHT: $07$
- DOWN/RIGHT: $05$
- DOWN: $0D$
- DOWN/LEFT: $09$
- LEFT: $0B$
- UP/LEFT: $0A$

J2  STRIGO - STRIG3 [0284,4] -- Joystick trigger sense

The four joystick trigger sense variables each contain a single bit indicating the position of the joystick trigger as shown below:

\[
\begin{array}{cccccccc}
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
\hline
 & & & & & & & +-----+-----+-----+-----+-----+
\end{array}
\]

where:
\[T = 0 \text{ indicates trigger pressed.}\]

Paddles

Up to eight paddle controllers can be connected to the computer, each with a potentiometer and a trigger sense.

J3  PADDL0 - PADDL7 [0270,8] -- Paddle position sense

There is a single-byte variable associated with each paddle position sense; the values range from 228 for full
counterclockwise rotation to 1 for full clockwise rotation.

The paddle values are often converted by the user, as shown below, to give a result of 0 for full counterclockwise rotation and 227 for full clockwise rotation:

\[
\text{VALUE} := 228 - \text{PADDLX};
\]

J4 PTRIGO - PTRIG7 [027C,8] -- Paddle trigger sense

The 8-paddle trigger sense variables each contain a single bit indicating the position of the paddle trigger as shown below:

```
7 6 5 4 3 2 1 0
+-----------------------+
| 0 0 0 0 0 0 0 0 | T |
|-----------------------|
```

where: \( T = 0 \) indicates trigger pressed.

Light Pen

The OS reads the position of a single light pen and stores the horizontal and vertical position codes in two variables; these codes are not the same as the actual screen coordinates. The pen position codes for different portions of the screen are shown below:

- **Left edge** -- 67.
  - Codes increase in increments of one to a value of 227, then go to 0 and continue to increase monotonically (one count per color clock).
- **Right edge** -- 7.
- **Upper edge** -- 16.
  - Codes increase in increments of one (one count per two raster lines).
- **Lower edge** -- 111.

The light pen hardware will read and latch the pen position 60 times per second, independent of the pen button position, which is separately sensed.

In order for the light pen to operate it must be positioned over a portion of the screen which has sufficient luminance to activate the photosensor in the pen; a blank (dark) screen will generally not provide enough luminance to utilize the light pen.

J5 LPENH [0234,1] -- Light pen horizontal position code

LPENH contains the horizontal position code for the light pen; the algorithm below (written in Pascal) shows the conversion from position code to screen coordinate (screen mode 7):

\[
\begin{align*}
\text{IF LPENH < 33} & \quad \{ \text{check for rollover point} \} \\
\text{THEN} & \quad \{ \text{adjust values to right of rollover} \}
\end{align*}
\]
XPOS := LPENH + 227
ELSE { no adjustment to left of rollover point }
   XPOS := LPENH;
XPOS := XPOS - 67; { adjust for left edge offset }
IF XPOS < 0 THEN XPOS := 0;
IF XPOS > 159 THEN XPOS := 159;

J6 LPENV [0235, 1] -- Light pen vertical position code

LPENV contains the vertical position code for the light pen; the
algorithm below (written in Pascal) shows the conversion from position
code to screen coordinate (screen mode 7):

YPOS := LPENV - 16; { adjust for upper edge offset }
IF YPOS < 0 THEN YPOS := 0;
IF YPOS > 95 THEN YPOS := 95;

J7 STICK0 - STICK3 [0278, 4] -- Light pen button sense

The light pen button sense is encoded in one of STICK0 - STICK3
(depending upon the actual controller port used) as shown
below:

```
7
+-+++++++
1
+001001T!
+-+++++++
```

where: T = 0 indicates the light pen button is pressed.

Driving Controllers

The driving controller has no position stops and thus allows unlimited
rotation in either direction; the output of the controller is a 2-bit
Gray code which can be used to determine the direction of rotation.
The controller is sensed using the same internal hardware as the
joystick, thus the same data base variables are used for both.
J8 STICK0 - STICK3 [027B, 4] -- Driving controller sense

The 4 driving controller sense variables contain an encoded rotation (position) sense value, as shown below:

```
7 6 5 4 3 2 1 0
+++++++++++++++-------
!0 0 0 0 1 i!val!i
+++++++++++++++-------
```

where a clockwise rotation of the controller produces the following continuous sequence of four values (shown in hexadecimal):

OF, OD, OC, OE, OF, OD, ........

and a counterclockwise rotation of the controller produces the following continuous sequence of four values:

OF, OE, OC, OD, OF, OE, ........

J9 STRIGO - STRIG3 [0284, 4] -- Driving trigger sense

The four driving trigger sense variables each contain a single bit indicating the position of the driving trigger as shown below:

```
7 6 5 4 3 2 1 0
+++++++++++++++-------
!0 0 0 0 0 0 !T!i
+++++++++++++++-------
```

where: T = 0 indicates trigger pressed.

K. DISK FILE MANAGER

See Section 5 for information relating to the Disk File Manager.

K1 FMSZPG* [0043, 7] -- FMS reserved space

FMSZPG is the reserved space in the database for the variables shown below; the names associated with K2 through K5 are not in the system equate file.

K2 ZBUFP* [0043, 2] -- Buffer pointer

K3 ZDRVA* [0045, 2] -- Drive pointer

K4 ZSBA* [0047, 2] -- Sector buffer pointer

OPERATING SYSTEM CO16555 -- Appendix L
K5 ERRNO* [0049,1] -- Error number

L. DISK UTILITY POINTER

L1 DSKUTL* [001A,2] -- Page-zero pointer variable

M. FLOATING POINT PACKAGE

See Section 8 for a description of the Floating Point Package.

M1 FRO [00D4,6] -- FP register 0

M2 FRE* [00DA,6] -- FP register (internal)

M3 FR1 [00E0,6] -- FP register 1

M4 FR2* [00E6,6] -- FP register 2 (internal)

M5 FRX* [00EC,1] -- Spare (unused)

M6 EXP* [00ED,1] -- Exponent value (internal)

M7 NSIGN* [00EE,1] -- Sign of mantissa (internal)

M8 ESIGN* [00EF,1] -- Sign of exponent (internal)

M9 FCHRFLG* [00F0,1] -- First character flag (internal)

M10 DIGRT* [00F1,1] -- Digits to right of decimal point

M11 CIX [00F2,1] -- Character index

M12 INBUFF [00F3,2] -- Input text buffer pointer

OPERATING SYSTEM CO16555 -- Appendix L
M13 ZTEMP1* [00F5,2] -- Temporary storage
M14 ZTEMP4* [00F7,2] -- Temporary storage
M15 ZTEMP3* [00F9,2] -- Temporary storage
M16 FLPTR [00FC,2] -- Pointer to FP number
M17 FPTR2* [00FE,2] -- FP package use
M18 LBPR1* [057E,1] -- LBUFF preamble
M19 LBPR2* [057F,1] -- LBUFF preamble
M20 LBUFF [0580,96] -- Text buffer
M21 PLYARG* [05E0,6] -- FP register (internal)
M22 FPSCR/FSCR* [05E6,6] -- FP register (internal)
M23 FPSCR1/SCR1* [05EC,6] -- FP register (internal)
M24 DEGFLG/RADFLG [00FB,1] -- Degrees/radians flag

DEGFLG = 0 indicates radians, 6 indicates degrees.

N. Power-Up and SYSTEM RESET

See Section 7 for details of the power-up and system reset operations.

RAM Sizing

During power-up and system reset the first non-RAM address above 1000 hex is located and its address retained using a nondestructive test. The first byte of every 4K memory "block" is tested to see if it is alterable; if so, the original value is restored and the next block is tested, and if not, that address is considered to be the end of RAM.
N1 RAMLO*/TRAMSZ* [0004,3] -- RAM data/test pointer (temporary)

RAMLO+1 contains the LSB of the address to be tested (always = 0) and TRAMSZ (same as RAMLO+2) contains the MSB of the address to be tested. RAMLO+0 contains the complemented value of the data originally contained in the memory location being tested.

Later in the initialization process these variables are used for totally unrelated functions; but first the value in TRAMSZ is moved to the variables RAMSIZ and MEMTOP+1.

N2 TSTDAT* [0007,1] -- Test data byte save

TSTDAT contains the original value of the memory location being tested.

Diskette/Cassette-Boot

As a part of the Power-up sequence, software can be booted from an attached disk drive or cassette player as explained in Section 10.

N3 DOSINI [000C,2] -- Diskette-boot initialization vector.

DOSINI contains the disk booted software initialization address from the beginning of the boot file (see Section 10) whenever a diskette-boot is successfully completed.

N4 CKEY* [004A,1] -- Cassette-boot request flag

CKEY is an internal flag used to indicate that the console [START] key was pressed during Power-up, thus indicating that a cassette-boot is desired. CKEY equals zero when no cassette-boot is requested, and is nonzero when a cassette-boot is requested. The flag is cleared to zero after a cassette-boot.

N5 CASSBT* [004B,1] -- Cassette-booting flag

CASSBT is used during the cassette-boot process to indicate to shared code that the cassette is being booted and not the diskette. CASSBT equal to zero indicates a diskette-boot, and nonzero indicates a cassette-boot.

N6 CASINI [0002,2] -- Cassette-boot initialization vector

CASINI contains the cassette-booted software initialization address from the beginning of the boot file (see Section 10) whenever a
cassette-boot is successfully completed.

N7 BOOT?* [0009,1] -- Successful diskette/cassette-boot flag.

BOOT? indicates to the initialization processor which, if any, of the boot operations went to successful completion. The flag values are set by the OS and the format for the variable is shown below:

```
7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+
!           !CID!
+-+-+-+-+-+-+-+
```

where: C = 1 indicates that the cassette-boot was completed.
D = 1 indicates that the diskette-boot was completed.

N8 DFLAGS* [0240,1] -- Diskette flags

DFLAGS contains the value of the first byte of the boot file, after a diskette-boot. See Section 10.

N9 DBSECT* [0241,1] -- Diskette-boot sector count

DBSECT is initially set to the value of the second byte of the boot file, during a diskette-boot, and is then used to control the number of additional diskette sectors read, if any.

N10 BOOTAD* [0242,2] -- Diskette-boot memory address

BOOTAD is initially set to the value of the third and fourth bytes of the boot file, during a diskette-boot, and is not modified thereafter.

Environment Control

If, at the end of a power-up or system reset, control is not given to one of the cartridges (as explained in Sections 7 and 10), then program control passes to the address contained in the data base variable DOSVEC.

N11 COLDST* [0244,1] -- Coldstart complete flag

COLDST is used by the initialization routine to detect the case of a system reset occurring before the completion of the power-up process. COLDST is set to $FF at the beginning of the power-up
sequence and is set to 0 at the completion; if a system reset occurs while the value is nonzero, the power-up sequence will be reinitiated (rather than initiating a system reset sequence).

N12 DOSVEC [000A,2] -- Noncartridge control vector

At the beginning of power-up the OS sets DOSVEC to point to the "blackboard" routine; DOSVEC can then be altered as a consequence of a diskette-boot or cassette-boot (as explained in Section 10) to establish a new control program. Control will be passed through DOSVEC on all power-up and system reset conditions in which a cartridge does not take control first.

System Reset

N13 WARMST [0008,1] -- Warmstart flag

WARMST equals $FF during a system reset (warmstart) initialization and equals 0 during a power-up initialization (coldstart).

P. INTERRUPTS

See Section 6 for a discussion of interrupt processing.

P1 CRITIC [0042,1] -- Critical code section flag

CRITIC is used to signal to the VBLANK interrupt processor that a critical code section is executing without IRQ interrupts being inhibited; the VBLANK interrupt processor will stop interrupt processing after stage 1 and before stage 2, just as if the 6502 processor I bit were set, when CRITIC is set.

CRITIC equal to zero indicates that the currently executing code section is noncritical, while any nonzero value indicates that the currently executing code section is critical.

P2 POKMSK [0010,1] -- POKEY interrupt mask

POKMSK is a software maintained interrupt mask that is used in conjunction with the enabling and disabling of the various POKEY interrupts. This mask is required because the POKEY interrupt enable register IRGEN [D20E] is a write-only register, and at any point in time the system can have several users independently enabling and disabling POKEY interrupts. POKMSK is updated by the
users to always contain the current content of IRGEN.

System Timers

The System Timers are discussed in detail in Section 6.

Realtime Clock

The realtime clock (or frame counter, as it is sometimes called) is incremented as part of the stage 1 VBLANK process as explained in Section 6.

P3 RTCLOK [0012,3] -- Realtime frame counter

RTCLOK+0 is the most significant byte, RTCLOK+1 the next most significant byte, and RTCLOK+2 the least significant byte. See the discussions at D3 and preceding B10 for OS use of RTCLOK.

System Timer 1

System Timer 1 is maintained as part of the stage 1 VBLANK process, and thus has the highest priority of any of the user timers.

P4 CDTMV1 [0218,2] -- System Timer 1 value

CDTMV1 contains zero when the timer is inactive, otherwise it contains the number of VBLANKs remaining until timeout. Also see H26.

P5 CDTMA1 [0226,2] -- System Timer 1 jump address

CDTMA1 contains the address to which to JSR should the timer timeout. See also H27 and Section 6.
System Timer 2

System Timer 2 is maintained as part of the stage 2 VBLANK process, and has the second highest priority of the user timers. The OS does not have any direct use for System Timer 2.

P6 CDTMV2 [021A,2] -- System Timer 2 value

CDTMV2 contains zero when the timer is inactive, otherwise it contains the number of VBLANKs remaining until timeout.

P7 CDTMA2 [0228,2] -- System Timer 2 jump address

CDTMA2 contains the address to which to JSR should the timer timeout. See Section 6.

System Timers 3, 4 and 5

System Timers 3, 4 and 5 are maintained as part of the stage 2 VBLANK process, and have the lowest priority of the user timers. The OS does not have any direct use for these timers.

P8 CDTMV3 [021C,2], CDTMV4 [021E,2] and CDTMV5 [0220,2]

These variables contain zero when the corresponding timers are inactive, otherwise they contain the number of VBLANKs remaining until timeout.

P9 CDTMF3 [022A,1], CDTMF4 [022C,1] and CDTMF5 [022E,2]

Each of these 1-byte variables will be set to zero should its corresponding timer timeout. The OS never modifies these bytes except to set them to zero upon timeout (and initialization).

RAM Interrupt Vectors

There are RAM vectors for many of the interrupt conditions within the system. See Section 6 for a discussion of the placing of values to these vectors.
NMI Interrupt Vectors

P10 VDSLST [0200,2] -- Display-list interrupt vector
This vector is not used by the OS. See Section 6.

P11 VVBLKI [0222,2] -- Immediate VBLANK vector
This vector is initialized to point to the OS stage 1 VBLANK

P12 VVBLKD [0224,2] -- Deferred VBLANK vector
This vector is initialized to point to the OS VBLANK exit routine. See Section 6.

IRQ Interrupt Vectors

P13 VIMIRQ [0216,2] -- General IRQ vector
This vector is initialized to point to the OS IRQ interrupt processor. See Section 6.

P14 VPRCED [0202,2] -- Serial I/O bus proceed signal
The serial bus line that produces this interrupt is not used in the current system. See Section 6.

P15 VINTER [0204,2] -- Serial I/O bus interrupt signal
The serial bus line that produces this interrupt is not used in the current system. See Section 6.

P16 V[BREAK] [0206,2] -- BRK instruction vector
This vector is initialized to point to a PLA, RTI sequence as the OS proper does not utilize the BRK instruction. See Section 6.

P17 VKEYBD [0208,2] -- Keyboard interrupt vector
This vector is initialized to point to the Keyboard Handler's interrupt service routine. See Section 6 and the discussion preceding E1.
P18  VSERIN [020A,2] -- Serial I/O bus receive data ready

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P19  VSEROR [020C,2] -- Serial I/O bus transmit ready

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P20  VSEROC [020E,2] -- Serial I/O bus transmit complete

This vector is initialized to point to the SIO utility's interrupt service routine. See Section 6.

P21  VTIMRI [0210,2], VTIMR2 [0212,2] and VTIMR4 [0214,2] -- POKEY timer vectors

The POKEY timer interrupts are not used by the OS. See Section 6.

Hardware Register Updates

As part of the stage 2 VBLANK process, certain hardware registers are updated from OS data base variables as explained in Section 6.

P22  SDMCTL* [022F,1] -- DMA control

SDMCTL is set to a value of $02 at the beginning of a Display Handler OPEN command, and then later set to a value of $22. The value of SDMCTL is stored to DMACTL [D400] as part of the stage 2 VBLANK process.

P23  SDLSTL* [0230,1] and SDLSTH* [0231,1] -- Display list address

The Display Handler formats a new display list with every OPEN command and puts the display list address in SDLSTL and SDLSTH. The value of these bytes are stored to DLISTL [D402] and DLISTH [D403] as part of the stage 2 VBLANK process.

0360-036F  IOC8 #2
0370-037F  IOC8 #3
0380-038F  IOC8 #4
0390-039F  IOC8 #5
03A0-03AF  IOC8 #6
03B0-03BF  IOC8 #7
NOTE: There is a potential timing problem associated with the updating of the hardware registers from the data base variables. Since the stage 2 VBLANK process is performed with interrupts enabled, it is possible for an IRQ interrupt to occur before the updating of DLISTH and DLISTL. If the processing of that interrupt (plus other nested interrupts) exceeds the vertical-blank delay (1 msec), then the display list pointer register will not have been updated when display list processing commences for the new frame, and a screen glitch will result.

P24 GPRIOR* [026F,1] -- Priority control

The Display Handler alters bits 6 and 7 of GPRIOR as part of establishing the GTIA mode. The value of GPRIOR is stored to PRIOR [D01B] as part of the stage 2 VBLANK process.

P25 CHACT* [02F3,1] -- Character control

The Display Handler sets CHACT to $02 on every OPEN command. The value of CHACT is stored to CHACTL [D401] as part of the stage 2 VBLANK process.

P26 CHBAS [02F4,1] -- Character address base

The Display Handler sets CHBAS to $EO on every OPEN command. The value of CHBAS is stored to CHBASE [D409] as part of the stage 2 VBLANK process. This variable controls the character subset for screen modes 1 and 2; a value of $EO provides the capital letters and number set whereas a value of $E2 provides the lowercase letters and special graphics set. See B55 for more information.

P27 PCOLRx [02C0,4] and COLORx [02C4,9] -- Color registers

See B7 and B8.

Internal Working Variables

P28 INTEMP* [02D,1] -- Temporary storage

INTEMP is used by the SETVBL (SETVBV) routine.
R. USER AREAS

The areas shown below are available to the user in a non-nested environment. See Section 4 for further information.

R1 [0080, 128]

R2 [0480, 640]
### ALPHABETICAL LIST OF DATA BASE VARIABLES

<table>
<thead>
<tr>
<th>NAME</th>
<th>VID</th>
<th>ADDRESS</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDCOR</td>
<td>D4</td>
<td>030E, 1</td>
<td></td>
</tr>
<tr>
<td>ADDRESS</td>
<td>B39</td>
<td>0064, 2</td>
<td></td>
</tr>
<tr>
<td>APPMHI</td>
<td>A3</td>
<td>000E, 2</td>
<td></td>
</tr>
<tr>
<td>ATACHR</td>
<td>B54</td>
<td>02FB, 1</td>
<td></td>
</tr>
<tr>
<td>ATRACT</td>
<td>B10</td>
<td>004D, 1</td>
<td></td>
</tr>
<tr>
<td>BFENHI</td>
<td>H17</td>
<td>0035, 1</td>
<td></td>
</tr>
<tr>
<td>BFENLO</td>
<td>H17</td>
<td>0034, 1</td>
<td></td>
</tr>
<tr>
<td>BITMSK</td>
<td>B50</td>
<td>006E, 1</td>
<td></td>
</tr>
<tr>
<td>BLIM</td>
<td>D10</td>
<td>028A, 1</td>
<td></td>
</tr>
<tr>
<td>BOOT?</td>
<td>N7</td>
<td>0009, 1</td>
<td></td>
</tr>
<tr>
<td>BOOTAD</td>
<td>N10</td>
<td>0242, 2</td>
<td></td>
</tr>
<tr>
<td>BOTSCR</td>
<td>B16</td>
<td>028F, 1</td>
<td></td>
</tr>
<tr>
<td>BPTR</td>
<td>D11</td>
<td>003D, 1</td>
<td></td>
</tr>
<tr>
<td>BRKKEY</td>
<td>E5</td>
<td>0011, 1</td>
<td></td>
</tr>
<tr>
<td>BUFADR</td>
<td>C1</td>
<td>0015, 2</td>
<td></td>
</tr>
<tr>
<td>BUFCNT</td>
<td>B42</td>
<td>006B, 1</td>
<td></td>
</tr>
<tr>
<td>BUFRFL</td>
<td>H21</td>
<td>0038, 1</td>
<td></td>
</tr>
<tr>
<td>BUFRHI</td>
<td>H16</td>
<td>0033, 1</td>
<td></td>
</tr>
<tr>
<td>BUFRLO</td>
<td>H16</td>
<td>0032, 1</td>
<td></td>
</tr>
<tr>
<td>BUFSTR</td>
<td>B43</td>
<td>006C, 2</td>
<td></td>
</tr>
<tr>
<td>CASBUF</td>
<td>D9</td>
<td>03FD, 131</td>
<td></td>
</tr>
<tr>
<td>CASFLG</td>
<td>D8</td>
<td>030F, 1</td>
<td></td>
</tr>
<tr>
<td>CASINI</td>
<td>N6</td>
<td>0002, 2</td>
<td></td>
</tr>
<tr>
<td>CASSBT</td>
<td>N5</td>
<td>004B, 1</td>
<td></td>
</tr>
<tr>
<td>CAUX1</td>
<td>H20</td>
<td>023C, 1</td>
<td></td>
</tr>
<tr>
<td>CAUX2</td>
<td>H20</td>
<td>023D, 1</td>
<td></td>
</tr>
<tr>
<td>CBAUDH</td>
<td>D1</td>
<td>02EF, 1</td>
<td></td>
</tr>
<tr>
<td>CBAUDL</td>
<td>D1</td>
<td>02EE, 1</td>
<td></td>
</tr>
<tr>
<td>COMND</td>
<td>H19</td>
<td>023B, 1</td>
<td></td>
</tr>
<tr>
<td>CDEVIC</td>
<td>H18</td>
<td>023A, 1</td>
<td></td>
</tr>
<tr>
<td>CDTMA1</td>
<td>P5, H27</td>
<td>0226, 2</td>
<td></td>
</tr>
<tr>
<td>CDTMA2</td>
<td>P7</td>
<td>022B, 2</td>
<td></td>
</tr>
<tr>
<td>CDTMF3</td>
<td>P9</td>
<td>022A, 1</td>
<td></td>
</tr>
<tr>
<td>CDTMF4</td>
<td>P9</td>
<td>022C, 1</td>
<td></td>
</tr>
<tr>
<td>CDTMF5</td>
<td>P9</td>
<td>022E, 1</td>
<td></td>
</tr>
<tr>
<td>CDTMV1</td>
<td>P4, H26</td>
<td>0226, 2</td>
<td></td>
</tr>
<tr>
<td>CDTMV2</td>
<td>P6</td>
<td>021A, 2</td>
<td></td>
</tr>
<tr>
<td>CDTMV3</td>
<td>P8</td>
<td>021C, 2</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Value</td>
<td>Offset</td>
<td>Type</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>CDTMV4</td>
<td>P8</td>
<td>021E,2</td>
<td></td>
</tr>
<tr>
<td>CDTMV5</td>
<td>P8</td>
<td>0220,2</td>
<td></td>
</tr>
<tr>
<td>CH</td>
<td>E3</td>
<td>02FC,1</td>
<td></td>
</tr>
<tr>
<td>CHKSNT</td>
<td>H14</td>
<td>003B,1</td>
<td></td>
</tr>
<tr>
<td>CH1</td>
<td>E1</td>
<td>02F2,1</td>
<td></td>
</tr>
<tr>
<td>CHACT</td>
<td>P25</td>
<td>02F3,1</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>B55</td>
<td>02FA,1</td>
<td></td>
</tr>
<tr>
<td>CHBAS</td>
<td>P26</td>
<td>02F4,1</td>
<td></td>
</tr>
<tr>
<td>CHKSNT</td>
<td>H14</td>
<td>003B,1</td>
<td></td>
</tr>
<tr>
<td>CIOCHR</td>
<td>G25</td>
<td>002F,1</td>
<td></td>
</tr>
<tr>
<td>CIX</td>
<td>M11</td>
<td>00F2,1</td>
<td></td>
</tr>
<tr>
<td>CKEY</td>
<td>N4</td>
<td>004A,1</td>
<td></td>
</tr>
<tr>
<td>COLAC</td>
<td>B24</td>
<td>0072,2</td>
<td></td>
</tr>
<tr>
<td>COLCRS</td>
<td>B2</td>
<td>0055,2</td>
<td></td>
</tr>
<tr>
<td>COLDST</td>
<td>N11</td>
<td>0244,1</td>
<td></td>
</tr>
<tr>
<td>COLINC</td>
<td>B21</td>
<td>007A,1</td>
<td></td>
</tr>
<tr>
<td>COLOR0</td>
<td>B8,P27</td>
<td>02C4,1</td>
<td></td>
</tr>
<tr>
<td>COLOR1</td>
<td>B8,P27</td>
<td>02C5,1</td>
<td></td>
</tr>
<tr>
<td>COLOR2</td>
<td>B8,P27</td>
<td>02C6,1</td>
<td></td>
</tr>
<tr>
<td>COLOR3</td>
<td>B8,P27</td>
<td>02C7,1</td>
<td></td>
</tr>
<tr>
<td>COLOR4</td>
<td>B8,P27</td>
<td>02CB,1</td>
<td></td>
</tr>
<tr>
<td>COLRSH</td>
<td>B11</td>
<td>004F,1</td>
<td></td>
</tr>
<tr>
<td>COUNTR</td>
<td>B23</td>
<td>007E,2</td>
<td></td>
</tr>
<tr>
<td>CRETRY</td>
<td>H11</td>
<td>0036,1</td>
<td></td>
</tr>
<tr>
<td>CRITIC</td>
<td>P1</td>
<td>0042,1</td>
<td></td>
</tr>
<tr>
<td>CRSINH</td>
<td>B1</td>
<td>02F0,1</td>
<td></td>
</tr>
<tr>
<td>CSTAT</td>
<td>S2</td>
<td>028B,1</td>
<td></td>
</tr>
<tr>
<td>DAUX1</td>
<td>H9</td>
<td>030A,1</td>
<td></td>
</tr>
<tr>
<td>DAUX2</td>
<td>H9</td>
<td>030B,2</td>
<td></td>
</tr>
<tr>
<td>DBSECT</td>
<td>N9</td>
<td>0241,1</td>
<td></td>
</tr>
<tr>
<td>DBUFIH</td>
<td>H6</td>
<td>0304,1</td>
<td></td>
</tr>
<tr>
<td>DBUFLO</td>
<td>H6</td>
<td>0305,1</td>
<td></td>
</tr>
<tr>
<td>DBYTHI</td>
<td>H8</td>
<td>030B,1</td>
<td></td>
</tr>
<tr>
<td>DBYTLO</td>
<td>H8</td>
<td>0309,1</td>
<td></td>
</tr>
<tr>
<td>DCB</td>
<td>H1</td>
<td>0300,12</td>
<td></td>
</tr>
<tr>
<td>DCOMMD</td>
<td>H4</td>
<td>0302,1</td>
<td></td>
</tr>
<tr>
<td>DDEVIC</td>
<td>H2</td>
<td>0300,1</td>
<td></td>
</tr>
<tr>
<td>DEQFLQ</td>
<td>M24</td>
<td>00FB,1</td>
<td></td>
</tr>
<tr>
<td>DELTAC</td>
<td>B22</td>
<td>0077,2</td>
<td></td>
</tr>
<tr>
<td>DELTAR</td>
<td>B22</td>
<td>0076,1</td>
<td></td>
</tr>
<tr>
<td>DFLAGS</td>
<td>N8</td>
<td>0240,1</td>
<td></td>
</tr>
<tr>
<td>DIGRT</td>
<td>M10</td>
<td>00F1,1</td>
<td></td>
</tr>
<tr>
<td>DINDEX</td>
<td>B35</td>
<td>0057,1</td>
<td></td>
</tr>
<tr>
<td>DMAPK</td>
<td>B28</td>
<td>02A0,1</td>
<td></td>
</tr>
<tr>
<td>DOSINI</td>
<td>N3</td>
<td>000C,2</td>
<td></td>
</tr>
<tr>
<td>DOSVEC</td>
<td>N12</td>
<td>000A,2</td>
<td></td>
</tr>
<tr>
<td>DRETRY</td>
<td>H12</td>
<td>0037,1</td>
<td></td>
</tr>
<tr>
<td>DRKMSK</td>
<td>B12</td>
<td>004E,1</td>
<td></td>
</tr>
<tr>
<td>DSKTIM</td>
<td>C2</td>
<td>0246,1</td>
<td></td>
</tr>
<tr>
<td>DSKUTL</td>
<td>L1</td>
<td>001A,2</td>
<td></td>
</tr>
<tr>
<td>DSPFLG</td>
<td>B27</td>
<td>02FE,1</td>
<td></td>
</tr>
<tr>
<td>DSTAT</td>
<td>B34</td>
<td>004C,1</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Offset</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>DSTATS</td>
<td>H5</td>
<td>0303, 1</td>
<td></td>
</tr>
<tr>
<td>DTIMLO</td>
<td>H7</td>
<td>0306, 1</td>
<td></td>
</tr>
<tr>
<td>DUNIT</td>
<td>H3</td>
<td>0301, 1</td>
<td></td>
</tr>
<tr>
<td>DUNUSE</td>
<td>S3</td>
<td>0307, 1</td>
<td></td>
</tr>
<tr>
<td>DVSTAT</td>
<td>G11</td>
<td>02EA, 4</td>
<td></td>
</tr>
<tr>
<td>EEXP</td>
<td>M6</td>
<td>00ED, 1</td>
<td></td>
</tr>
<tr>
<td>ENDPT</td>
<td>B25</td>
<td>0074, 2</td>
<td></td>
</tr>
<tr>
<td>ERRFLG</td>
<td>H30</td>
<td>023F, 1</td>
<td></td>
</tr>
<tr>
<td>(ERRNO</td>
<td>K5</td>
<td>0049, 1</td>
<td></td>
</tr>
<tr>
<td>ESCFLG</td>
<td>B26</td>
<td>02A2, 1</td>
<td></td>
</tr>
<tr>
<td>EISIGN</td>
<td>M8</td>
<td>00EF, 1</td>
<td></td>
</tr>
<tr>
<td>FCHRFL</td>
<td>M9</td>
<td>00F0, 1</td>
<td></td>
</tr>
<tr>
<td>FEOD</td>
<td>D12</td>
<td>003F, 1</td>
<td></td>
</tr>
<tr>
<td>FILDAT</td>
<td>B17</td>
<td>02FD, 1</td>
<td></td>
</tr>
<tr>
<td>FILFLG</td>
<td>B18</td>
<td>02B7, 1</td>
<td></td>
</tr>
<tr>
<td>FLPTR</td>
<td>M16</td>
<td>00FC, 2</td>
<td></td>
</tr>
<tr>
<td>FMSZPG</td>
<td>K1</td>
<td>0043, 7</td>
<td></td>
</tr>
<tr>
<td>FPSCR</td>
<td>M22</td>
<td>05E6, 6</td>
<td></td>
</tr>
<tr>
<td>FPSCR1</td>
<td>M23</td>
<td>05EC, 6</td>
<td></td>
</tr>
<tr>
<td>FPTR2</td>
<td>M17</td>
<td>00FE, 2</td>
<td></td>
</tr>
<tr>
<td>FRO</td>
<td>M1</td>
<td>00D4, 6</td>
<td></td>
</tr>
<tr>
<td>FR1</td>
<td>M3</td>
<td>00E0, 6</td>
<td></td>
</tr>
<tr>
<td>FR2</td>
<td>M4</td>
<td>00E6, 6</td>
<td></td>
</tr>
<tr>
<td>FRE</td>
<td>M2</td>
<td>00DA, 6</td>
<td></td>
</tr>
<tr>
<td>FREQ</td>
<td>D15</td>
<td>0040, 1</td>
<td></td>
</tr>
<tr>
<td>FMRADR</td>
<td>B41</td>
<td>0068, 2</td>
<td></td>
</tr>
<tr>
<td>FRX</td>
<td>M5</td>
<td>00EC, 1</td>
<td></td>
</tr>
<tr>
<td>FSCCR</td>
<td>M22</td>
<td>05E6, 6</td>
<td></td>
</tr>
<tr>
<td>FSCCR1</td>
<td>M23</td>
<td>05EC, 6</td>
<td></td>
</tr>
<tr>
<td>FTYP</td>
<td>D13</td>
<td>003E, 1</td>
<td></td>
</tr>
<tr>
<td>GPRIOR</td>
<td>P24</td>
<td>026F, 1</td>
<td></td>
</tr>
<tr>
<td>HATABS</td>
<td>G12</td>
<td>031A, 38</td>
<td></td>
</tr>
<tr>
<td>HOLD1</td>
<td>B30</td>
<td>0051, 1</td>
<td></td>
</tr>
<tr>
<td>HOLD2</td>
<td>B31</td>
<td>029F, 1</td>
<td></td>
</tr>
<tr>
<td>HOLD3</td>
<td>B32</td>
<td>029D, 1</td>
<td></td>
</tr>
<tr>
<td>HOLD4</td>
<td>B20</td>
<td>02BC, 1</td>
<td></td>
</tr>
<tr>
<td>HOLCH</td>
<td>E7</td>
<td>007C, 1</td>
<td></td>
</tr>
<tr>
<td>ICAX1</td>
<td>G9</td>
<td>034A, 1</td>
<td></td>
</tr>
<tr>
<td>ICAX1Z</td>
<td>G21</td>
<td>002A, 1</td>
<td></td>
</tr>
<tr>
<td>ICAX2</td>
<td>G9</td>
<td>034B, 1</td>
<td></td>
</tr>
<tr>
<td>ICAX2Z</td>
<td>G21</td>
<td>002B, 1</td>
<td></td>
</tr>
<tr>
<td>ICBAH</td>
<td>G6</td>
<td>0345, 1</td>
<td></td>
</tr>
<tr>
<td>ICBAHZ</td>
<td>G18</td>
<td>0025, 1</td>
<td></td>
</tr>
<tr>
<td>ICBAL</td>
<td>G6</td>
<td>0344, 1</td>
<td></td>
</tr>
<tr>
<td>ICBALZ</td>
<td>G18</td>
<td>0024, 1</td>
<td></td>
</tr>
<tr>
<td>ICBLH</td>
<td>G8</td>
<td>0349, 1</td>
<td></td>
</tr>
<tr>
<td>ICBLHZ</td>
<td>G20</td>
<td>0029, 1</td>
<td></td>
</tr>
<tr>
<td>ICBLL</td>
<td>G8</td>
<td>0348, 1</td>
<td></td>
</tr>
<tr>
<td>ICBLLLZ</td>
<td>G20</td>
<td>002B, 1</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Address</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>PADDL5</td>
<td>J3</td>
<td>0275,1</td>
<td></td>
</tr>
<tr>
<td>PADDL6</td>
<td>J3</td>
<td>0276,1</td>
<td></td>
</tr>
<tr>
<td>PADDL7</td>
<td>J3</td>
<td>0277,1</td>
<td></td>
</tr>
<tr>
<td>PPBTNT</td>
<td>F3</td>
<td>001D,1</td>
<td></td>
</tr>
<tr>
<td>PBUSF2</td>
<td>F2</td>
<td>001E,1</td>
<td></td>
</tr>
<tr>
<td>PCOLR0</td>
<td>B7, P27</td>
<td>02C0,1</td>
<td></td>
</tr>
<tr>
<td>PCOLR1</td>
<td>B7, P27</td>
<td>02C1,1</td>
<td></td>
</tr>
<tr>
<td>PCOLR2</td>
<td>B7, P27</td>
<td>02C2,1</td>
<td></td>
</tr>
<tr>
<td>PCOLR3</td>
<td>B7, P27</td>
<td>02C3,1</td>
<td></td>
</tr>
<tr>
<td>PLYARG</td>
<td>M21</td>
<td>05E0,6</td>
<td></td>
</tr>
<tr>
<td>POKMSK</td>
<td>P2</td>
<td>0010,1</td>
<td></td>
</tr>
<tr>
<td>PRNBUF</td>
<td>F1</td>
<td>03C0,40</td>
<td></td>
</tr>
<tr>
<td>PTEMP</td>
<td>F4</td>
<td>001F,1</td>
<td></td>
</tr>
<tr>
<td>PTIMOT</td>
<td>F5</td>
<td>001C,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG0</td>
<td>J4</td>
<td>027C,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG1</td>
<td>J4</td>
<td>027D,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG2</td>
<td>J4</td>
<td>027E,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG3</td>
<td>J4</td>
<td>027F,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG4</td>
<td>J4</td>
<td>0280,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG5</td>
<td>J4</td>
<td>0281,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG6</td>
<td>J4</td>
<td>0282,1</td>
<td></td>
</tr>
<tr>
<td>PTRIG7</td>
<td>J4</td>
<td>0283,1</td>
<td></td>
</tr>
<tr>
<td>RADFLG</td>
<td>M24</td>
<td>00FB,1</td>
<td></td>
</tr>
<tr>
<td>RAMLO</td>
<td>N1</td>
<td>0004,3</td>
<td></td>
</tr>
<tr>
<td>RAMSZ</td>
<td>A5</td>
<td>02E4,1</td>
<td></td>
</tr>
<tr>
<td>RAMTOP</td>
<td>A4</td>
<td>006A,1</td>
<td></td>
</tr>
<tr>
<td>RECEVN</td>
<td>H22</td>
<td>0039,1</td>
<td></td>
</tr>
<tr>
<td>RMARGIN</td>
<td>B6</td>
<td>0053,1</td>
<td></td>
</tr>
<tr>
<td>ROWAC</td>
<td>B24</td>
<td>0070,2</td>
<td></td>
</tr>
<tr>
<td>ROWCRS</td>
<td>B2</td>
<td>0054,1</td>
<td></td>
</tr>
<tr>
<td>ROWINC</td>
<td>B21</td>
<td>0079,1</td>
<td></td>
</tr>
<tr>
<td>RTCLK</td>
<td>P3</td>
<td>0012,3</td>
<td></td>
</tr>
<tr>
<td>SAVADR</td>
<td>B41</td>
<td>006B,2</td>
<td></td>
</tr>
<tr>
<td>SAVIO</td>
<td>D7</td>
<td>0316,1</td>
<td></td>
</tr>
<tr>
<td>SAVMSC</td>
<td>B36</td>
<td>005B,2</td>
<td></td>
</tr>
<tr>
<td>SCRFLG</td>
<td>B9</td>
<td>02BB,1</td>
<td></td>
</tr>
<tr>
<td>SDLSTH</td>
<td>P23</td>
<td>0231,1</td>
<td></td>
</tr>
<tr>
<td>SDLSTL</td>
<td>P23</td>
<td>0230,1</td>
<td></td>
</tr>
<tr>
<td>SDMCTL</td>
<td>P22</td>
<td>022F,1</td>
<td></td>
</tr>
<tr>
<td>SHFAMT</td>
<td>B29</td>
<td>006F,1</td>
<td></td>
</tr>
<tr>
<td>SHFLOK</td>
<td>E6</td>
<td>02BE,1</td>
<td></td>
</tr>
<tr>
<td>SOUND</td>
<td>H10</td>
<td>0041,1</td>
<td></td>
</tr>
<tr>
<td>SRTMR</td>
<td>E8</td>
<td>022B,1</td>
<td></td>
</tr>
<tr>
<td>SSFLAG</td>
<td>E4</td>
<td>02FF,1</td>
<td></td>
</tr>
<tr>
<td>SSKCTL</td>
<td>H32</td>
<td>0232,1</td>
<td></td>
</tr>
<tr>
<td>STACKP</td>
<td>H28</td>
<td>0318,1</td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>H31</td>
<td>0030,1</td>
<td></td>
</tr>
<tr>
<td>STICK0</td>
<td>J1, J7, J8</td>
<td>027B,1</td>
<td></td>
</tr>
<tr>
<td>STICK1</td>
<td>J1, J7, J8</td>
<td>0279,1</td>
<td></td>
</tr>
<tr>
<td>STICK2</td>
<td>J1, J7, J8</td>
<td>027A,1</td>
<td></td>
</tr>
<tr>
<td>STICK3</td>
<td>J1, J7, J8</td>
<td>027B,1</td>
<td></td>
</tr>
<tr>
<td>STICK4</td>
<td>J2, J7, J9</td>
<td>0284,1</td>
<td></td>
</tr>
</tbody>
</table>

Operating System CO16555 — Appendix L
STRIG1  J2, J7, J9  0285, 1
STRIG2  J2, J7, J9  0286, 1
STRIG3  J2, J7, J9  0284, 4
SUBTMP  B48  029E, 1
SWPFLG  B44  007B, 1

TABMAP  B13  02A3, 15
TEMP    H23  023E, 1
TEMP1   D5   0312, 2
TEMP3   D6   0315, 1
TIMER1  D3   030C, 2
TIMER2  D3   0310, 2
TIMFLG  D2, H25  0317, 1
TINDEX  B49  0293, 1
TMPCHR  B33  0050, 1
TMPCOL  B46  02B9, 2
TMPLLT  B47  02A1, 1
TMPROW  B46  02B8, 1
TOADR   B40  0066, 2
TRMSZ   N1   0004, 3
TSTAT   H29  0319, 1
TSTDAT  N2   0007, 1
TXTCOL  B4   0291, 2
TXTMSC  B52  0294, 2
TXTTLD  B53  0296, 6
TXTROW  B4   0290, 1

USAREA  R1   0080, 128

VBREAK  P16  0206, 2
VDSLST  P10  0200, 2
VIMIRG  P13  0216, 2
VINTER  P15  0214, 2
VKEYBD  P17  0208, 2
VPRCED  P14  0202, 2
VSERIN  P18  020A, 2
VSEROC  P20  020E, 2
VSEROR  P19  020C, 2
VTIMR1  P21  0210, 2
VTIMR2  P21  0212, 2
VTIMR4  P21  0214, 2
VVBLKD  P12  0224, 2
VVBLKI  P11  0222, 2

WARMST  N13  0008, 1
WMODE   D14  0289, 1

XMTDON  H24  003A, 1

(ZBUFF  K2)  0043, 2
(ZDRVA  K3)  0045, 2
(ZDGB  G13)  0020, 16
(ZSBA  K4)  0047, 2
ZTEMP1  M13  00F5, 2

OPERATING SYSTEM CO16555 -- Appendix L
### MEMORY ADDRESS ORDERED LIST OF DATABASE VARIABLES

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>VID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-0001</td>
<td>S7</td>
<td>LNZBS</td>
</tr>
<tr>
<td>0002-0003</td>
<td>N6</td>
<td>CASINI</td>
</tr>
<tr>
<td>0004-0006</td>
<td>N1</td>
<td>RAMLO, TRAMSZ</td>
</tr>
<tr>
<td>0007</td>
<td>N2</td>
<td>TSTDAT</td>
</tr>
<tr>
<td>0008</td>
<td>N13</td>
<td>WARMST</td>
</tr>
<tr>
<td>0009</td>
<td>N7</td>
<td>BOOT?</td>
</tr>
<tr>
<td>000A-000B</td>
<td>N12</td>
<td>DOSVEC</td>
</tr>
<tr>
<td>000C-000D</td>
<td>N3</td>
<td>DOSINI</td>
</tr>
<tr>
<td>000E-000F</td>
<td>A3</td>
<td>APPMHI</td>
</tr>
<tr>
<td>0010</td>
<td>P2</td>
<td>POKMSK</td>
</tr>
<tr>
<td>0011</td>
<td>E5</td>
<td>BRKKEY</td>
</tr>
<tr>
<td>0012-0014</td>
<td>P3</td>
<td>RTCLK</td>
</tr>
<tr>
<td>0015-0016</td>
<td>C1</td>
<td>BUFADR</td>
</tr>
<tr>
<td>0017</td>
<td>G23</td>
<td>ICCOMT</td>
</tr>
<tr>
<td>001A-001B</td>
<td>L1</td>
<td>DSKUTL</td>
</tr>
<tr>
<td>001C</td>
<td>F5</td>
<td>PTIMOT</td>
</tr>
<tr>
<td>001D</td>
<td>F3</td>
<td>PBPNT</td>
</tr>
<tr>
<td>001E</td>
<td>F2</td>
<td>PBUFSZ</td>
</tr>
<tr>
<td>001F</td>
<td>F4</td>
<td>PTEMP</td>
</tr>
<tr>
<td>0020</td>
<td>G13, G14</td>
<td>ICHIDZ</td>
</tr>
<tr>
<td>0021</td>
<td>G15</td>
<td>ICDNOZ</td>
</tr>
<tr>
<td>0022</td>
<td>G16</td>
<td>ICCOMZ</td>
</tr>
<tr>
<td>0023</td>
<td>G17</td>
<td>ICOBAS</td>
</tr>
<tr>
<td>0024-0025</td>
<td>G18</td>
<td>ICBALZ, ICBAZ</td>
</tr>
<tr>
<td>0026-0027</td>
<td>G19</td>
<td>ICPTLZ, ICPTHZ</td>
</tr>
<tr>
<td>0028-0029</td>
<td>G20</td>
<td>ICBLLZ, ICBLHZ</td>
</tr>
<tr>
<td>002A-002B</td>
<td>G21</td>
<td>ICAX1Z, ICAX2Z</td>
</tr>
<tr>
<td>002C-002F</td>
<td>G22, G24, G25</td>
<td>ICSPRZ</td>
</tr>
<tr>
<td>0030</td>
<td>H31</td>
<td>STATUS</td>
</tr>
<tr>
<td>0031</td>
<td>H13</td>
<td>CHKSUM</td>
</tr>
<tr>
<td>0032-0033</td>
<td>H16</td>
<td>BUFRLO, BUFFRHI</td>
</tr>
<tr>
<td>0034-0035</td>
<td>H17</td>
<td>BFENLO, BFENHI</td>
</tr>
<tr>
<td>0036</td>
<td>H11</td>
<td>CRETRY</td>
</tr>
<tr>
<td>0037</td>
<td>H12</td>
<td>DRETRY</td>
</tr>
<tr>
<td>0038</td>
<td>H21</td>
<td>BUFRLI</td>
</tr>
<tr>
<td>0039</td>
<td>H22</td>
<td>RECVDN</td>
</tr>
<tr>
<td>003A</td>
<td>H24</td>
<td>XMTDON</td>
</tr>
<tr>
<td>003B</td>
<td>H14</td>
<td>CHKSNT</td>
</tr>
<tr>
<td>003C</td>
<td>H15</td>
<td>NOCKSM</td>
</tr>
<tr>
<td>003D</td>
<td>D11</td>
<td>BPTR</td>
</tr>
<tr>
<td>003E</td>
<td>D13</td>
<td>FTYPE</td>
</tr>
<tr>
<td>003F</td>
<td>D12</td>
<td>FEOF</td>
</tr>
<tr>
<td>0040</td>
<td>D15</td>
<td>FREQ</td>
</tr>
<tr>
<td>0041</td>
<td>H10</td>
<td>SOUNDR</td>
</tr>
<tr>
<td>0042</td>
<td>P1</td>
<td>CRITIC</td>
</tr>
<tr>
<td>0043-0049</td>
<td>K1, K2, K3, K4, K5, ZBUFF, ZBUF, ZDRVA, ZSBA</td>
<td></td>
</tr>
<tr>
<td>004A</td>
<td>N4</td>
<td>CKEY</td>
</tr>
<tr>
<td>004B</td>
<td>N5</td>
<td>CASSBT</td>
</tr>
<tr>
<td>004C</td>
<td>B34</td>
<td>DSTAT</td>
</tr>
</tbody>
</table>

266 OPERATING SYSTEM C016555 -- Appendix L
ATRACT
DRKMSK
COLRSH
TMPCHR
HOLD1
LMARGN
RMARGN
ROWCRS, COLCRS
INDEX
SAVMSC
OLOROW, OLDROW, OLDCHR
OLDADDR
NEWROW, NEUCOL
LOGCOL
ADRESS
MLTTEMP, OPNTMP, TOADR
SAVADR/FRMADR
RAMTOP
BUFCNT
BITMSK
SHFAMT
ROWAC, COLAC
ENDPT
DELTAR, DELTAC
ROWINC, COLINC
SWPFLG
HOLDCH
INSDAT
COUNTR
SEE FLOATING POINT VARIABLE LIST AT END.

6502 STACK
VDSLST
VPRCED
VINTER
VBREAK
VKEYBD
VSRIN
VSROR
VSRORC
VITMR1, VITMR2, VITMR4
VIMIRG
CDTMV1
CDTMV2
CDTMV3, CDTMV4, CDTMV5
VVBLK1
VVBLK2
VVBLK3
CDTMA1
CDTMA2
CDTMA3
CDTMA4
CDTMA5
CDTMA6
022B  E8  SRTIMR
022C  P9  CDTMF4
022D  P28  INTEMP
022E  P9  CDTMF5
022F  P22  SDMCTL
0230-0231  P23  SDLSTL, SDLSTH
0232  H32  SSKCTL
023A  H18  CDEVIC
023B  H19  CCOMND
023C-023D  H20  CAUX1, CAUX2
023E  H23  TEMPE
023F  H30  ERRFLG
0240  N8  DFLAGS
0241  N9  DBSECT
0242-0243  N10  BOOTAD
0244  N11  COLDST
0246  C2  DSKTIM
0247-026E  B51  LINBUF
026F  P24  QPRIORITY
0270-0277  J3  PADDLO -- PADDL7
0278-027B  J1, J7, J8  STICKO -- STICK3
027C-0283  J4  PTRIGO -- PTRIG7
0284-0287  J2, J7, J9  STRINGO -- STRIG3
0289  D14  WMODE
028A  D10  BLIM
028B-028F  S10  unused
0290-0292  B4  TXTROW, TXTCOL
0293  B49  TINDEX
0294-0295  B52  TXTMSC
0296-029B  B53  TXTOLD
029D  B32  HOLD3
029E  B48  SUBTMP
029F  B31  HOLD2
02A0  B28  DMASK
02A1  B47  TMPLBT
02A2  B26  ESCFLG
02A3-02B1  B13  TABMAP
02B2-02B5  B14  LOGMAP
02B6  E9  INVFLG
02B7  B18  FILFLG
02B8-02BA  B46  TMPROW, TMPCOL
02BB  B9  SCRFLG
02BC  B20  HOLD4
02BE  E6  SHFLOK
02BF  B16  BDTSCR
02C0-02C3  B7, P27  PCOLRO -- PCOLR3
02C4-02CB  B8, P27  PCOLRO -- PCOLR4
02E4  A5  RAMSIZ
02E5-02E6  A2  MEMTOP
02E7-02E8  A1  MEMLO
02EA-02ED  Q11  DVSTAT
02EE-02EF  D1  CHBAUDL, CHBAUDH
02F0  B1  CRNISH
02F1  E2  KEYDEL

OPERATING SYSTEM CO16555 -- Appendix L
Appendix L

OPERATING SYSTEM CO16555 -- Appendix L
### Floating Point Package Variables

<table>
<thead>
<tr>
<th>Address</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00D4-00D9</td>
<td>M1</td>
<td>FRO</td>
</tr>
<tr>
<td>00DA-00DF</td>
<td>M2</td>
<td>FRE</td>
</tr>
<tr>
<td>00E0-00E5</td>
<td>M3</td>
<td>FR1</td>
</tr>
<tr>
<td>00E6-00EB</td>
<td>M4</td>
<td>FR2</td>
</tr>
<tr>
<td>00EC</td>
<td>M5</td>
<td>FRX</td>
</tr>
<tr>
<td>00ED</td>
<td>M6</td>
<td>EEXP</td>
</tr>
<tr>
<td>00EE</td>
<td>M7</td>
<td>NSIGN</td>
</tr>
<tr>
<td>00EF</td>
<td>M8</td>
<td>ESIGN</td>
</tr>
<tr>
<td>00F0</td>
<td>M9</td>
<td>FCHRFLG</td>
</tr>
<tr>
<td>00F1</td>
<td>M10</td>
<td>DIGRT</td>
</tr>
<tr>
<td>00F2</td>
<td>M11</td>
<td>CIX</td>
</tr>
<tr>
<td>00F3-00F4</td>
<td>M12</td>
<td>INBUFF</td>
</tr>
<tr>
<td>00F5-00F6</td>
<td>M13</td>
<td>ZTEMP1</td>
</tr>
<tr>
<td>00F7-00F8</td>
<td>M14</td>
<td>ZTEMP4</td>
</tr>
<tr>
<td>00F9-00FA</td>
<td>M15</td>
<td>ZTEMP3</td>
</tr>
<tr>
<td>00FB</td>
<td>M24</td>
<td>RADFLG/DEGFLG</td>
</tr>
<tr>
<td>00FC-00FD</td>
<td>M16</td>
<td>FLPTR</td>
</tr>
<tr>
<td>00FE-00FF</td>
<td>M17</td>
<td>FPTR2</td>
</tr>
<tr>
<td>057E</td>
<td>M18</td>
<td>LBPR1</td>
</tr>
<tr>
<td>057F</td>
<td>M19</td>
<td>LBPR2</td>
</tr>
<tr>
<td>05E0-05FF</td>
<td>M20</td>
<td>LBFEND, LBUFF</td>
</tr>
<tr>
<td>05E0-05E5</td>
<td>M21</td>
<td>PLYARG</td>
</tr>
<tr>
<td>05E6-05EB</td>
<td>M22</td>
<td>FPSCR/FSCR</td>
</tr>
<tr>
<td>05EC-05F1</td>
<td>M23</td>
<td>FPSCR1/SCR1</td>
</tr>
</tbody>
</table>
INDEX

The subject index contains three forms of references:

Section number, such as '3.'
Appendix, such as 'App B'
Variable ID from Appendix L, such as 'B7'.

ATARI standards 12
ATASCII B54-55, 5, App D-G
attract mode B10-12, 6.
bit mapped graphics B28-B29, 5, App H
blackboard mode 3, N12, 7, 12
BNF 1
boot 3, 4, N3-10, 5, 7, 10
BREAK E5, 6, 12
cartridge 3, 4, 7, 10
cassette baud rate determine D1-D7
cassette-boot 3, N3-10, 7, 10
cassette device D1-D15, 3, 5
Cassette Handler (C) 5
CID (Central I/O Utility) Q1-25, 5, 9
CID/user interface Q1-11, 5, App A, App B
CID/Handler interface Q12-22, 9
CLOSE I/O command 5, 9
coldstart (see 'Power-up') B7-8, 5, 6
color control B26-27, 5, App D
control characters P1, 6
critical section B1-4, 5
cursor database 4
DCB (Device Control Block) H1-9, 5, 9
DELETE I/O command 5
development system 13
device/filename specification 5
Device Handler 5, 9
device table 2, Q12, 5, 7, 9
disk-boot 3, N3-10, 5, 7, 10
disk device 5
Disk File Manager (D) K1-5, 5
Disk Handler (resident) C1-2, 5
display device (screen) B54-55, 5, App E, App H
Display Handler (S) B1-55, 5
display list 4, P10
DOS (Disk Utilities) L1, 12
DRAW I/O command B17-25, 5
driving controller J8-9
Educational System Format Cassettes 5
error handling G5, H5, H11-12, 9, App B-C

OPERATING SYSTEM CO16555 -- INDÈX 271
EOF (end-of-file) 5

File Management System 5
FILL I/O command B17-25, 5
floating point package 2, 4, M1-24, 8, App J
FORMAT I/O command 5
free memory 4, A1-3, R1-2, 4, 7
game controllers 3, J1-9, 6, 11
GET CHARACTER I/O command 5, 9
GET RECORD I/O command 5, 9
GET STATUS I/O command G11, 5, 9

Handler (see 'device handler' and individual device handlers)

initialization, cartridge 7
initialization, Handler 7, 9
initialization, interrupt 6
initialization, system 4, 7, 10
internal display code 5, B54
interrupts 2, P1-28, 6
interrupt mask P2, 6
inverse video (display) E9, 5
I/O 2, 4, 5, 9
IOCB (I/O Control Block) G1-10, 5, 9
I/O retry logic H11-12

joystick J1-2

keyboard Autorepeat E8
keyboard device 5
Keyboard Handler (K) E1-9, 5, App F
keyboard key debouncing E1-3

light pen 11, App J
LNBUG 13
LOCK I/O command 5
logical text lines (screen) B14-15, 5

memory (see 'RAM', 'ROM' and 'free memory')
memory dynamics A1-5, N1-2, 4, 5, 7
memory map 4

NOTE I/O command 5

OPEN I/O command 5, 9

paddle J3-4
page 0 4, M1-17, R1, 9
page 1 4, 9
peripheral devices 3
POINT I/O command 5
Power-up 2, N1-13, 4, 7, 12
printer device 5, App G

OPERATING SYSTEM CO16555 -- INDEX
<table>
<thead>
<tr>
<th>Printer Handler (P)</th>
<th>PUT CHARACTER I/O command</th>
<th>PUT RECORD I/O command</th>
</tr>
</thead>
<tbody>
<tr>
<td>program development</td>
<td>5, 9</td>
<td>5, 9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RAM</th>
<th>record (I/O)</th>
<th>RENAME I/O command</th>
<th>RESET</th>
<th>ROM (OS)</th>
<th>RS-232-C Handler (R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3, 4, 9</td>
<td>5</td>
<td>2, N1-13, 6, 7, 12</td>
<td>1, 4</td>
<td>5, 9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Screen Editor (E)</th>
<th>screen margins</th>
<th>screen modes</th>
<th>scrolling (text)</th>
<th>serial I/O bus</th>
<th>[SHIFT]/CONTROL lock</th>
<th>SIO (Serial bus I/O Utility)</th>
<th>sound control (SIO)</th>
<th>SPECIAL I/O commands</th>
<th>split screen</th>
<th>start/stop (display)</th>
<th>stage 1 VBLANK process</th>
<th>stage 2 VBLANK process</th>
<th>tabs (Screen Editor)</th>
<th>timeout (device)</th>
<th>timers (system)</th>
<th>UNLOCK I/O command</th>
<th>user workspace</th>
<th>vectors, RAM</th>
<th>vectors, ROM</th>
<th>vertical blank interrupt</th>
<th>warmstart (see 'RESET')</th>
<th>wild-card (disk filename)</th>
<th>ZIOCB (Zero-page IOCB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1-55, 5</td>
<td>B5-6, 5, 7</td>
<td>4, 5, App H</td>
<td>B9, 5</td>
<td>3, 5, 9, App I</td>
<td>E6-7, 5</td>
<td>H1-32, P13-21, 5, 9, App C</td>
<td>H10, 11</td>
<td>5, 9</td>
<td>B16, 5</td>
<td>E4, 5, 12</td>
<td>P3-5, 6</td>
<td>P6-9, P22-27, 6</td>
<td>B13, 5</td>
<td>H25-27, 9</td>
<td>P3-9, 6</td>
<td>5</td>
<td>5</td>
<td>P5, P7, P10-21, 6, 9</td>
<td>5, 9, App J</td>
<td>5</td>
<td>G13-22, 9, 0020, 16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OPERATING SYSTEM CO16555 -- INDEX

273
TO ALL PERSONS RECEIVING THIS DOCUMENT

Reproduction is forbidden without the specific written permission of ATARI, INC. Sunnyvale, CA 94086. No right to reproduce this document, nor the subject matter thereof, is granted unless by written agreement with, or written permission from the Corporation.

Every effort has been made to ensure that this manual accurately documents this product of the ATARI Home Computer Division. However, due to the ongoing improvement and update of the computer software and hardware, ATARI, INC. cannot guarantee the accuracy of printed material after the date of publication and disclaims liability for changes, errors, or omissions.
LIST X
; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
; SYSTEM LISTING, MODIFIED TO ASSEMBLE ON THE MICROTEC CROSS
; ASSEMBLER.
; THIS VERSION IS THE ONE WHICH WAS BURNED INTO ROM.
; THERE IS A RESIDUAL PIECE OF CODE WHICH IS FOR LNBUG. THIS
; IS AT LOCATION $9000 WHICH IS NOT IN ROM.
; THIS IS THE REVISION B EPROM VERSION
.PAGE

; COLLEEN OPERATING SYSTEM EQUATE FILE:

; NTSC/PAL ASSEMBLY FLAG

PALFLG = 0 ; 0 = NTSC 1 = PAL

; MODULE ORIGIN TABLE

CHRORG = $E000 ; CHARACTER SET
VECTBL = $E400 ; VECTOR TABLE
VCTABL = $E480 ; RAM VECTOR INITIAL VALUE TABLE
CIOORG = $E466 ; CENTRAL I/O HANDLER
INTORG = $E6D5 ; INTERRUPT HANDLER
SIDROR = $E944 ; SERIAL I/O DRIVER
EDE = $EDE ; DISK HANDLER
PRNROR = $EE78 ; PRINTER HANDLER
CASORR = $EF41 ; CASSETTE HANDLER
MONORG = $E4A6 ; MONITOR/POWER UP MODULE
KBDORG = $E6D5 ; KEYBOARD/DISPLAY HANDLER

; VECTOR TABLE

; HANDLER ENTRY POINTS ARE CALLED OUT IN THE FOLLOWING VECTOR TABLE. THESE ARE THE ADDRESSES MINUS ONE.

; EXAMPLE FOR EDITOR

; EDITOR

E400 OPEN
2 CLOSE
4 GET
6 PUT
8 STATUS
A SPECIAL
C JUMP TO POWER ON INITIALIZATION ROUTINE
.F NOT USED

E400 EDITRV = $E400 ; EDITOR
E410 SCRENV = $E410 ; TELEVISION SCREEN
E420 KEYBDV = $E420 ; KEYBOARD
E430 PRINTV = $E430 ; PRINTER
E440 CASETV = $E440 ; CASSETTE

; JUMP VECTOR TABLE

; THE FOLLOWING IS A TABLE OF JUMP INSTRUCTIONS
64: ; TO VARIOUS ENTRY POINTS IN THE OPERATING SYSTEM.
66: DISKIV = $E450 ; DISK INITIALIZATION
67: DSKINV = $E453 ; DISK INTERFACE
68: CIOV = $E456 ; CENTRAL INPUT OUTPUT ROUTINE
69: SIDV = $E459 ; SERIAL INPUT OUTPUT ROUTINE
70: SETVBV = $E45C ; SET SYSTEM TIMERS ROUTINE
71: SYSVBV = $E45F ; SYSTEM VERTICAL BLANK CALCULATIONS
72: XITVBV = $E462 ; EXIT VERTICAL BLANK CALCULATIONS
73: SIDINV = $E465 ; SERIAL INPUT OUTPUT INITIALIZATION
74: SENDEV = $E468 ; SEND ENABLE ROUTINE
75: INTINV = $E46B ; INTERRUPT HANDLER INITIALIZATION
76: CIOINV = $E46E ; CENTRAL INPUT OUTPUT INITIALIZATION
77: E471 ; BLACKBOARD MODE
78: E474 ; WARM START ENTRY POINT
79: COLDsv = $E477 ; COLD START ENTRY POINT
80: Rblokv = $E47A ; CASSETTE READ BLOCK ENTRY POINT VECTOR
81: E47D ; CASSETTE OPEN FOR INPUT VECTOR
82: VCTABL = $E480
83: ; OPERATING SYSTEM EQUATES
84: ; COMMAND CODES FOR I0CB
87: OPEN = 3 ; OPEN FOR INPUT/OUTPUT
88: GETREC = 5 ; GET RECORD (TEXT)
89: GETCHR = 7 ; GET CHARACTER(S)
90: PUTREC = 9 ; PUT RECORD (TEXT)
91: PUTCHR = $8 ; PUT CHARACTER(S)
92: CLOSE = $C ; CLOSE DEVICE
93: STATIS = $D ; STATUS REQUEST
94: SPECIL = $E ; BEGINNING OF SPECIAL ENTRY COMMANDS
96: ; SPECIAL ENTRY COMMANDS
98: DRAWLN = $11 ; DRAW LINE
99: DRAWLINE = $12 ; DRAW LINE WITH RIGHT FILL
100: RENAME = $20 ; RENAME DISK FILE
101: DELETE = $21 ; DELETE DISK FILE
102: FORMAT = $22 ; FORMAT
103: LOCKFL = $23 ; LOCK FILE TO READ ONLY
104: UNLOCK = $24 ; UNLOCK LOCKED FILE
105: POINT = $25 ; POINT SECTOR
106: NOTE = $26 ; NOTE SECTOR
107: IOCFFRE = $FF ; IOC "FREE"
108: ; AUX1 EQUATES
109: ; () INDICATES WHICH DEVICES USE BIT
110: APPEND = $1 ; OPEN FOR WRITE APPEND (D), OR SCREEN READ (D)
111: DIRECT = $2 ; OPEN FOR DIRECTORY ACCESS (D)
112: OPNIN = $4 ; OPEN FOR INPUT (ALL DEVICES)
113: OPNOT = $8 ; OPEN FOR OUTPUT (ALL DEVICES)
114: OPNINO = OPNIN+OPNOT ; OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
115: MXDMOD = $10 ; OPEN FOR MIXED MODE (E,S)
116: INSLCR = $20 ; OPEN WITHOUT CLEARING SCREEN (E,S)
### Device Names

- `SCREDT` = 'E'  
  - Screen Editor (R/W)
- `KBD` = 'K'  
  - Keyboard (R only)
- `DISPLY` = 'S'  
  - Screen Display (R/W)
- `PRINTR` = 'P'  
  - Printer (W only)
- `CASSET` = 'C'  
  - Cassette
- `MODEM` = 'M'  
  - Modem
- `DISK` = 'D'  
  - Disk (R/W)

### System EOL (Carriage Return)

- `CR` = $9B

### Operating System Status Codes

- **SUCCESS** = $01
  - Successful Operation
- **BREAK KEY ABORT** = $80
- **IOCB ALREADY OPEN** = $81
- **NON-EXISTANT DEVICE** = $82
- **IOCB OPENED FOR WRITE ONLY** = $83
- **INVALID COMMAND** = $84
- **DEVICE OR FILE NOT OPEN** = $85
- **INVALID IOCB NUMBER** = $86
- **END OF FILE** = $87
- **TRUNCATED RECORD** = $88
- **PERIPHERAL DEVICE TIME OUT** = $89
- **DEVICE DOES NOT ACKNOWLEDGE COMMAND** = $8A
- **PERIPHERAL DEVICE ERROR** = $8B
  - Operation Not Comp
- **BAD SCREEN MODE NUMBER** = $8C
- **FUNCTION NOT IMPLEMENTED IN HANDLER** = $8D
- **INSUFFICIENT MEMORY FOR SCREEN MODE** = $8E

### Page Zero RAM Assignments

- **LINZBS**: .RES 2
  - Linbug Ram (Will Be Replaced By Monitor Ram)
- **CASINI**: .RES 2
  - Cassette Init Location
- **RAMLO**: .RES 2
  - Ram Pointer For Memory Test
- **TRAMSZ**: .RES 1
  - Temporary Register For Ram Size
ERR LINE ADDR B1 B2 B3 B4
172 0007 TSTDAT: .RES 1 ;RAM TEST DATA REGISTER
173
174 ; CLEARED ON COLDSTART ONLY
175 0008 WARMST: .RES 1 ;WARM START FLAG
176 0009 BOOT?: .RES 1 ;SUCCESSFUL BOOT FLAG
177 000A DOSVEC: .RES 2 ;DISK SOFTWARE START VECTOR
178 000C DOSINI: .RES 2 ;DISK SOFTWARE INIT ADDRESS
179 000E APPMHI: .RES 2 ;APPLICATIONS MEMORY HI LIMIT
180
181 ; CLEARED ON COLD OR WARM START
182 0010 INTZBS =# ;INTERRUPT HANDLER
183 0010 POKMSK: .RES 1 ;SYSTEM MASK FOR POKEY IRQ ENABLE
184 0011 BRKKEY: .RES 1 ;BREAK KEY FLAG
185 0012 RTCLK: .RES 3 ;REAL TIME CLOCK (IN 16 MSEC UNITS)
186
187 0015 BUFADR: .RES 2 ;INDIRECT BUFFER ADDRESS REGISTER
188
189 0017 ICCOMT: .RES 1 ;COMMAND FOR VECTOR
190
191 0018 DSKFMS: .RES 2 ;DISK FILE MANAGER POINTER
192 001A DSKUTL: .RES 2 ;DISK UTILITIES POINTER
193
194 001C PTIMDT: .RES 1 ;PRINTER TIME OUT REGISTER
195 001D PBPN: .RES 1 ;PRINT BUFFER POINTER
196 001E PBUFSZ: .RES 1 ;PRINT BUFFER SIZE
197 001F PTEMP: .RES 1 ;TEMPORARY REGISTER
198
199 0020 IIOC =# ;ZERO PAGE I/O CONTROL BLOCK
200 0010 IIOCBSZ = 16 ;NUMBER OF BYTES PER IIOC
201 0080 MAXIOC = 8* IIOCBSZ ;LENGTH OF THE IIOC AREA
202 0020 IIOCBS =# ;IIOCBS
203 0020 ICHIDZ: .RES 1 ;HANDLER INDEX NUMBER (FF = IIOC BS FREE)
204 0021 ICNOZ: .RES 1 ;DEVICE NUMBER (DRIVE NUMBER)
205 0022 ICCOMZ: .RES 1 ;COMMAND CODE
206 0023 ICSTA1: .RES 1 ;STATUS OF LAST IIOC ACTION
207 0024 ICBALZ: .RES 1 ;BUFFER ADDRESS LOW BYTE
208 0025 ICBAHZ: .RES 1 ;DEVICE NUMBER (DRIVE NUMBER)
209 0026 ICPTLZ: .RES 1 ;PUT BYTE ROUTINE ADDRESS - 1
210 0027 ICPTHZ: .RES 1 ;PUT BYTE ROUTINE ADDRESS - 1
211 0028 ICBLLZ: .RES 1 ;BUFFER LENGTH LOW BYTE
212 0029 ICBHLZ: .RES 1 ;BUFFER LENGTH LOW BYTE
213 002A ICAX1Z: .RES 1 ;AUXILIARY INFORMATION FIRST BYTE
214 002B ICAX2Z: .RES 1 ;AUXILIARY INFORMATION FIRST BYTE
215 002C ICSPRZ: .RES 4 ;TWO SPARE BYTES (CIO LOCAL USE)
216 002E ICIDNO = ICSPRZ+2 ;IIOC NUMBER X 16
217 002F CIOCHR = ICSPRZ+3 ;CHARACTER BYTE FOR CURRENT OPERATION
218
219 0030 STATUS: .RES 1 ;INTERNAL STATUS STORAGE
220 0031 CHKSUM: .RES 1 ;CHECKSUM (SINGLE BYTE SUM WITH CARRY)
221 0032 BUFRL0: .RES 1 ;POINTER TO DATA BUFFER (LO BYTE)
222 0033 BUFRL1: .RES 1 ;POINTER TO DATA BUFFER (HI BYTE)
223 0034 BFNLO: .RES 1 ;NEXT BYTE PAST END OF THE DATA BUFFER (LO B
224 0035 BFNHI: .RES 1 ;NEXT BYTE PAST END OF THE DATA BUFFER (HI B
225 0036 CRETRY: .RES 1 ;NUMBER OF COMMAND FRAME RETRIES
ERR LINE ADDR B1 B2 B3 B4

226 0037 DRETRY: .RES 1 : NUMBER OF DEVICE RETRIES
227 003B BUFRL: .RES 1 : DATA BUFFER FULL FLAG
228 0039 RECVDN: .RES 1 : RECEIVE DONE FLAG
229 003A XMTDON: .RES 1 : TRANSMISSION DONE FLAG
230 003B CHKSGT: .RES 1 : CHECKSUM SENT FLAG
231 003C NOCKSM: .RES 1 : NO CHECKSUM FOLLOWS DATA FLAG
232 ;
233 ;
234 003D BPTR: .RES 1
235 003E FTYPE: .RES 1
236 003F FREQ: .RES 1
237 0040 SOUNDR: .RES 1 : NOISY I/O FLAG. (ZERO IS QUIET)
238 0041 CRITIC: .RES 1 : DEFINES CRITICAL SECTION (CRITICAL IF NON-Z
239 0042 DSTAT: .RES 1
240 ;
241 0043 FMS2PO: .RES 7 : DISK FILE MANAGER SYSTEM ZERO PAGE
242 ;
243 ;
244 004A CKEY: .RES 1 :_FLAG SET WHEN GAME START PRESS
245 004B CASBT: .RES 1 : CASSETTE BOOT FLAG
246 004C DSTAT: .RES 1 : DISPLAY STATUS
247 ;
248 004D ATRACT: .RES 1 : ATRACT FLAG
249 004E DRKMSK: .RES 1 : DARK ATRACT MASK
250 004F COLRSH: .RES 1 : ATRACT COLOR SHIFTER (EOR'ED WITH PLAYFIELD
251 ;
252 0052 LEDGE = 2 : L_MARGIN'S VALUE AT COLD START
253 0057 REDGE = 39 : R_MARGIN'S VALUE AT COLD START
254 0050 TMPCHR: .RES 1
255 0051 HOLD: .RES 1
256 0052 LMARGIN: .RES 1 : LEFT MARGIN (SET TO 1 AT POWER ON)
257 0053 RMARGIN: .RES 1 : RIGHT MARGIN (SET TO 38 AT POWER ON)
258 0054 ROWCRS: .RES 1
259 0055 COLCRS: .RES 2
260 0057 DINDEX: .RES 1
261 0058 SAVMSC: .RES 2
262 005A OLDCOL: .RES 1
263 005B OLDCHR: .RES 2
264 005D OLCHR: .RES 1 : DATA UNDER CURSOR
265 005E OLADR: .RES 2
266 0060 NEWROW: .RES 1 : POINT DRAW GOES TO
267 0061 NEWCOL: .RES 2
268 0063 LOGCOL: .RES 1 : POINTS AT COLUMN IN LOGICAL LINE
269 0064 ADRESS: .RES 2
270 0066 MLT_TMP: .RES 2
271 0066 OPNTMP = MLLTMP : FIRST BYTE IS USED IN OPEN AS TEMP
272 006B SAVADR: .RES 2
273 006A RAMTOP: .RES 1 : RAM SIZE DEFINED BY POWER ON LOGIC
274 006E BUFCNT: .RES 1 : BUFFER COUNT
275 0068 BUFSTR: .RES 2 : EDITOR GETCH POINTER
276 006C BMASK: .RES 1 : BIT MASK
277 006F SHTAMP: .RES 1
278 0070 ROWAC: .RES 2
279 0072 COLAC: .RES 2
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>280</td>
<td>0074</td>
<td>ENDPT: .RES 2</td>
</tr>
<tr>
<td>281</td>
<td>0076</td>
<td>DELTAR: .RES 1</td>
</tr>
<tr>
<td>282</td>
<td>0077</td>
<td>DELTAC: .RES 2</td>
</tr>
<tr>
<td>283</td>
<td>0079</td>
<td>ROWINC: .RES 1</td>
</tr>
<tr>
<td>284</td>
<td>007A</td>
<td>COLINC: .RES 1</td>
</tr>
<tr>
<td>285</td>
<td>007B</td>
<td>SWPFLO: .RES 1</td>
</tr>
<tr>
<td>286</td>
<td>007C</td>
<td>HOLDC: .RES 1</td>
</tr>
<tr>
<td>287</td>
<td>007D</td>
<td>INSDAT: .RES 1</td>
</tr>
<tr>
<td>288</td>
<td>007E</td>
<td>COUNTR: .RES 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>289</td>
<td></td>
<td></td>
</tr>
<tr>
<td>290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>291</td>
<td></td>
<td></td>
</tr>
<tr>
<td>292</td>
<td></td>
<td></td>
</tr>
<tr>
<td>293</td>
<td></td>
<td></td>
</tr>
<tr>
<td>294</td>
<td></td>
<td></td>
</tr>
<tr>
<td>295</td>
<td></td>
<td></td>
</tr>
<tr>
<td>296</td>
<td></td>
<td></td>
</tr>
<tr>
<td>297</td>
<td></td>
<td></td>
</tr>
<tr>
<td>298</td>
<td></td>
<td></td>
</tr>
<tr>
<td>299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>301</td>
<td></td>
<td></td>
</tr>
<tr>
<td>302</td>
<td></td>
<td></td>
</tr>
<tr>
<td>303</td>
<td></td>
<td></td>
</tr>
<tr>
<td>304</td>
<td></td>
<td></td>
</tr>
<tr>
<td>305</td>
<td></td>
<td></td>
</tr>
<tr>
<td>306</td>
<td></td>
<td></td>
</tr>
<tr>
<td>307</td>
<td></td>
<td></td>
</tr>
<tr>
<td>308</td>
<td></td>
<td></td>
</tr>
<tr>
<td>309</td>
<td></td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>0200</td>
<td>*<em>INTABS =</em></td>
</tr>
<tr>
<td>311</td>
<td>0200</td>
<td>INTERRUPT RAM</td>
</tr>
<tr>
<td>312</td>
<td>0200</td>
<td>DISPLAY LIST NMI VECTOR</td>
</tr>
<tr>
<td>313</td>
<td>0200</td>
<td>PROCEED LINE IRQ VECTOR</td>
</tr>
<tr>
<td>314</td>
<td>0200</td>
<td>INTERRUPT LINE IRQ VECTOR</td>
</tr>
<tr>
<td>315</td>
<td>0200</td>
<td>SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR</td>
</tr>
<tr>
<td>316</td>
<td>0200</td>
<td>POKEY KEYBOARD IRQ VECTOR</td>
</tr>
<tr>
<td>317</td>
<td>0200</td>
<td>POKEY SERIAL INPUT READY IRQ</td>
</tr>
<tr>
<td>318</td>
<td>0200</td>
<td>POKEY SERIAL OUTPUT READY IRQ</td>
</tr>
<tr>
<td>319</td>
<td>0200</td>
<td>POKEY SERIAL OUTPUT COMPLETE IRQ</td>
</tr>
<tr>
<td>320</td>
<td>0210</td>
<td>POKEY TIMER 1 IRQ</td>
</tr>
<tr>
<td>321</td>
<td>0212</td>
<td>POKEY TIMER 2 IRQ</td>
</tr>
<tr>
<td>322</td>
<td>0214</td>
<td>POKEY TIMER 4 IRQ</td>
</tr>
<tr>
<td>323</td>
<td>0216</td>
<td>IMMEDIATE IRQ VECTOR</td>
</tr>
<tr>
<td>324</td>
<td>0218</td>
<td>COUNT DOWN TIMER 1</td>
</tr>
<tr>
<td>325</td>
<td>021A</td>
<td>COUNT DOWN TIMER2</td>
</tr>
<tr>
<td>326</td>
<td>021C</td>
<td>COUNT DOWN TIMER 3</td>
</tr>
<tr>
<td>327</td>
<td>021E</td>
<td>COUNT DOWN TIMER 4</td>
</tr>
<tr>
<td>328</td>
<td>0220</td>
<td>COUNT DOWN TIMER 5</td>
</tr>
<tr>
<td>329</td>
<td>0222</td>
<td>IMMEDIATE VERTICAL BLANK NMI VECTOR</td>
</tr>
<tr>
<td>330</td>
<td>0224</td>
<td>DEFERRED VERTICAL BLANK NMI VECTOR</td>
</tr>
<tr>
<td>331</td>
<td>0226</td>
<td>COUNT DOWN TIMER 1 JSR ADDRESS</td>
</tr>
<tr>
<td>332</td>
<td>0228</td>
<td>COUNT DOWN TIMER 2 JSR ADDRESS</td>
</tr>
<tr>
<td>333</td>
<td>022A</td>
<td>COUNT DOWN TIMER 3 FLAG</td>
</tr>
</tbody>
</table>

**BO - FF ARE RESERVED FOR USER APPLICATIONS**

**NOTE: SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS**

**PAGE 1 - STACK**

**PAGE TWO RAM ASSIGNMENTS**
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>6500 ASSEMBLER VER 1.0MR</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>334</td>
<td>022C</td>
<td>CDTMF4: .RES 1</td>
<td>COUNT DOWN TIMER 4 FLAG</td>
<td></td>
</tr>
<tr>
<td>335</td>
<td>022D</td>
<td>INTEMP: .RES 1</td>
<td>IAN'S TEMP (RENAMED FROM TI BY POPULAR DEMA)</td>
<td></td>
</tr>
<tr>
<td>336</td>
<td>022E</td>
<td>CDTMF5: .RES 1</td>
<td>COUNT DOWN TIMER FLAG 5</td>
<td></td>
</tr>
<tr>
<td>337</td>
<td>022F</td>
<td>SDMCTL: .RES 1</td>
<td>SAVE DMACTL REGISTER</td>
<td></td>
</tr>
<tr>
<td>338</td>
<td>0230</td>
<td>SDLSTL: .RES 1</td>
<td>SAVE DISPLAY LIST LOW  BYTE</td>
<td></td>
</tr>
<tr>
<td>339</td>
<td>0231</td>
<td>SDLSTH: .RES 1</td>
<td>SAVE DISPLAY LIST HI  BYTE</td>
<td></td>
</tr>
<tr>
<td>340</td>
<td>0232</td>
<td>SSKCTL: .RES 1</td>
<td>RSKCTL REGISTER RAM</td>
<td></td>
</tr>
<tr>
<td>341</td>
<td>0233</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>342</td>
<td>0234</td>
<td>LPENH: .RES 1</td>
<td>LIGHT PEN HORIZONTAL VALUE</td>
<td></td>
</tr>
<tr>
<td>343</td>
<td>0235</td>
<td>LPENV: .RES 1</td>
<td>LIGHT PEN VERTICAL VALUE</td>
<td></td>
</tr>
<tr>
<td>344</td>
<td>0236</td>
<td>BRKKY: .RES 2</td>
<td>BREAK KEY VECTOR</td>
<td></td>
</tr>
<tr>
<td>345</td>
<td>0237</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>346</td>
<td>0238</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>347</td>
<td>0239</td>
<td>CDEVIC: .RES 1</td>
<td>COMMAND FRAME BUFFER - DEVICE</td>
<td></td>
</tr>
<tr>
<td>348</td>
<td>023A</td>
<td>CCOMND: .RES 1</td>
<td>COMMAND</td>
<td></td>
</tr>
<tr>
<td>349</td>
<td>023B</td>
<td>CAUX1: .RES 1</td>
<td>COMMAND AUX BYTE 1</td>
<td></td>
</tr>
<tr>
<td>350</td>
<td>023C</td>
<td>CAUX2: .RES 1</td>
<td>COMMAND AUX BYTE 2</td>
<td></td>
</tr>
<tr>
<td>351</td>
<td>023D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>352</td>
<td>023E</td>
<td>TEMP: .RES 1</td>
<td>TEMPORARY RAM CELL</td>
<td></td>
</tr>
<tr>
<td>353</td>
<td>023F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>354</td>
<td>0240</td>
<td>ERRFLG: .RES 1</td>
<td>ERROR FLAG - ANY DEVICE Error EXCEPT TIME 0</td>
<td></td>
</tr>
<tr>
<td>355</td>
<td>0241</td>
<td>DFLAGS: .RES 1</td>
<td>DISK FLAGS FROM SECTOR ONE</td>
<td></td>
</tr>
<tr>
<td>356</td>
<td>0242</td>
<td>DBSECT: .RES 1</td>
<td>NUMBER OF DISK BOOT SECTORS</td>
<td></td>
</tr>
<tr>
<td>357</td>
<td>0243</td>
<td>BOOTADR: .RES 2</td>
<td>ADDRESS WHERE DISK BOOT LOADER WILL BE PUT</td>
<td></td>
</tr>
<tr>
<td>358</td>
<td>0244</td>
<td>COLDST: .RES 1</td>
<td>COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)</td>
<td></td>
</tr>
<tr>
<td>359</td>
<td>0245</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>360</td>
<td>0246</td>
<td>DSKTIM: .RES 1</td>
<td>DISK TIME OUT REGISTER</td>
<td></td>
</tr>
<tr>
<td>361</td>
<td>0247</td>
<td>LINBUF: .RES 40</td>
<td>CHAR LINE BUFFER</td>
<td></td>
</tr>
<tr>
<td>362</td>
<td>0248</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>363</td>
<td>0249</td>
<td>GPRIO: .RES 1</td>
<td>GLOBAL PRIORITY CELL</td>
<td></td>
</tr>
<tr>
<td>364</td>
<td>024A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>024B</td>
<td>PTRIGO: .RES 1</td>
<td>PADDLE TRIGGER O</td>
<td></td>
</tr>
<tr>
<td>366</td>
<td>024C</td>
<td>PTRIG1: .RES 1</td>
<td>JOYSTICK O RAM CELL</td>
<td></td>
</tr>
<tr>
<td>367</td>
<td>024D</td>
<td>PTRIG2: .RES 1</td>
<td>JOYSTICK O RAM CELL</td>
<td></td>
</tr>
<tr>
<td>368</td>
<td>024E</td>
<td>PTRIG3: .RES 1</td>
<td>JOYSTICK O RAM CELL</td>
<td></td>
</tr>
<tr>
<td>369</td>
<td>024F</td>
<td>PTRIG4: .RES 1</td>
<td>JOYSTICK O RAM CELL</td>
<td></td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4

442
443 02C9 .RES 23 ; SPARE
444
445
446
447 02E0 GLBABS == ; GLOBAL VARIABLES
448
449 02E0 .RES 4 ; SPARE
450
451 02E4 RAMSIZ: .RES 1 ; RAM SIZE (HI BYTE ONLY)
452 02E5 MEMTOP: .RES 2 ; TOP OF AVAILABLE USER MEMORY
453 02E7 MEMLO: .RES 1 ; BOTTOM OF AVAILABLE USER MEMORY
454 02E9 .RES 1 ; SPARE
455 02EA DVSTAT: .RES 4 ; STATUS BUFFER
456 02EE CBAUDL: .RES 1 ; CASSETTE BAUD RATE LOW BYTE
457 02EF CBAUDH: .RES 1 ;
458
459 02F0 CRSINH: .RES 1 ; CURSOR INHIBIT (00 = CURSOR ON)
460 02F1 KEYDEL: .RES 1 ; KEY DELAY
461 02F2 CH1: .RES 1 ;
462
463 02F3 CHACT: .RES 1 ; CHACTL REGISTER RAM
464 02F4 CHBAS: .RES 1 ; CHBAS REGISTER RAM
465
466 02F5 .RES 5 ; SPARE BYTES
467
468 02FA CHAR: .RES 1 ; ATASCII CHARACTER
469 02FB ATACHR: .RES 1 ; ATASCII CHARACTER FOR KEYBOARD
470 02FC CH: .RES 1 ; GLOBAL VARIABLE FOR KEYBOARD
471 02FD FILDAT: .RES 1 ; RIGHT FILL DATA (DRAW)
472 02FE DSPFLG: .RES 1 ; DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
473 02FF SSFLAG: .RES 1 ; START/STOP FLAG FOR PAGING (CNTL 1). CLEAR
474
475
476
477
478
479
480
481 ; PAGE THREE RAM ASSIGNMENTS
482
483 0300 DCB == ; DEVICE CONTROL BLOCK
484 0300 DDEVIC: .RES 1 ; PERIPHERAL UNIT 1 BUS I.D. NUMBER
485 0301 DUNIT: .RES 1 ; UNIT NUMBER
486 0302 DCOMND: .RES 1 ; BUS COMMAND
487 0303 DSTATS: .RES 1 ; COMMAND TYPE/STATUS RETURN
488 0304 DBUFLO: .RES 1 ; DATA BUFFER POINTER LOW BYTE
489 0305 DBUFIH: .RES 1 ;
490 0306 DTIMLO: .RES 1 ; DEVICE TIME OUT IN 1 SECOND UNITS
491 0307 DUNUSE: .RES 1 ; UNUSED BYTE
492 0308 DBYTEO: .RES 1 ; NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
493 0309 DBYTEH: .RES 1 ;
494 030A DAUX1: .RES 1 ; COMMAND AUXILIARY BYTE 1
495 030B DAUX2: .RES 1 ;
ERR LINE ADDR B1 B2 B3 B4

6500 ASSEMBLER VER 1.0MR

PAGE 11

496
497 030C
498 030D
499 030E
500 030F
501
502 0310
503 0311
504 0312
505 0313
506 0314
507 0315
508 0316
509 0317
510 0318
511 0319
512 031A
513 031B
514 031C
515 031D
516 031E
517 031F
518 0320
519 0321
520 0322
521 0323
522 0324
523 0325
524 0326
525 0327
526 0328
527 0329
528 032A
529 032B
530 032C
531 032D
532 032E
533 032F
534 0330
535 0331
536 0332
537 0333
538 0334
539 0335
540 0336
541 0337
542 0338
543 0339
544 033A
545 033B
546 033C
547 033D
548 033E
549 033F

i ORDERED ACCESS BLOCKS

i I/O CONTROL BLOCKS

i HANDLER INDEX NUMBER (FF = I/OCB FREE)

i DEVICE NUMBER (DRIVE NUMBER)

i COMMAND CODE

i STATUS OF LAST I/OCB ACTION

i BUFFER ADDRESS LOW BYTE

i AUXILIARY INFORMATION FIRST BYTE

i FOUR SPARE BYTES

i PRINTER BUFFER

i SPARE BYTES

i USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE

i SPARE
PAGE FIVE RAM ASSIGNMENTS

PAGE FIVE IS RESERVED AS A USER WORK SPACE

NOTE: SEE FLOATING POINT SUBROUTINE AREA FOR PAGE FIVE CELLS

PAGE SIX RAM ASSIGNMENTS

PAGE SIX IS RESERVED AS A USER'S USER WORK SPACE

FLOATING POINT SUBROUTINES

FPREC = 6 ;FLOAING PT PRECISION (# OF BYTES)

;IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR

AFP = $DB00 ;ASCII->FLOATING POINT (FP)

INBUFF+CIX -> FRO, CIX, CARRY

FASC = $DBE6 ;FP -> ASCII FRO->LBUFF (INBUFF)

IFP = $D9AA ;INTEGER -> FP

O-$FFFF (LSB,MSB) IN FRO,FRO+1->FRO

FPI = $D9D2 ;FP -> INTEGER FRO -> FRO,FRO+1, CARRY

$DA60 ;FRO <- FRO - FR1 .CARRY

$DA66 ;FRO <- FRO + FR1 .CARRY

$DD89 ;FLOAING LOAD REGO FRO <- (X,Y)

$DD8D ;FLODOP = " " " FRO <- (FLPTR)

$DD9B ; " " REO1 FR1 <- (X,Y)

$DD9C ; " " FR1 <-(FLPTR)

$DDA7 ; FLOATING STORE REGO (X,Y) <- FRO

$DDAB ;FSTOP = " " (FLPTR) <- FRO

$DDB0 ;FMOVE = $DDB6 ;FR1 <- FRO

PLYARG = $DD40 ;FRO <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR

INPUT: (X,Y) = A(N),A(N-1)...

ACC = # OF COEFFICIENTS = DEGREE+1

FRO = Z

EXP = $DDC0 ;FRO <- E**FRO = EXP(0) FRO * LOG10(E)) CARRY

EXP10 = $DDCC ;FRO <- 10**FRO CARRY

LOG = $DDEC ;FRO <- LN(FRO) = LOG10(FRO)/LOG10(E) CARRY

LOG10 = $DED1 ;FRO <- LOG10 (FRO) CARRY

THE FOLLOWING ARE IN BASIC CARTRIDGE:

SIN = $BD81 ;FRO <- SIN(FRO) DEGFLG=0 =>RADS, 6=DEG. CA

COS = $BD73 ;FRO <- COS(FRO) CARRY

ATAN = $BE43 ;FRO <- ATAN(FRO) CARRY

SQRT = $BE81 ;FRO <- SQUARED (FRO) CARRY
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>604</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>605</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>606</td>
<td>00D4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>607</td>
<td>00DA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>608</td>
<td>00E0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>609</td>
<td>00E6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>610</td>
<td>00EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>611</td>
<td>00ED</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>612</td>
<td>00EE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>613</td>
<td>00EF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>614</td>
<td>00F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>615</td>
<td>00F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>616</td>
<td>00F2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>617</td>
<td>00F3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>618</td>
<td>00F5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>619</td>
<td>00F7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>620</td>
<td>00F9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>621</td>
<td>00FB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>622</td>
<td>00FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>623</td>
<td>00F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>624</td>
<td>00F6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>625</td>
<td>00FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>626</td>
<td>00FE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>627</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>628</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>629</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>630</td>
<td>057E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>631</td>
<td>057F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>632</td>
<td>05B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>633</td>
<td>05E0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>634</td>
<td>05E6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>635</td>
<td>05EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>636</td>
<td>05E6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>637</td>
<td>05EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>638</td>
<td>05FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**COLLEEN MNEMONICS**

<table>
<thead>
<tr>
<th>POKEY</th>
<th>$200</th>
<th>VBLANK ACTION:</th>
<th>DESCRIPTION:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0T0</td>
<td>POKEY+0</td>
<td>P0T0--&gt;PADDL0</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>P0T1</td>
<td>POKEY+1</td>
<td>P0T1--&gt;PADDL1</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>P0T2</td>
<td>POKEY+2</td>
<td>P0T2--&gt;PADDL2</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>P0T3</td>
<td>POKEY+3</td>
<td>P0T3--&gt;PADDL3</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>P0T4</td>
<td>POKEY+4</td>
<td>P0T4--&gt;PADDL4</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>P0T5</td>
<td>POKEY+5</td>
<td>P0T5--&gt;PADDL5</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>P0T6</td>
<td>POKEY+6</td>
<td>P0T6--&gt;PADDL6</td>
<td>0-227 IN RAM CELL</td>
</tr>
<tr>
<td>ADDR</td>
<td>B1 B2 B3 B4</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>D207</td>
<td>POT7 = POKEY+7 ; POT7 --&gt; PADDL7</td>
<td>0-227 IN RAM CELL</td>
<td></td>
</tr>
<tr>
<td>D208</td>
<td>ALLPOT = POKEY+8 ; ???</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D209</td>
<td>KBCODE = POKEY+9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20A</td>
<td>RANDOM = POKEY+10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20B</td>
<td>POT90 = POKEY+11 ; STROBED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20D</td>
<td>SERIN = POKEY+13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20E</td>
<td>IROST = POKEY+14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20F</td>
<td>SKSTAT = POKEY+15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D200</td>
<td>AUDF1 = POKEY+0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D201</td>
<td>AUDC1 = POKEY+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D202</td>
<td>AUDF2 = POKEY+2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D203</td>
<td>AUDC2 = POKEY+3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D204</td>
<td>AUDF3 = POKEY+4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D205</td>
<td>AUDC3 = POKEY+5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D206</td>
<td>AUDF4 = POKEY+6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D207</td>
<td>AUDC4 = POKEY+7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D208</td>
<td>AUDCTL = POKEY+8 ; NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D209</td>
<td>STIMER = POKEY+9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20A</td>
<td>SKRES = POKEY+10 ; NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20D</td>
<td>SEROUT = POKEY+13 ; NONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20E</td>
<td>IREGN = POKEY+14 ; POKMSK --&gt; IREGN (AFFECTED BY OPEN S: OR E: )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D20F</td>
<td>SKCTL = POKEY+15 ; SSKCTL --&gt; SKCTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D000</td>
<td>CTIA = $D000 ; VBLANK ACTION: DESCRIPTION:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D001</td>
<td>HPSP0 = CTIA+0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D002</td>
<td>HPSP1 = CTIA+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D003</td>
<td>HPSP2 = CTIA+2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D004</td>
<td>HPSP3 = CTIA+3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D005</td>
<td>HPSPM0 = CTIA+4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D006</td>
<td>HPSM1 = CTIA+5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D007</td>
<td>HPSM2 = CTIA+6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D008</td>
<td>HPSM3 = CTIA+7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D009</td>
<td>SIZEP0 = CTIA+8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D00A</td>
<td>SIZEP1 = CTIA+9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D00B</td>
<td>SIZEP2 = CTIA+10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D00C</td>
<td>SIZEP3 = CTIA+11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D00D</td>
<td>GRAFTP0 = CTIA+13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D00E</td>
<td>GRAFTP1 = CTIA+14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D00F</td>
<td>GRAFTP2 = CTIA+15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D010</td>
<td>GRAFTP3 = CTIA+16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D011</td>
<td>GRAFTP = CTIA+17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D012</td>
<td>COLPM0 = CTIA+18 ; PCOLRO --&gt; COLPM0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D013</td>
<td>COLPM1 = CTIA+19 ; PCOLROI --&gt; COLPM1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D014</td>
<td>COLPM2 = CTIA+20 ; PCOLR2 --&gt; COLPM2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D015</td>
<td>COLPM3 = CTIA+21 ; PCOLR3 --&gt; COLPM3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D016</td>
<td>COLP0 = CTIA+22 ; COLOR0 --&gt; COLP0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D017</td>
<td>COLP1 = CTIA+23 ; COLOR1 --&gt; COLP1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D018</td>
<td>COLP2 = CTIA+24 ; COLOR2 --&gt; COLP2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D019</td>
<td>COLP3 = CTIA+25 ; COLOR3 --&gt; COLP3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D01A</td>
<td>COLBK = CTIA+26 ; COLOR4 --&gt; COLBK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D01B</td>
<td>PRIOR = CTIA+27 ; (ON OPEN S: OR E: )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D01C</td>
<td>VDELAY = CTIA+28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D01D</td>
<td>GRACATL = CTIA+29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERR</td>
<td>LINE</td>
<td>ADDR</td>
<td>B1</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
<td>----</td>
</tr>
<tr>
<td>712</td>
<td>D01E</td>
<td>HITCLR = CTIA+30</td>
<td></td>
</tr>
<tr>
<td>713</td>
<td>D01F</td>
<td>CONSOL = CTIA+31</td>
<td></td>
</tr>
<tr>
<td>714</td>
<td>D000</td>
<td>MOPF = CTIA+0</td>
<td></td>
</tr>
<tr>
<td>715</td>
<td>D001</td>
<td>M1PF = CTIA+1</td>
<td></td>
</tr>
<tr>
<td>716</td>
<td>D002</td>
<td>M2PF = CTIA+2</td>
<td></td>
</tr>
<tr>
<td>717</td>
<td>D003</td>
<td>M3PF = CTIA+3</td>
<td></td>
</tr>
<tr>
<td>718</td>
<td>D004</td>
<td>POPF = CTIA+4</td>
<td></td>
</tr>
<tr>
<td>719</td>
<td>D005</td>
<td>P1PF = CTIA+5</td>
<td></td>
</tr>
<tr>
<td>720</td>
<td>D006</td>
<td>P2PF = CTIA+6</td>
<td></td>
</tr>
<tr>
<td>721</td>
<td>D007</td>
<td>P3PF = CTIA+7</td>
<td></td>
</tr>
<tr>
<td>722</td>
<td>D008</td>
<td>MOPL = CTIA+8</td>
<td></td>
</tr>
<tr>
<td>723</td>
<td>D009</td>
<td>M1PL = CTIA+9</td>
<td></td>
</tr>
<tr>
<td>724</td>
<td>D00A</td>
<td>M2PL = CTIA+10</td>
<td></td>
</tr>
<tr>
<td>725</td>
<td>D00B</td>
<td>M3PL = CTIA+11</td>
<td></td>
</tr>
<tr>
<td>726</td>
<td>D00C</td>
<td>POPL = CTIA+12</td>
<td></td>
</tr>
<tr>
<td>727</td>
<td>D00D</td>
<td>P1PL = CTIA+13</td>
<td></td>
</tr>
<tr>
<td>728</td>
<td>D00E</td>
<td>P2PL = CTIA+14</td>
<td></td>
</tr>
<tr>
<td>729</td>
<td>D00F</td>
<td>P3PL = CTIA+15</td>
<td></td>
</tr>
<tr>
<td>730</td>
<td>D010</td>
<td>TRIG0 = CTIA+16</td>
<td></td>
</tr>
<tr>
<td>731</td>
<td>D011</td>
<td>TRIG1 = CTIA+17</td>
<td></td>
</tr>
<tr>
<td>732</td>
<td>D012</td>
<td>TRIG2 = CTIA+18</td>
<td></td>
</tr>
<tr>
<td>733</td>
<td>D013</td>
<td>TRIG3 = CTIA+19</td>
<td></td>
</tr>
<tr>
<td>734</td>
<td>D040</td>
<td>ANTIC = #D400</td>
<td></td>
</tr>
<tr>
<td>735</td>
<td>D400</td>
<td>DMACPCTL = ANTIC+0</td>
<td></td>
</tr>
<tr>
<td>736</td>
<td>D401</td>
<td>CHACTL = ANTIC+1</td>
<td></td>
</tr>
<tr>
<td>737</td>
<td>D402</td>
<td>DLISCTL = ANTIC+2</td>
<td></td>
</tr>
<tr>
<td>738</td>
<td>D403</td>
<td>DLISHT = ANTIC+3</td>
<td></td>
</tr>
<tr>
<td>739</td>
<td>D404</td>
<td>HSCROL = ANTIC+4</td>
<td></td>
</tr>
<tr>
<td>740</td>
<td>D405</td>
<td>VSRCRO = ANTIC+5</td>
<td></td>
</tr>
<tr>
<td>741</td>
<td>D407</td>
<td>PBMBASE = ANTIC+7</td>
<td></td>
</tr>
<tr>
<td>742</td>
<td>D409</td>
<td>CHBASE = ANTIC+9</td>
<td></td>
</tr>
<tr>
<td>743</td>
<td>D40A</td>
<td>WSNC = ANTIC+10</td>
<td></td>
</tr>
<tr>
<td>744</td>
<td>D40B</td>
<td>VCOUNT = ANTIC+11</td>
<td></td>
</tr>
<tr>
<td>745</td>
<td>D40C</td>
<td>PENV = ANTIC+12</td>
<td></td>
</tr>
<tr>
<td>746</td>
<td>D40D</td>
<td>PENV = ANTIC+13</td>
<td></td>
</tr>
<tr>
<td>747</td>
<td>D40E</td>
<td>NMIE = ANTIC+14</td>
<td></td>
</tr>
<tr>
<td>748</td>
<td>D40F</td>
<td>NMIE = ANTIC+15</td>
<td></td>
</tr>
<tr>
<td>749</td>
<td>D40F</td>
<td>NMIRES = ANTIC+15</td>
<td></td>
</tr>
<tr>
<td>750</td>
<td>D40F</td>
<td>NMIST = ANTIC+15</td>
<td></td>
</tr>
<tr>
<td>751</td>
<td>D300</td>
<td>PIA = #D300</td>
<td></td>
</tr>
<tr>
<td>752</td>
<td>D300</td>
<td>PORTA = PIA+0</td>
<td></td>
</tr>
<tr>
<td>753</td>
<td>D301</td>
<td>PORTB = PIA+1</td>
<td></td>
</tr>
<tr>
<td>754</td>
<td>D302</td>
<td>PACTL = PIA+2</td>
<td></td>
</tr>
<tr>
<td>755</td>
<td>D303</td>
<td>PBCTL = PIA+3</td>
<td></td>
</tr>
<tr>
<td>756</td>
<td>;</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>757</td>
<td>;</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>758</td>
<td>;</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>759</td>
<td>;</td>
<td>;</td>
<td>PAGE</td>
</tr>
<tr>
<td>ERR LINE</td>
<td>ADDR</td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>760</td>
<td>.PAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>761</td>
<td>LIST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>762</td>
<td>.TITLE</td>
<td>'CENTRAL INPUT/OUTPUT (CIO) 2-7-79'</td>
<td></td>
</tr>
<tr>
<td>763</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>764 0030</td>
<td>ASCZER</td>
<td>'0'</td>
<td></td>
</tr>
<tr>
<td>765 003A</td>
<td>CDOLON</td>
<td>$3A</td>
<td></td>
</tr>
<tr>
<td>766 009B</td>
<td>EOL</td>
<td>$9B</td>
<td></td>
</tr>
</tbody>
</table>
ERROR ROUTINE ADDRESS EQUATE

ERRTNH = ERRTN/256 "MOVED TO LINE 788"
ERRTNL = (-ERRTNH)*256 + ERRTN "MOVED TO LINE 789"

ERROR ROUTINE FOR ILLEGAL PUT
ERRRTN = -1
ERRTNH = ERRTN/256
ERRTNL = (-ERRTNH)*256 + ERRTN

ERROR ROUTINE VECTOR FOR USERS
*CIOV
JMP CIO ; GO TO CIO

CIO INIT JUMP VECTOR FOR POWER UP
*CIOINV
JMP CIOINT ; GO TO INIT

ERROR ROUTINE ADDRESS EQUATE

ERRTNH = ERRTN/256 "MOVED TO LINE 788"
ERRTNL = (-ERRTNH)*256 + ERRTN "MOVED TO LINE 789"

CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)

CIOINIT: LDX #0
CIOII: LDA #IOCFRE ; SET ALL IOC B'S TO FREE
STA ICHID, X ; BY SETTING HANDLER ID'S #FF

LDX 10
LDA IIOCFRE
STA ICHID, X

LDX 10
LDA ICPL, X ; POINT PUT TO ERROR ROUTINE

LDA ICPL, X
ADC IIOCBSZ ; BUMP INDEX BY SIZE

LDX INOTOPN ; IOCB NOT OPEN

LDA ICPTH, X
CMP #MAXIOC ; DONE?

BCC CIOII ; NO

RTS ; YES, RETURN

; ERROR ROUTINE FOR ILLEGAL PUT
ERRTN = -1
ERRTNH = ERRTN/256
ERRTNL = (-ERRTNH)*256 + ERRTN
LDY #NOTOPN ; IOCB NOT OPEN
RTS
ERR LINE ADDR B1 B2 B3 B4
807
808 ;
809 ; CIO LOCAL RAM (USES SPARE BYTES IN ZERO PAGE IOCB)
810 002C
811 ;
812 ; CIO MAIN ROUTINE
813 ;
814 ; CIO INTERFACES BETWEEN USER AND INPUT/OUTPUT DE
815 E4C4 85 2F
816 E4C6 86 2E
817 ;
818 ; CHECK FOR LEGAL IOCB
819 E4CB BA
820 E4C9 29 0F
821 E4CB D0 04
822 E4CD E0 80
823 E4CF 90 05
824 ;
825 ; INVALID IOCB NUMBER -- RETURN ERROR
826 E4D1 A0 B6
827 E4D3 4C 18 E6
828 ;
829 ; MOVE USER IOCB TO ZERO PAGE
830 E4D6 A0 00
831 E4DB BD 40 03
832 E4DB 99 20 00
833 E4DE E8
834 E4DF C8
835 E4E0 C0 0C
836 E4E2 90 F4
837 ;
838 ; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
839 E4E4 A0 84
840 E4E6 A5 22
841 E4EB C9 03
842 E4EA 90 25
843 E4EC A8
844 ;
845 ; MOVE COMMAND TO ZERO BASE FOR INDEX
846 E4ED C0 0E
847 E4EF 90 02
848 E4F1 A0 0E
849 E4F3 B4 17
850 E4F5 B9 C6 E6
851 E4FB F0 0F
852 E4FA C9 02
853 E4FC F0 35
854 E4FE C9 0B
855 E500 B0 4C
856 E502 C9 04
857 E504 F0 63
858 E506 4C C9 E5

CENTRAL INPUT/OUTPUT (CIO) 2-7-79

PAGE 18
ERR LINE ADDR B1 B2 B3 B4 CENTRAL INPUT/OUTPUT (CIO) 2-7-79

859 .PAGE
860 ; OPEN COMMAND
861 ; FIND DEVICE HANDLER IN HANDLER ADDRESS TABLE
862 ; ERROR -- IOC5 ALREADY OPEN
863 E509  A5  20 CIOPEN: LDA  ICHIDZ  ; GET HANDLER ID
864 E508  C9  FF  CMP  #IDCFRE  ; IS THIS IOC5 CLOSED?
865 E50D  F0  05  BEQ  IOC6  ; YES
866 E50E  A5  81 CIERR3: LDY  #PRVOPN  ; ERROR CODE
867 E511  4C  1B  E6  IOC7: .JSR  DEVSRC  ; CALL DEVICE SEARCH
868 E514  20  9E  E6  IOC6: .JSR  COMENT  ; INITIALIZE IOC5 FOR OPEN
869 E517  B0  F8  ; DEVICE FOUND, INITIALIZE IOC5 FOR OPEN
870 E519  20  3D  E6  ; COMPUTE HANDLER ENTRY POINT
871 E51C  B0  F3  ; GO TO HANDLER FOR INITIALIZATION
872 E51E  20  B9  E6  ; STORE PUT BYTE ADDRESS-1 INTO IOC5
873 E521  A9  0B  LDA  #PUTCHR  ; SIMULATE PUT CHARACTER
874 E523  85  17  STA  ICCOMT  ; COMPUTE ENTRY POINT
875 E525  20  3D  E6  STA  ICPTLZ  ; TO PUT BYTE ADDRESS
876 E528  A5  2C  LDA  ICSPRZ  ; MOVE COMPUTED VALUE
877 E52A  B5  26  STA  ICPTHZ  ; RETURN TO USER
878 E52C  A5  2D  STA  ICPTLZ  ; TO PUT BYTE ADDRESS
879 E52E  B5  27  STA  ICPTHZ  ; RETURN TO USER
880 E530  4C  1D  E6  JMP  CIRTN2  ; RETURN TO USER

PAGE 19
ERR LINE ADDR B1 B2 B3 B4 CENTRAL INPUT/OUTPUT (CIO) 2-7-79 PAGE 20

894 .PAGE
895 ;
896 ;
897 ; CLOSE COMMAND
898 E533 A0 01 CICLOS: LDY #SUCCES I ASSUME GOOD CLOSE
899 E535 B4 23 STY ICSTAZ I COMPUTE HANDLER ENTRY POINT
900 E537 20 3D E6 JSR COMENT I GO IF ERROR IN COMPUTE
901 E53A B0 03 BCS CICLO2 I GO TO HANDLER TO CLOSE DEVICE
902 E53C 20 89 E6 JSR GOHAND I GO TO HANDLER TO CLOSE DEVICE
903 E53F A9 FF CICLO2: LDA #IOCFRE I GET IOCB "FREE" VALUE
904 E541 B5 20 STA ICIDIZ I SET HANDLER ID
905 E543 A9 E4 LDA #ERRTML I SET PUT BYTE TO POINT TO ERROR
906 E545 B5 27 STA ICPTLZ
907 E547 A9 C0 LDA #ERRTNL
908 E549 B5 26 STA ICPTLZ
909 E54B 4C 1D E6 JMP CIRTN2 I RETURN
910 ;
911 ;
912 ; STATUS AND SPECIAL REQUESTS
913 ; DO IMPLIED OPEN IF NECESSARY AND GO TO DEVICE
914 E54E A5 20 CISTSP: LDA ICIDIZ I IS THERE A HANDLER ID?
915 E550 C9 FF CMP #IOCFRE I YES
916 E552 D0 05 BNE CIST1
917 ;
918 ; IOCB IS FREE, DO IMPLIED OPEN
919 E554 20 9E E6 JSR DEVSRC I FIND DEVICE IN TABLE
920 E557 B0 B8 BCS CIER4 I GO IF ERROR IN COMPUTE
921 ;
922 ; COMPUTE AND GO TO ENTRY POINT IN HANDLER
923 E559 20 3D E6 CIST1: JSR COMENT I COMPUTE HANDLER ENTRY VECTOR
924 E55C 20 89 E6 JSR GOhAND I GO TO HANDLER
925 ;
926 ; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
927 E55F A6 2E LDX ICIDNO I IOCB INDEX
928 E561 BD 40 03 LDA ICID.X I GET ORIGINAL HANDLER ID
929 E564 B5 20 STA ICIDIZ I RESTORE ZERO PAGE
930 E566 4C 1D E6 JMP CIRTN2 I RETURN
; READ -- DO GET COMMANDS
CIREAD: LDA ICCOMZ ; GET COMMAND BYTE
        AND ICAX1Z ; IS THIS READ LEGAL?
        BNE RCI1A ; YES

; ILLEGAL READ -- IOC0 OPENED FOR WRITE ONLY
        LDY #WRONLY ; ERROR CODE
        RCI1B: JMP CIRTN1 ; RETURN

; COMPUTE AND CHECK ENTRY POINT
RCI1A: JSR COMENT ; COMPUTE ENTRY POINT
        BCS RCI1B ; GO IF ERROR IN COMPUTE

; GET RECORD OR CHARACTERS
LDA ICBLIZ
        ORA ICBLIZ+1 ; IS BUFFER LENGTH ZERO?
        BNE RCI3 ; NO

; LOOP TO FILL BUFFER OR END RECORD
RCI3: JSR GOHAND ; GO TO HANDLER TO GET BYTE
        STA CIOCHR ; SAVE BYTE
        BMI RCI4 ; END TRANSFER IF ERROR
        LDY #0
        STA (ICBALZ),Y ; PUT BYTE IN USER BUFFER
        JSR INCBFP ; INCREMENT BUFFER POINTER
        LDA ICCOMZ ; GET COMMAND CODE
        STA (ICBALZ),Y ; PUT BYTE IN USER BUFFER
        .JSR DECBFL ; YES, DECREMENT BUFFER LENGTH
        JMP RCI4 ; END TRANSFER

; CHECK FOR EOL ON TEXT RECORDS
        LDA CIOCHR ; GET BYTE
        CMP #EOL ; IS IT AN EOL?
        BNE RCI1 ; NO

; CHECK BUFFER FULL
RCI1: JSR DECBFL ; DECREMENT BUFFER LENGTH
        BNE RCI3 ; CONTINUE IF NON ZERO
; BUFFER FULL, RECORD NOT ENDED
; DISCARD BYTES UNTIL END OF RECORD
979 E5AC A5 22 RC12: LDA ICCOMZ ;GET COMMAND BYTE
980 E5AE 29 02 AND #2 ;IS IT GET CHARACTER?
981 E5B0 D0 11 BNE RC14 ;YES, END TRANSFER
982
983 ; LOOP TO WAIT FOR EOL
984 E5B2 20 89 E6 RC16: JSR G0HAND ;GET BYTE FROM HANDLER
985 E5B5 B5 2F STA CIACHR ;SAVE CHARACTER
986 E5B7 30 0A BMI RC14 ;GO IF ERROR
987
988 ; TEXT RECORD, WAIT FOR EOL
989 E5B9 A5 2F LDA CIACHR ;GET G0T BYTE
990 E5BB C9 98 CMP #EOL ;IS IT EOL?
991 E5BD D0 F3 BNE RC16 ;NO, CONTINUE
992
993 ; END OF RECORD, BUFFER FULL -- SEND TRUNCATED RECORD MESSAGE
994 E5BF A9 89 RC111: LDA #TRNRCD ;ERROR CODE
995 E5C1 B5 23 STA ICSTAZ ;STORE IN IOCB
996
997 ; TRANSFER DONE
998 E5C3 20 77 E6 RC14: JSR SUBBFL ;SET FINAL BUFFER LENGTH
999 E5C6 4C 1D E6 JMP CIRTN2 ;RETURN
1000 ; WRITE -- DO PUT COMMANDS
1001 CIWRIT: LDA ICCOMZ ;GET COMMAND BYTE
1002 AND ICAX12 ;IS THIS WRITE LEGAL?
1003 BNE WCI1A ;YES
1004 ; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
1005 LDY #RDONLY ;ERROR CODE
1006 WCI1B: JMP CIRTN1 ;RETURN
1007 ; COMPUTE AND CHECK ENTRY POINT
1008 WCI1A: JSR COMENT ;COMPUTE HANDLER ENTRY POINT
1009 BCS WCI1B ;GO IF ERROR IN COMPUTE
1010 ; PUT RECORD OR CHARACTERS
1011 WCI3: LOY #0
1012 ;LOOP TO TRANSFER BYTES FROM BUFFER TO HANDLER
1013 LDA ICBLZ ; GET BYTE FROM BUFFER
1014 STA CIOCHR ;SAVE
1015 WCI4: JSR GOHAND ;GO PUT BYTE
1016 BMI WC15 ;END IF ERROR
1017 JSR INCBFp ;INCREMENT BUFFER POINTER
1018 ; CHECK FOR TEXT RECORD
1019 LDA ICCOMZ ;GET COMMAND BYTE
1020 AND #2 ;IS IT PUT RECORD?
1021 BNE WCI1 ;NO
1022 ; TEXT RECORD -- CHECK FOR EOL TRANSFER
1023 LDA CIOCHR ;GET LAST CHARACTER
1024 CMP #EOL ;IS IT AN EOL?
1025 BNE WC15 ;NO
1026 JSR DECBFp ;DECREMENT BUFFER LENGTH
1027 JMP WC15 ;END TRANSFER
1028 ; CHECK FOR BUFFER EMPTY
1029 JSR DECBFp ;DECREMENT BUFFER LENGTH
1030 BNE WCI3 ;CONTINUE IF NON ZERO
1046 .PAGE
1047 ; BUFFER EMPTY, RECORD NOT FILLED
1048 ; CHECK TYPE OF TRANSFER
1049 1050 E60A A5 22 WC12: LDA ICCOMZ ; GET COMMAND CODE
1051 E60C 29 02 AND #2 ; IS IT PUT CHARACTER?
1052 E60E D0 03 SNE WC15 ; YES, END TRANSFER
1053
1054 ; PUT RECORD (TEXT), BUFFER EMPTY, SEND EOL
1055 E610 A9 9B LDA #EOL
1056 E612 20 89 E6 JSR GOHAND ; GO TO HANDLER
1057
1058 ; END PUT TRANSFER
1059 E615 20 77 E6 WC15: JSR SUBBFL ; SET ACTUAL PUT BUFFER LENGTH
1060 E618 4C 1D E6 JMP CIRTN2 ; RETURN
CIO RETURNS

CIRTN1: STY ICSTAZ ; SAVE STATUS

CIRTN2: LDY ICIDNO ; GET IOCB INDEX

CIRTN3: LOA IOCBAS,X ; ZERO PAGE

CIRTN4: LDA CIOCHR ; GET LAST CHARACTER

CENTRAL INPUT/OUTPUT (CIO) 2-7-79

PAGE 25
; CIO SUBROUTINES
; COMENT -- CHECK AND COMPUTE HANDLER ENTRY POINT
; ILLEGAL HANDLER INDEX MEANS DEVICE NOT OPEN FOR OPERATION
LDY #ICBLL+1, Y ; GET LOW BYTE OF VECTOR FROM HANDLER ITSELF AND SAVE
LDA (ICSPRZ), Y ; GET HI BYTE OF VECTOR
STA ICSPRZ+1 ; SET LO BYTE
LDA COMTAB-3, Y ; GET COMMAND OFFSET
STA ICSPRZ+2 ; AND SAVE IN POINTER
LDY ICCOMT ; GET COMMAND CODE
STA ICSPRZ+3 ; GET COMMAND CODE
LDA HATABS+2, Y ; GET HI BYTE OF ADDRESS
STA ICSPRZ ; AND SAVE IN POINTER
LDY ICHIDZ ; GET HANDLER INDEX
CPY #MAXDEV+1 ; IS IT A LEGAL INDEX?
BCC COM1 ; YES
; USE HANDLER ADDRESS TABLE AND COMMAND TABLE TO GET VECTOR
; COM1: LDA HATABS+1, Y ; GET LOW BYTE OF ADDRESS
STA ICSPRZ+4 ; AND SAVE IN POINTER
LDY ICHIDZ ; GET HANDLER INDEX
CPY #MAXDEV+1 ; IS IT A LEGAL INDEX?
BCC COM1 ; YES
; DECBFL -- DECREMENT BUFFER LENGTH DOUBLE BYTE
; Z FLAG = 0 ON RETURN IF LENGTH = 0 AFTER DECREMENT
DEC ICBLIZ ; DECREMENT LOW BYTE
LDA ICBLIZ ; CHECK IT
LDA ICBLIZ+1 ; DID IT GO BELOW?
BNE DECBF1 ; NO
DEC ICBLIZ+1 ; DECREMENT HI BYTE
ORA ICBLIZ+1 ; SET Z IF BOTH ARE ZERO
RTS
; INCBFP -- INCREMENT WORKING BUFFER POINTER
INC ICBLIZ ; BUMP LOW BYTE
BNE INCBF1 ; GO IF NOT ZERO
INC ICBLIZ+1 ; ELSE, BUMP HI BYTE
RTS
; SUBBFL -- SET BUFFER LENGTH = BUFFER LENGTH - WORKING BYTE COUNT
LDA ICBLIZ+1, X ; GET LOW BYTE OF INITIAL LENGTH
LDX ICHIDZ ; GET HANDLER INDEX
SEC
LDA ICHIDZ+1, Y ; GET HIGH BYTE
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR B1 B2 B3 B4</th>
<th>CENTRAL INPUT/OUTPUT (CIO) 2-7-79</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1141</td>
<td>E67D E5 28</td>
<td>SBC ICBLZ : SUBTRACT FINAL LOW BYTE</td>
<td></td>
</tr>
<tr>
<td>1142</td>
<td>E67F 85 28</td>
<td>STA ICBLZ : AND SAVE BACK</td>
<td></td>
</tr>
<tr>
<td>1143</td>
<td>E681 BD 49 03</td>
<td>LDA ICBLHZ.X : GET HI BYTE</td>
<td></td>
</tr>
<tr>
<td>1144</td>
<td>E684 E5 27</td>
<td>SBC ICBLZ+1</td>
<td></td>
</tr>
<tr>
<td>1145</td>
<td>E686 85 29</td>
<td>STA ICBLHZ</td>
<td></td>
</tr>
<tr>
<td>1146</td>
<td>E688 60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>1147</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1148</td>
<td></td>
<td>AQHAN D -- GO INDIRECT TO A DEVICE HANDLER</td>
<td></td>
</tr>
<tr>
<td>1149</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1150</td>
<td></td>
<td>; Y= STATUS ON RETURN. N FLAG=1 IF ERROR ON RETURN</td>
<td></td>
</tr>
<tr>
<td>1151</td>
<td>E689 A0 92</td>
<td>AQHAN D: LDY #FNCNOT ; PREPARE NO FUNCTION STATUS FOR HANDLER RTS</td>
<td></td>
</tr>
<tr>
<td>1152</td>
<td>E688 20 93 E6</td>
<td>JSR CIJUMP ; USE THE INDIRECT JUMP</td>
<td></td>
</tr>
<tr>
<td>1153</td>
<td>E68E 84 23</td>
<td>STY ICSTAZ ; SAVE STATUS</td>
<td></td>
</tr>
<tr>
<td>1154</td>
<td>E690 C0 00</td>
<td>CPY #0 ; AND SET N FLAG</td>
<td></td>
</tr>
<tr>
<td>1155</td>
<td>E692 60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>1156</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1157</td>
<td></td>
<td>AQHAN D: GO HAND间接 TO HANDLER BY PAUL'S METHOD</td>
<td></td>
</tr>
<tr>
<td>1158</td>
<td>E693 AA</td>
<td>CIJUMP: TAX ; SAVE A</td>
<td></td>
</tr>
<tr>
<td>1159</td>
<td>E694 A5 2D</td>
<td>LDA ICSPRZ+1 ; GET JUMP ADDRESS HI BYTE</td>
<td></td>
</tr>
<tr>
<td>1160</td>
<td>E696 48</td>
<td>PHA ; PUT ON STACK</td>
<td></td>
</tr>
<tr>
<td>1161</td>
<td>E697 A5 2C</td>
<td>LDA ICSPRZ ; GET JUMP ADDRESS LO BYTE</td>
<td></td>
</tr>
<tr>
<td>1162</td>
<td>E699 48</td>
<td>PHA ; PUT ON STACK</td>
<td></td>
</tr>
<tr>
<td>1163</td>
<td>E69A 8A</td>
<td>TXA ; RESTORE A</td>
<td></td>
</tr>
<tr>
<td>1164</td>
<td>E69B A6 2E</td>
<td>LDX ICIDNO ; GET IOCHAR INDEX</td>
<td></td>
</tr>
<tr>
<td>1165</td>
<td>E69D 60</td>
<td>RTS ; GO TO HANDLER INDIRECTLY</td>
<td></td>
</tr>
</tbody>
</table>
; DEVSRC -- DEVICE SEARCH, FIND DEVICE IN HANDLER ADDRESS TABLE
; LOOP TO FIND DEVICE
DEVSRC: LDY #0

LDY (ICBALZ),Y ; GET DEVICE NAME FROM USER
BEQ CIERR2 ; INITIAL COMPARE INDEX

CMP HLABS,Y ; IS THIS THE DEVICE?
BEQ DEVS2 ; YES

DEY ; ELSE, POINT TO NEXT DEVICE NAME

BPL DEVS1 ; CONTINUE FOR ALL DEVICES

; NO DEVICE FOUND, DECLARE NON-EXISTENT DEVICE ERROR
CIERR2: LDY #NONDEV ; ERROR CODE
SEC ; SHOW ERROR
BCS DEVS4 ; AND RETURN

; FOUND DEVICE, SET ICHID, ICDNO, AND INIT DEVICE
DEVS2: TYA

STA ICHIDZ ; SAVE HANDLER INDEX

LDY #1

LDA (ICBALZ),Y ; GET DEVICE NUMBER (DRIVE NUMBER)

SBC #$A ; SUBTRACT ASCII ZERO

CMP #$A ; IS NUMBER IN RANGE?

BCC DEVS3 ; YES

LDA #1 ; NO, DEFAULT TO ONE

DEVS3: STA ICDNOZ ; SAVE DEVICE NUMBER

CLC ; SHOW NO ERROR

; RETURN

RTS
; PAGE
; CIO ROM TABLES
; COMMAND TABLE
; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER

COMTAB: .BYTE 0,4,4,4,4,4,6,6,6,6,2,8,10

LENGTH =--CIOINT
CRNTP1 =\n
CIOSPR: .BYTE INTORG-CRNTP1 ;^QCIOL IS TOO LONG
.TITLE 'interrupt handler'

lives on dki: intvh.src

srtim2 = 6 ; second repeat interval

this is to make dos 2 work which used an absolute address

**=e912

1224 E912 4C ED E8

JMP setvbl

**=setvbl

1226 E45C 4C ED E8

JMP setvbl

1227 E45F 4C AE E7

JMP sysvbl

1228 E462 4C 05 E9

JMP xitvbl

1229 *intinv

1230 E468 4C D5 E6

JMP ihinit

1231

**=vctabl+intabs-vdslst

1234 E480 90 E7

.word syrti ; vdslst

1235 E482 8F E7

.word syir08 ; vprecd

1236 E484 8F E7

.word syir08 ; vinter

1237 E486 8F E7

.word syir08 ; vbreak

1238

1239 E488

.res 8

1240 E490 8F E7

.word syir08 ; vtimr1

1241 E492 8F E7

.word syir08 ; vtimr2

1242 E494 8F E7

.word syir08 ; vtimr4

1243 E496 06 E7

.word syir08 ; vimirq

1244 E498 00 00 00 00

.word 0,0,0,0,0 ; cdtmv1-4

1245 E49C 00 00 00 00

1246 E4A0 00 00

1247 E4A2 AE E7

.word sysvbl ; vvblki

1248 E4A4 05 E9

.word xitvbl ; vvblkd

1249

1250 *900C

1251

1252 900C A9 E6

lda #pirgh ; set up ram vectors for linbug version

1253 900E 8D F9 FF

sta $fff9

1254 9011 A9 F3

lda #pirgl

1255 9013 8D F8 FF

sta $fff8

1256 9016 A9 E7

lda $pnmih

1257 9018 8D F8 FF

sta $fff8

1258 9018 A9 91

lda $pnm1

1259 901D 8D FA FF

sta $fffa

1260 9020 60

rts
1261  **INTORG**
1262  **PAGE**
1263  ; IRG HANDLER
1264  ; JUMP THRU IMMEDIATE IRG VECTOR, WHICH ORDINARILY POINTS TO
1265  ; SYSTEM IRG: DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
1266  ;
1267  ;
1268  **INTORG**
1269  E6D5 A9 40  IHINIT: LDA **#40** ;VLB ON BUF DLST OFF***FOR NOW***
1270  E6D7 8D 0E D4  STA NMIEN ;ENABLE DISPLAY LIST, VERTICAL BLANK
1271  E6DA A9 38  LDA **#38** ;LOOK AT DATA DIRECTION REGISTERS IN PIA
1272  E6DC 8D 02 D3  STA PACTL
1273  E6DF 8D 03 D3  STA PBCTL
1274  E6E2 A9 00  LDA **#0** ;MAKE ALL INPUTS
1275  E6E4 8D 00 D3  STA PORTA
1276  E6E7 8D 01 D3  STA PORTB
1277  E6EA A9 3C  LDA **#3C** ;BACK TO PORTS
1278  E6EF 8D 03 D3  STA PBCTL
1279  E6F2 60  RTS
1280  E6F3 6C 16 02  PIRG: JMP (VIMIRG)
1281  E6F6 80  CMPTAB: .BYTE **$80** ;BREAK KEY
1282  E6F7 40  .BYTE **$40** ;KEY STROKE
1283  E6F8 04  .BYTE **$04** ;TIMER 4
1284  E6F9 02  .BYTE **$02** ;TIMER 2
1285  E6FA 01  .BYTE **$01** ;TIMER 1
1286  E6FB 08  .BYTE **$08** ;SERIAL OUT COMPLETE
1287  E6FC 10  .BYTE **$10** ;SERIAL OUT READY
1288  E6FD 20  .BYTE **$20** ;SERIAL IN READY
1289  E6FE 12  ; THIS IS A TABLE OF OFFSETS INTO PAGE 2. THEY POINT TO
1290  E6FF 36  ADRTAB: .BYTE BKKY-INTABS
1291  E700 14  .BYTE VKEYBD-INTABS
1292  E701 12  .BYTE VIMR4-INTABS
1293  E702 10  .BYTE VIMR2-INTABS
1294  E703 0E  .BYTE VIMR1-INTABS
1295  E704 0C  .BYTE VSEROC-INTABS
1296  E705 0A  .BYTE VSEROR-INTABS
1297  E706 48  ;SYIRG: PHA ;SAVE ACCUMULATOR
1298  E707 AD 0E D2  LDA IRGST ;CHECK FOR SERIAL IN
1299  E708 29 20  AND **#20**
1300  E709 D0 0D  BNE SYIRG2
1301  E70A A9 DF  LDA **#DF** ;MASK ALL OTHERS
1302  E70B 8D 0E D2  STA IRGEN
1303  E70C 8D 0E D2  LDA POKMSK
1304  E70D 8D 0E D2  STA IRGEN
1305  E70E 6C 0A 02  JMP (VSERIN)
1306  E70F 48  ;SYIRG2: TXA ;PUT X INTO ACC
1307  E710 AD 0E D2  PHA E71C 48  ;SAVE X ONTO STACK
1308  E711 A2 06  LDX **#6** ;START WITH SIX OFFSET
1309  E712 8D F6 0E  LOOPM: LDA CMPTAB,X ;LOAD MASK
1310  E713 0A 06  CPX **#5** ;CHECK TO SEE IF COMPLETE IS SET
1311  E714 00 05
ERR LINE ADDR B1 B2 B3 B4 INTERRUPT HANDLER

1315 E724 DO 04 BNE LOOPM2
1316 E726 25 10 AND POKMSK ; IS THIS INTERRUPT ENABLED?
1317 E728 F0 05 BEG LL
1318 E72A 2C 0E D2 LOOPM2: BIT IRGST ; IS IT THE INTERRUPT?
1319 E72D F0 06 BEG
1320 E72F CA LL: DEX ; NO DEC X AND TRY NEXT MASK
1321 E730 10 ED BNE LOOPM
1322 E732 4C 62 E7 JMP SYIRG8 ; DONE BUT NO INTERRUPT
1323 E735 49 FF JMPR: EOR #$FF ; COMPLEMENT MASK
1324 E737 8D 0E D2 STA IRGEN ; ENABLE ALL OTHERS
1325 E739 A5 10 LDA POKMSK ; GET POKE MASK
1326 E73C BD 0E D2 STA IRGEN ; ENABLE THOSE IN POKE MASK
1327 E73F BD FE E6 LDA POKMSK ; GET POKE MASK
1328 E742 AA TAX
1329 E743 BD 00 02 LDA INTABS,X ; GET ADDRESS LOW PART
1330 E746 BD 8C 02 STA JVECK ; PUT IN VECTOR
1331 E749 BD 01 02 LDA INTABS+1,X ; GET ADDRESS HIGH PART
1332 E74C BD 8D 02 STA JVECK+1 ; PUT IN VECTOR HIGH PART
1333 E74F 68 PLA ; PULL X REGISTER FROM STACK
1334 E750 AA TAX ; PUT IT INTO X
1335 E751 6C 8C 02 JMP (JVECK) ; JUMP TO THE PROPER ROUTINE
1336 E754 00 00 LDA #0 ; BREAK KEY ROUTINE
1337 E756 85 11 STA BRKKEY ; SET BREAK KEY FLAG
1338 E758 BD FF 02 STA SSFLAG ; START/STOP FLAG
1339 E75B BD F0 02 STA CRSINH ; CURSOR INHIBIT
1340 E75E BD 40 PLA ; TURN OFF ATRACT MODE
1341 E760 68 PL A ; EXIT FROM INT
1342 E761 40 RTI
1343 E762 68 SYIRG8: PLA
1344 E763 AA TAX
1345 E764 2C 02 D3 BIT PACTL ; PROCEED ***I GUESS***
1346 E767 10 06 BPL SYIRG9
1347 E769 AD 00 D3 LDA PORTA ; CLEAR INT STATUS BIT
1348 E76C 6C 02 02 JMP (VPRCED)
1349 E76F 2C 03 D3 SYIRG9: BIT PBCTL ; INTERRUPT ***I GUESS***
1350 E772 10 06 BPL SYIRG9
1351 E774 AD 01 D3 LDA PORTB ; CLEAR INT STATUS
1352 E777 6C 04 02 JMP (VINTER)
1353 E77A 68 PLA
1354 E77B BD 8C 02 SYIRG: STA JVECK
1355 E77E 68 PLA
1356 E77F 4B PH A
1357 E780 29 10 AND #$10 ; B BIT OF P REGISTER
1358 E782 F0 07 BEG SYRTI2
1359 E784 AD 8C 02 LDA JVECK
1360 E787 48 PH A
1361 E78B 6C 06 02 JMP (VBREAK)
1362 E78B AD 8C 02 SYRTI2: LDA JVECK
1363 E78E 4B PH A
1364 E790 68 SYIRG8: PLA
1365 E790 40 SYRTI: RTI ; UNIDENTIFIED INTERRUPT, JUST RETURN.
ERR LINE ADDR B1 B2 B3 B4 INTERRUPT HANDLER

1366 .PAGE
1367 ;
1368 ; NMI HANDLER
1369 ;
1370 ; DETERMINE CAUSE AND JUMP THRU VECTOR
1371 ;
1372 E791 2C OF D4 PNMI: BIT NMIST
1373 E794 10 03 BPL PNMI1 ; SEE IF DISPLAY LIST
1374 E796 6C 00 02 JMP (VDSLST)
1375 E799 48 PNMI1: PHA
1376 E79A AD OF D4 LDA NMIST
1377 E79D 29 20 AND #$20 ; SEE IF RESET
1378 E79F F0 03 BEG **+5
1379 E7A1 4C 74 E4 JMP WARMSV ; GO THRU WARM START JUMP
1380 E7A4 8A TXA ; SAVE REGISTERS
1381 E7A5 48 PHA
1382 E7A6 98 TYA
1383 E7A7 48 PHA
1384 E7AB 8D OF D4 STA NMRES ; RESET INTERRUPT STATUS
1385 E7AB 6C 22 02 JMP (VVBLKI) ; JUMP THRU VECTOR
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>INTERRUPT HANDLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1386</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1387</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1388</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1389</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1390</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1391</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1392</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1393</td>
<td>E7AE</td>
<td>E6</td>
<td>14</td>
<td></td>
<td></td>
<td>ERR LINE ADDR B1 B2 B3 B4</td>
</tr>
<tr>
<td>1394</td>
<td>E7BD</td>
<td>D0</td>
<td>08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1395</td>
<td>E7B2</td>
<td>E6</td>
<td>4D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1396</td>
<td>E7B4</td>
<td>E6</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1397</td>
<td>E7B6</td>
<td>D0</td>
<td>02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1398</td>
<td>E7BB</td>
<td>E6</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1399</td>
<td>E7BA</td>
<td>A9</td>
<td>FE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td>E7BC</td>
<td>D0</td>
<td>08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1401</td>
<td>E7BE</td>
<td>A4</td>
<td>4D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1402</td>
<td>E7C0</td>
<td>10</td>
<td>06</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1403</td>
<td>E7C2</td>
<td>85</td>
<td>4D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1404</td>
<td>E7C4</td>
<td>A6</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1405</td>
<td>E7C6</td>
<td>A9</td>
<td>F6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1406</td>
<td>E7CB</td>
<td>85</td>
<td>4E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1407</td>
<td>E7CA</td>
<td>B6</td>
<td>4F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1408</td>
<td>E7CC</td>
<td>A2</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1409</td>
<td>E7CE</td>
<td>20</td>
<td>D0</td>
<td>E8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1410</td>
<td>E7D1</td>
<td>D0</td>
<td>03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1411</td>
<td>E7D3</td>
<td>20</td>
<td>CA</td>
<td>E8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1412</td>
<td>E7D6</td>
<td>A5</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1413</td>
<td>E7DB</td>
<td>D0</td>
<td>08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1414</td>
<td>E7DA</td>
<td>BA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1415</td>
<td>E7DB</td>
<td>BD</td>
<td>04</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1416</td>
<td>E7DE</td>
<td>29</td>
<td>04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1417</td>
<td>E7E0</td>
<td>F0</td>
<td>03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1418</td>
<td>E7E2</td>
<td>4C</td>
<td>05</td>
<td>E9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1419</td>
<td>E7E5</td>
<td>AD</td>
<td>0D</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1420</td>
<td>E7EB</td>
<td>BD</td>
<td>35</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1421</td>
<td>E7EB</td>
<td>AD</td>
<td>0C</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1422</td>
<td>E7EE</td>
<td>BD</td>
<td>34</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1423</td>
<td>E7F1</td>
<td>AD</td>
<td>31</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1424</td>
<td>E7F4</td>
<td>BD</td>
<td>03</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1425</td>
<td>E7F7</td>
<td>AD</td>
<td>30</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1426</td>
<td>E7FA</td>
<td>BD</td>
<td>02</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1427</td>
<td>E7FD</td>
<td>AD</td>
<td>2F</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1428</td>
<td>E800</td>
<td>BD</td>
<td>00</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1429</td>
<td>E803</td>
<td>AD</td>
<td>6F</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1430</td>
<td>E806</td>
<td>BD</td>
<td>18</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1431</td>
<td>E809</td>
<td>A2</td>
<td>08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1432</td>
<td>E80B</td>
<td>BE</td>
<td>1F</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1433</td>
<td>E80E</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1434</td>
<td>E80F</td>
<td>BD</td>
<td>C0</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1435</td>
<td>E812</td>
<td>43</td>
<td>4F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1436</td>
<td>E814</td>
<td>25</td>
<td>4E</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1437</td>
<td>E816</td>
<td>9D</td>
<td>12</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1438</td>
<td>E819</td>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1439</td>
<td>E81A</td>
<td>10</td>
<td>F2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>INTERRUPT HANDLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1386</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1387</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1388</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1389</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1390</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1391</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1392</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1393</td>
<td>E7AE</td>
<td>E6</td>
<td>14</td>
<td>ERR LINE ADDR B1 B2 B3 B4</td>
</tr>
<tr>
<td>1394</td>
<td>E7BD</td>
<td>D0</td>
<td>08</td>
<td></td>
</tr>
<tr>
<td>1395</td>
<td>E7B2</td>
<td>E6</td>
<td>4D</td>
<td></td>
</tr>
<tr>
<td>1396</td>
<td>E7B4</td>
<td>E6</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1397</td>
<td>E7B6</td>
<td>D0</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>1398</td>
<td>E7BB</td>
<td>E6</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>1399</td>
<td>E7BA</td>
<td>A9</td>
<td>FE</td>
<td></td>
</tr>
<tr>
<td>1400</td>
<td>E7BC</td>
<td>D0</td>
<td>08</td>
<td></td>
</tr>
<tr>
<td>1401</td>
<td>E7BE</td>
<td>A4</td>
<td>4D</td>
<td></td>
</tr>
<tr>
<td>1402</td>
<td>E7C0</td>
<td>10</td>
<td>06</td>
<td></td>
</tr>
<tr>
<td>1403</td>
<td>E7C2</td>
<td>85</td>
<td>4D</td>
<td></td>
</tr>
<tr>
<td>1404</td>
<td>E7C4</td>
<td>A6</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>1405</td>
<td>E7C6</td>
<td>A9</td>
<td>F6</td>
<td></td>
</tr>
<tr>
<td>1406</td>
<td>E7CB</td>
<td>85</td>
<td>4E</td>
<td></td>
</tr>
<tr>
<td>1407</td>
<td>E7CA</td>
<td>B6</td>
<td>4F</td>
<td></td>
</tr>
<tr>
<td>1408</td>
<td>E7CC</td>
<td>A2</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>1409</td>
<td>E7CE</td>
<td>20</td>
<td>D0</td>
<td>E8</td>
</tr>
<tr>
<td>1410</td>
<td>E7D1</td>
<td>D0</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>1411</td>
<td>E7D3</td>
<td>20</td>
<td>CA</td>
<td>E8</td>
</tr>
<tr>
<td>1412</td>
<td>E7D6</td>
<td>A5</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>1413</td>
<td>E7DB</td>
<td>D0</td>
<td>08</td>
<td></td>
</tr>
<tr>
<td>1414</td>
<td>E7DA</td>
<td>BA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1415</td>
<td>E7DB</td>
<td>BD</td>
<td>04</td>
<td>01</td>
</tr>
<tr>
<td>1416</td>
<td>E7DE</td>
<td>29</td>
<td>04</td>
<td></td>
</tr>
<tr>
<td>1417</td>
<td>E7E0</td>
<td>F0</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>1418</td>
<td>E7E2</td>
<td>4C</td>
<td>05</td>
<td>E9</td>
</tr>
<tr>
<td>1419</td>
<td>E7E5</td>
<td>AD</td>
<td>0D</td>
<td>D4</td>
</tr>
<tr>
<td>1420</td>
<td>E7EB</td>
<td>BD</td>
<td>35</td>
<td>02</td>
</tr>
<tr>
<td>1421</td>
<td>E7EB</td>
<td>AD</td>
<td>0C</td>
<td>D4</td>
</tr>
<tr>
<td>1422</td>
<td>E7EE</td>
<td>BD</td>
<td>34</td>
<td>02</td>
</tr>
<tr>
<td>1423</td>
<td>E7F1</td>
<td>AD</td>
<td>31</td>
<td>02</td>
</tr>
<tr>
<td>1424</td>
<td>E7F4</td>
<td>BD</td>
<td>03</td>
<td>D4</td>
</tr>
<tr>
<td>1425</td>
<td>E7F7</td>
<td>AD</td>
<td>30</td>
<td>02</td>
</tr>
<tr>
<td>1426</td>
<td>E7FA</td>
<td>BD</td>
<td>02</td>
<td>D4</td>
</tr>
<tr>
<td>1427</td>
<td>E7FD</td>
<td>AD</td>
<td>2F</td>
<td>02</td>
</tr>
<tr>
<td>1428</td>
<td>E800</td>
<td>BD</td>
<td>00</td>
<td>D4</td>
</tr>
<tr>
<td>1429</td>
<td>E803</td>
<td>AD</td>
<td>6F</td>
<td>02</td>
</tr>
<tr>
<td>1430</td>
<td>E806</td>
<td>BD</td>
<td>18</td>
<td>D0</td>
</tr>
<tr>
<td>1431</td>
<td>E809</td>
<td>A2</td>
<td>08</td>
<td></td>
</tr>
<tr>
<td>1432</td>
<td>E80B</td>
<td>BE</td>
<td>1F</td>
<td>D0</td>
</tr>
<tr>
<td>1433</td>
<td>E80E</td>
<td>58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1434</td>
<td>E80F</td>
<td>BD</td>
<td>C0</td>
<td>02</td>
</tr>
<tr>
<td>1435</td>
<td>E812</td>
<td>43</td>
<td>4F</td>
<td></td>
</tr>
<tr>
<td>1436</td>
<td>E814</td>
<td>25</td>
<td>4E</td>
<td></td>
</tr>
<tr>
<td>1437</td>
<td>E816</td>
<td>9D</td>
<td>12</td>
<td>D0</td>
</tr>
<tr>
<td>1438</td>
<td>E819</td>
<td>CA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1439</td>
<td>E81A</td>
<td>10</td>
<td>F2</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>1386</th>
<th>. PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1387</td>
<td>; SYSTEM VBLANK ROUTINE</td>
</tr>
<tr>
<td>1388</td>
<td>; INC FRAME COUNTER. PROCESS COUNTDOWN Timers. Exit if I WAS SET, CLEAR</td>
</tr>
<tr>
<td>1389</td>
<td>; SET DLSTL, DLSTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.</td>
</tr>
<tr>
<td>1390</td>
<td>; INC FRAME COUNTER</td>
</tr>
</tbody>
</table>

**ERR LINE ADDR B1 B2 B3 B4 INTERRUPT HANDLER PAGE 34**
ERR LINE ADDR B1 B2 B3 B4 INTERRUPT HANDLER

1440 E81C AD F4 02 LDA CHBAS
1441 E81F BD 09 D4 STA CHBASE
1442 E822 AD F3 02 LDA CHACT
1443 E825 BD 01 D4 STA CHACTL
1444 E828 A2 02 LDX #2 ;POINT TO TIMER 2
1445 E82A 20 D0 00 JSR DCTIMR ;IF DIDNT GO ZERO
1446 E82D DO 03 BNE SYSVB4 ;GO JUMP TO TIMER2 ROUTINE
1448 E832 A2 02 SYSVB4: LDX #2 ;RESTORE X
1449 E834 EB SYSVBB: INX
1450 E835 EB INX
1451 E836 BD 18 02 LDA CDTMV1,X
1452 E839 1D 19 02 ORA CDTMV1+1,X
1453 E83C FO 06 BEG SYSVB6A ;IF KEY DOWN
1454 E83E 20 D0 00 JSR DCTIMR ;DECREMENT AND SET FLAG IF NONZERO
1455 E841 9D 26 02 STA CDTMV3-4,X
1456 E844 E0 08 SYSVB6A: CPX #8 iSEE IF DONE ALL 3
1457 E846 DO EC BNE SYSVB6A iLOOP
1458 E848 AD F1 02 LDA SKSTAT ;CHECK DEBOUNCE COUNTER
1459 E84B 29 04 AND #$04 ;KEY DOWN BIT
1460 E84D F0 08 BEG SYSVB6A ;KEY UP SO COUNT IT
1461 E84F AD F1 02 LDA KEYDEL ;KEY DELAY COUNTER
1462 E852 FO 03 BEQ SYSVB6A ;IF COUNTED DOWN ALREADY
1463 E854 CE F1 02 DEC KEYDEL ;COUNT IT
1464 E857 AD 2B 02 SYSVB7: LDA SRTIMR ;CHECK SOFTWARE REPEAT TIMER
1465 E85A F0 17 BEQ SYSVB7 ;DOESN'T COUNT
1466 E85C AD OF D2 LDA SKSTAT
1467 E85F 29 04 AND #$04 ;CHECK KEY DOWN BIT
1468 E861 D0 60 BNE SYSVB6 ;BRANCH IF NO LONGER DOWN
1469 E863 CE 2B 02 DEC SRTIMR ;COUNT FRAME OF KEY DOWN
1470 E866 DO 0B BNE SYSVB7 ;BRANCH IF NOT RUN OUT
1471 E869 A9 06 ;TIMER RUN OUT - RESET AND SIMULATE KEYBOARD IRG
1472 E86A BD 2B 02 LDA #SRTIM2 ;TIMER VALUE
1473 E86B AD 09 D2 STA SRTIMR ;SET TIMER
1474 E86D AD 09 D2 LDA KBCODE ;GET THE KEY
1475 E870 BD FC 02 STA CH ;PUT INTO CH
1476 E873 A0 01 ;READ GAME CONTROLLERS
1477 E875 A2 03 SYSVB7: LDY #1
1478 E877 B9 00 D3 STLOOP: LDA PORTA,Y
1479 E87A 4A LSR A
1480 E87B 4A LSR A
1481 E87C 4A LSR A
1482 E87D 4A LSR A
1483 E87E 9D 7B 02 STA STICKO,X ;STORE JOYSTICK
1484 E881 CA DEX
1485 E882 B9 00 D3 LDA PORTA,Y
1486 E885 29 0F AND #$F
1487 E887 9D 7B 02 STA STICKO,X ;STORE JOYSTICK
1488 E88A CA DEX
1489 E88B 88 DEY
1494  EBBC  10  E9  BPL  STLOOP  
1495  
1496  EBBE  A2  03  
1497  EB90  BD  10  D0  STRL:  LDA  TRIGO,X  ;MOVE JOYSTICK TRIGGERS  
1498  EB93  9D  B4  02  STA  STRIGO,X  
1499  EB96  BD  00  D2  LDA  POTO,X  ;MOVE POT VALUES  
1500  EB99  9D  70  02  STA  PADDLO,X  
1501  EB9C  BD  04  D2  LDA  POT4,X  
1502  EB9F  BD  74  02  STA  PADDL4,X  
1503  EBA2  CA  DEX  
1504  EBA3  10  EB  BPL  STRL  
1505  EBA5  BD  08  D2  STA  PTOGO  ;START POTS FOR NEXT TIME  
1506  
1507  EBA8  A2  06  LDX  #3  
1508  EBA9  A0  03  LDR  A  
1509  EBAE  B9  78  02  PTRLP:  LDA  STICKO,Y  ;TRANSFER BITS FROM JOYSTICKS  
1510  EBAF  4A  LSR  A  ;TO PADDLE TRIGGERS  
1511  EBB0  4A  LSR  A  
1512  EBB1  4A  LSR  A  
1513  EBB2  9D  7D  02  STA  PTRIG1,X  
1514  EBB5  A9  00  LDA  #0  
1515  EBB7  2A  ROL  A  
1516  EBB8  9D  7C  02  STA  PADDLO,X  
1517  EBBB  CA  DEX  
1518  EBBE  CA  DEX  
1519  EBBF  88  DEY  
1520  EBBE  10  EC  BPL  PTRLP  
1521  
1522  ECB0  6C  24  02  JMP  (VBLK/256)  ;GO TO DEFERRED VBLANK ROUTINE  
1523  00EB  
1524  0073  SV7H = SYSVB7/256  
1525  025C  SV7L = (-256)*SV7H+SYSVB7  
1526  EBC3  A9  00  SYSVB6:  LDA  #0  
1527  EBC8  FO  A9  BEQ  SYSVB7  ;UNCOND  
1528  EBC9  6C  26  02  JTIMR1:  JMP  (CDTMA1)  
1529  EBCD  6C  28  02  JTIMR2:  JMP  (CDTMA2)  
1530  ; SUBROUTINE TO DECREMENT A COUNTDOWN TIMER  
1531  ; ENTRY X=OFFSET FROM TIMER 1  
1532  ; EXIT A,P=ZERO IF WENT ZERO, FF OTHERWISE  
1533  ;  
1534  1535  EBD0  BC  18  02  DCTIMR:  LDY  CDTMV1,X  ;LO BYTE  
1536  EBD3  D0  08  BNE  DCTIM1  ;NONZERO, GO DEC IT  
1537  EBD5  BC  19  02  LDY  CDTMV1+1,X  ;SEE IF BOTH ZERO  
1538  EBD8  F0  10  BEQ  DCTXF  ;YES, EXIT NONZERO  
1539  EBD8  DE  19  02  DEC  CDTMV1+1,X  ;DEC HI BYTE  
1540  EBD9  DE  18  02  DCTIM1:  DEC  CDTMV1,X  ;DEC LO BYTE  
1541  EBE0  D0  08  BNE  DCTXF  
1542  EBE2  BC  19  02  LDY  CDTMV1+1,X  
1543  EBE5  D0  03  BNE  DCTXF  
1544  EBE7  A9  00  LDA  #0  ;WENT ZERO, RETURN ZERO  
1545  EBE9  60  RTS  
1546  EBEA  A9  FF  DCTXF:  LDA  #$FF  ;RETURN NONZERO  
1547  EBE8  60  RTS
SUBROUTINE TO SET VERTICAL BLANK VECTORS AND TIMERS
ENTRY X=HI,Y=LO BYTE TO SET
A= 1-5 TIMERS 1-5
6 IMM VBLANK
7 DEF VBLANK
SETVBL: ASL A
MUL BY 2
STA INTEMP
LDX #5
STA WSYNC
WASTE 20 CPU CYCLES
DEX TO ALLOW VBLANK TO HAPPEN
BNE SETLOP IF THIS IS LINE "7C"
LOX #5
STA CDTMV1-1,X
E8F8 DO BNE SETLOP IF THIS IS LINE "7C"
LOX INTEMP
90 17 02 STA COTMV1-1,X
TYA
TAY
PLA UNSTACK Y
PLA UNSTACK X
PLA UNSTACK A
RTI AND GO BACK FROM WHENCE.
PIRG = PIRG/256
PIRGL = (-256)*PIRG+PIRG
PNMIH = PNMI/256
PNMIL = (-256)*PNMI+PNMI
**=14
SPARE BYTE OR MODULE TOO LONG FLAG
CRNTP2 ==
INTSPR: BYTE SIOORG-CRNTP2 ; GINTHV IS TOO LONG
.TITLE 'SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)'

COLLEEN OPERATING SYSTEM

SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)
WITH SOFTWARE BAUD RATE CORRECTION ON CASSETTE

AL MILLER 3-APR-79

THIS MODULE HAS ONE ENTRY POINT. IT IS CALLED BY THE DEVICE HANDLERS. IT INTERPRETS A PREVIOUSLY ESTABLISHED DEVICE CONTROL BLOCK (STORED IN GLOBAL RAM) TO ISSUE COMMANDS TO THE SERIAL BUS TO CONTROL TRANSMITTING AND RECEIVING DATA.
; SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)

; 1604 PAGE
; 1605 EQUATES
; 1606 ; DCD DEVICE BUS ID NUMBERS
; 1607 FLOPPY = $30
; 1608 PRINTR = $40
; 1609 CASSET = $60
; 1610 CASSET = $60
; 1611
; 1612 ; BUS COMMANDS
; 1613 ;
; 1614 READ = 'R
; 1615 WRITE = 'W
; 1616 ;STATIS = 'S
; 1617 ;FORMAT = '!
; 1618
; 1619 ; COMMAND AUX BYTES
; 1620 ;
; 1621 SIDWAY = 'S
; 1622 NORMAL = 'N
; 1623 DOUBLE = 'D
; 1624 PLOT = 'P
; 1625
; 1626 ; BUS RESPONSES
; 1627 ;
; 1628 ACK = 'A
; 1629 NACK = 'N
; 1630 COMPLT = 'C
; 1631 ERROR = 'E
; 1632
; 1633 ; MISCELLANEOUS EQUATES
; 1634 ;
; 1635 B192LO = $28
; 1636 B192HI = $00
; 1637 B600LO = $CC
; 1638 B600HI = $05
; 1639 HITONE = $05
; 1640 LOTONE = $07
; 1641
; 1642 WIRGLO = 150
; 1643 RIRGLO = 100
; 1644 WSRQ = 13
; 1645 RSRQ = 8
; 1646 ; IF PALFLG = 1
; 1647 ; WRITE INTER RECORD GAP (IN 1/60 SEC)
; 1648 ; READ INTER RECORD GAP (IN 1/60 SEC)
; 1649 ; SHORT WRITE INTER RECORD GAP
; 1650 ; SHORT READ INTER RECORD GAP
; 1651 ; ENDIF
; 1652 ; IF PALFLG-1
; 1653 ; WRITE INTER RECORD GAP (IN 1/60 SEC)
; 1654 ; READ INTER RECORD GAP (IN 1/60 SEC)
; 1655 ; SHORT WRITE INTER RECORD GAP
; 1656 ; SHORT READ INTER RECORD GAP
; 1657 ;
ERR LINE ADDR B1 B2 B3 B4

1658          ENDIF
1659  0000    WIRGHI =  0
1660  0000    RIRGHI =  0
1661          
1662  0034    NCOMLO = $34 ;PIA COMMAND TO LOWER NOT COMMAND LINE
1663  003C    NCOMHI = $3C ;PIA COMMAND TO RAISE NOT COMMAND LINE
1664  0034    MOTRGO = $34 ;PIA COMMAND TO TURN ON CASSETTE MOTOR
1665  003C    MOTRST = $3C ;PIA COMMAND TO TURN OFF MOTOR
1666          
1667  0002    TEMPHI = TEMP/256 ;ADDRESS OF TEMP CELL (HI BYTE)
1668  003E    TEMPLE = (-256)*TEMPHI+TEMP ;(LO BYTE)
1669  0002    CBUFHI = CDEVIC/256 ;ADDRESS OF COMMAND BUFFER (HI BYTE)
1670  003A    CBUFLO = (-256)*CBUFHI+CDEVIC ;(LO BYTE)
1671          
1672  000D    CRETRI = 13 ;NUMBER OF COMMAND FRAME RETRIES
1673  0001    DRETRI =  1 ;NUMBER OF DEVICE RETRIES
1674  0002    CTIMLO =  2 ;COMMAND FRAME ACK TIME OUT (LO BYTE)
1675  0000    CTIMHI =  0 ;COMMAND FRAME ACK TIME OUT (HI BYTE)
1676          
1677          
1678          ;JTADRH = JTIMER/256 ;HI BYTE OF JUMP TIMER ROUTINE ADDR "M
1679          ;JTADRL = (-256)*JTADRH+JTIMER ;"MOVED TO LINE 1428"
ERR LINE ADDR B1 B2 B3 B4 SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)

1681 .PAGE
1682 SID
1683
1684
1685
1686 E459 4C 59 E9 **SIOV JMP SIO :SIO ENTRY POINT
1687
1688
1689 E455 4C 44 E9 **SIOINV JMP SIOINT :SIO INITIALIZATION ENTRY POINT
1690
1691
1692 E468 4C F2 EB **SENDEV JMP SENDEV :SEND ENABLE ENTRY POINT
1693
1694
1695
1696 E46A 0F EB **VCTABL-INTABS+VSERIN
1697 E4BC 90 EA
1698 E4BE CF EA
1699
1700
1701
1702
1703
1704
1705
1706 E944 A9 3C SIOINT: LDA #MOTRST
1707 E946 8D 02 D3 STA PACTL :TURN OFF MOTOR
1708
1709 E949 A9 3C LDA #NCOMHI
1710 E94B 8D 03 D3 STA PBCTL :RAISE NOT COMMAND LINE
1711
1712
1713 E94E A9 03 LDA #3
1714 E950 8D 32 02 STA SSKCTL :GET POKEY OUT OF INITIALIZE MODE
1715 E953 85 41 STA SOUNDR :INIT POKE ADDRESS FOR QUIET I/O
1716 E955 8D 0F D2 STA SKCTL
1717
1718
1719 E958 60 RTS :RETURN
1720
1721
1722
1723
1724
1725
1726 E959 BA SID: TSX
1727 E95A BE 18 03 STX STACKP :SAVE STACK POINTER
1728 E95D A9 01 LDA #1
1729 E95F 85 42 STA CRITIC
1730
1731 E961 AD 00 03 LDA DDEVIC
1732 E964 C9 60 CMP #CASET
1733 E966 DD 03 BNE NOTCST :BRANCH IF NOT CASSETTE
1734 E968 4C 80 EB JMP CASENT :OTHERWISE JUMP TO CASSETTE ENTER
ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT

; NOTCST: LDA #0
; INIT CASSETTE FLAG TO NO CASSETTE

; LDA #DRETRI
; SET NUMBER OF DEVICE RETRIES

; CMDND: LDA #CRETRI
; SET NUMBER OF COMMAND FRAME RETRIES

; SEND A COMMAND FRAME

; CONFIRM: LDA #B192LO
; SET BAUD RATE TO 19200

; LDA #B192HI
; SET BUS ID NUMBER

; LDA DCOMND
; SET BUS COMMAND

; LDA DAUX1
; STORE COMMAND FRAME AUX BYTES 1 AND 2

; LDA DAUX2
; DONE SETTING UP COMMAND BUFFER

; LDA #CBUFLO
; DONE SETTING UP BUFFER POINTER

; LDA #CBUFS
; DONE SETTING UP BUFFER POINTER

; LDA #CBUFHI
; DONE SETTING UP BUFFER POINTER

; JSR SENDIN
; SEND THE COMMAND FRAME TO A SMART DEVICE

; LDA ERRFLG
; BRANCH IF AN ERROR RECEIVED

; BNE BADCOM
; BRANCH IF ACK RECEIVED

; BNE ACKREC
; A NACK OR TIME OUT OCCURED
ERR LINE ADDR B1 B2 B3 B4

SID (SERIAL BUS INPUT/OUTPUT CONTROLLER)

<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1789</td>
<td>E9C1</td>
<td>10</td>
<td>B5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1790</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1791</td>
<td>E9C3</td>
<td>4C</td>
<td>06</td>
<td>EA</td>
<td></td>
</tr>
<tr>
<td>1792</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1793</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1794</td>
<td>E9C6</td>
<td>AD</td>
<td>03</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>1795</td>
<td>E9C9</td>
<td>10</td>
<td>OC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1796</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1797</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1798</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1799</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1801</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1802</td>
<td>E9CB</td>
<td>A9</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1803</td>
<td>E9CD</td>
<td>85</td>
<td>36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1804</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1805</td>
<td>E9CF</td>
<td>20</td>
<td>6A</td>
<td>EB</td>
<td></td>
</tr>
<tr>
<td>1806</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1807</td>
<td>E9D2</td>
<td>20</td>
<td>8A</td>
<td>EC</td>
<td></td>
</tr>
<tr>
<td>1808</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1809</td>
<td>E9D5</td>
<td>F0</td>
<td>EB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1811</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1812</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1813</td>
<td>E9D7</td>
<td>20</td>
<td>75</td>
<td>EC</td>
<td></td>
</tr>
<tr>
<td>1814</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1815</td>
<td>E9DA</td>
<td>A9</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1816</td>
<td>E9DC</td>
<td>8D</td>
<td>3F</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>1817</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1818</td>
<td>E9E2</td>
<td>F0</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1819</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1820</td>
<td>E9E4</td>
<td>2C</td>
<td>03</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>1821</td>
<td>E9E7</td>
<td>70</td>
<td>07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1822</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1823</td>
<td>E9E9</td>
<td>AD</td>
<td>3F</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>1824</td>
<td>E9EC</td>
<td>D0</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1825</td>
<td>E9EE</td>
<td>F0</td>
<td>1D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1826</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1827</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1828</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1829</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1830</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1831</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1832</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1833</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1834</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1835</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1836</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1837</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1838</td>
<td>E9F0</td>
<td>20</td>
<td>6A</td>
<td>EB</td>
<td></td>
</tr>
<tr>
<td>1839</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1840</td>
<td>E9F3</td>
<td>20</td>
<td>E0</td>
<td>EA</td>
<td></td>
</tr>
<tr>
<td>1841</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1842</td>
<td>E9F6</td>
<td>AD</td>
<td>3F</td>
<td>02</td>
<td></td>
</tr>
</tbody>
</table>

B1: BPL COMFRM; SO BRANCH IF ANY RETRIES LEFT
B2: JMP DERR1; OTHERWISE, JUMP TO RETURN SECTION
B3: LDA DSTATS; ACK WAS RECEIVED
B4: BPL WATCOM; BRANCH TO WAIT FOR COMPLETE.

1796: IF THERE IS NO DATA TO BE SENT

1800: SEND A DATA FRAME TO PERIPHERAL

1802: LDA #CRETRI; SET NUMBER OF RETRIES
1803: STA CRETRY
1805: JSR LDPNTR; LOAD BUFFER POINTER WITH DCB INFORMATION
1807: JSR SENDIN; GO SEND THE DATA FRAME TO A SMART DEVICE
1809: BEG BADCOM; BRANCH IF BAD

1815: WATCOM: JSR SITMOT; SET DDEVICE TIME OUT VALUES IN Y, X
1817: LDA #$00; CLEAR ERROR FLAG
1818: STA ERRFLG
1820: JSR WAITER; SET UP TIMER AND WAIT
1821: BEG DERR; BRANCH IF TIME OUT
1824: DEVICE DID NOT TIME OUT

1826: BIT DSTATS; BRANCH IF MORE DATA Follows
1827: BVS MODATA; BRANCH IF AN ERROR OCCURRED
1829: LDA ERRFLG; BRANCH IF AN ERROR OCCURRED
1830: BNE DERR1; BRANCH IF AN ERROR OCCURRED
1831: BEG RETURN; OTHERWISE RETURN

1838: MODATA: JSR LDPNTR; LOAD BUFFER POINTER WITH DCB INFORMATION
1840: JSR RECEIV; GO RECEIVE A DATA FRAME

DERR: LDA ERRFLG
ERR LINE ADDR B1 B2 B3 B4 SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)

1843 E9F9 F0 05 BEG NOTERR ;BRANCH IF NO ERROR PRECEEDED DATA
1844
1845 E9FB AD 19 03 LDA TSTAT ;GET TEMP STATUS
1846 E9FE 85 30 STA STATUS ;STORE IN REAL STATUS
1847
1848
1849 EA00 A5 30 NOTERR: LDA STATUS
1850 EA02 C9 01 CMP #$SUCCES
1851 EA04 F0 07 BEG RETURN ;BRANCH IF COMPLETELY SUCCESSFUL
1852
1853 EA06 C6 37 DERR1: DEC DRETRY ;BRANCH IF OUT OF DEVICE RETRIES
1854 EA08 30 03 BMI RETURN ;BRANCH IF OUT OF DEVICE RETRIES
1855
1856 EA0A 4C 74 E9 JMP COMMND ;OTHERWISE, ONE MORE TIME
1857
1858
1859
1860
1861 EA0D 20 5F EC RETURN: JSR SENDDS ;DISABLE POKEY INTERRUPTS
1862 EA10 A9 00 LDA #$0
1863 EA12 85 42 STA CRITIC
1864 EA14 A4 30 LDY STATUS ;RETURN STATUS IN Y
1865 EA16 8C 03 03 STY DSTATS ;AND THE DCB STATUS WORD
1866 EA19 60 RTS RETURN
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876 EA1A A9 00 WAIT: LDA #$00 ;CLEAR ERROR FLAG
1877 EA1C 8D 3F 02 STA ERRFLG ;LOAD BUFFER POINTER WITH ADDRESS
1878
1879 EA1F 18 CLC
1880 EA20 A9 3E LDA #$TEMPLO ;OF TEMPORARY RAM CELL
1881 EA22 85 32 STA BUFRLD
1882 EA24 69 01 ADC #$1
1883 EA26 85 34 STA BUFHLD
1884 EA28 A9 02 LDA #$TEMPHI
1885 EA2A 85 33 STA BUFHLO ;ALSO SET BUFFER END +1 ADDRESS
1886 EA2C 85 35 STA BUFHLO ;DONE LOADING POINTER
1887
1888 EA2E A9 FF LDA #$FF ;SET NO CHECKSUM Follows DATA FLAG
1889 EA30 85 3C STA NOCKSM
1890
1891 EA32 20 E0 EA JSR RECEIV ;GO RECEIVE A BYTE
1892
1893 EA35 A0 FF LDY #$FF ;ASSUME SUCCESS
1894 EA37 A5 30 LDA STATUS
1895 EA39 C9 01 CMP #$SUCCES
1896 EA3B D0 19 BNE NWOK ;BRANCH IF IT DID NOT WORK OK
ERR LINE ADDR B1 B2 B3 B4 SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER) PAGE 45

1897 ;
1898 ;
1899 ;
1900 ;
1901 EA3D AD 3E 02 WOK: LDA TEMP ;MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
1902 EA40 C9 41 CMP #ACK ;WAS ACTUALLY AN ACK OR COMPLETE
1903 EA42 F0 21 BEG GOOD
1904 EA44 C9 43 CMP #COMPLT
1905 EA46 F0 1D BEG GOOD
1906 ;
1907 EA48 C9 45 CMP #ERROR ;BRANCH IF DEVICE DID NOT SEND BACK
1908 EA4A D0 06 BNE NOTDER ;A DEVICE ERROR CODE
1909 ;
1910 EA4C A9 90 LDA #ERROR
1911 EA4E 85 30 STA STATUS ;SET DEVICE ERROR STATUS
1912 EA50 D0 04 BNE NWOK
1913 ;
1914 EA52 A9 B8 NOTDER: LDA #$DACK ;OTHERWISE SET NACK STATUS
1915 EA54 85 30 STA STATUS
1916 ;
1917 EA56 A5 30 NWOK: LDA STATUS
1918 EA58 C9 8A CMP #TIMOUT
1919 EA5A F0 07 BEG BAD ;BRANCH IF TIME OUT
1920 ;
1921 EA5C A9 FF LDA #$FF
1922 EA5E 8D 3F 02 STA ERRFLG ;SET SOME ERROR FLAG
1923 EA61 D0 02 BNE GOOD ;RETURN WITH OUT SETTING Y = 0
1924 ;
1925 EA63 A0 00 BAD: LDY #0
1926 ;
1927 EA65 A5 30 GOOD: LDA STATUS
1928 EA67 8D 19 03 STA TSTAT
1929 EA6A 60 RTS ;RETURN
1930 ;
1931 ;
1932 ;
1933 ;
1934 ;
1935 ;
1936 ;
1937 ;
1938 ;
1939 ;
1940 EA6B A9 01 SEND: LDA #SUCCESS ;ASSUME SUCCESS
1941 EA6D 85 30 STA STATUS
1942 ;
1943 EA6F 20 F2 EB JSR SENDEN ;ENABLE SENDING
1944 ;
1945 EA72 A0 00 LDY #0
1946 EA74 84 31 STY CHKSUM ;CLEAR CHECK SUM
1947 EA76 84 3B STY CKSNT ;CHECKSUM SENT FLAG
1948 EA78 84 3A STY XMTDON ;TRANSMISSION DONE FLAG
1949 ;
1950 ;
ERR LINE ADDR B1 B2 B3 B4

1951 EA7A B1 32 LDA (BUFRLO),Y ;PUT FIRST BYTE FROM BUFFER
1952 EA7C BD DD BD STA SEROUT ;INTO THE SERIAL OUTPUT REGISTER
1953
1954
1955 EA7F B5 31 STA CHKSUM ;PUT IT IN CHECKSUM
1956
1957 EA81 A5 11 NOTDON: LDA BRKKEY
1958 EA83 DD 03 BNE NTRBKO ;JUMP IF BREAK KEY PRESSED
1959 EA85 4C A0 ED JMP BROKE
1960
1961 EA86 A5 3A NTBRKO: LDA XMTDON ;LOOP UNTIL TRANSMISSION IS DONE
1962 EA8C FF 05 BEQ NOTDON
1963
1964 EA8D 02 05 EC JSR SENDDS ;DISABLE SENDING
1965
1966 EA8F 00 02 STA SEROUT
1967
1968
1969
1970
1971
1972
1973
1974
1975 EA90 98 ISRODN: TYA ;OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
1976 EA91 48 PHA ;SAVE Y REG ON STACK
1977
1978 EA92 E6 32 INC BUFRLO ;INCREMENT BUFFER POINTER
1979 EA94 DO 02 BNE NOWRPO
1980 EA96 E6 33 INC BUFRHI
1981
1982 EA98 A5 32 NOWRPO: LDA BUFRLO ;CHECK IF PAST END OF BUFFER
1983 EA9A C5 34 CMP BFENLO
1984 EA9C A5 33 LDA BUFRHI ;HIGH PART
1985 EA9E E5 35 SBC BFENHI
1986 EAAC 90 0C BCC NOTEND ;BRANCH IF NOT PAST END OF BUFFER
1987
1988 EAAC A5 3B LDA CHKSNT
1989 EAAC DD 08 BNE RELONE ;BRANCH IF CHECKSUM ALREADY SENT
1990
1991 EAAB A5 31 LDA CHKSUM
1992 EAAB BD 00 BD STA SEROUT ;SEND CHECK SUM
1993 EAAB A9 FF LDA #$FF
1994 EAAD B5 38 STA CHKSNT ;SET CHECKSUM SENT FLAG
1995 EAAF DD 09 BNE CHKDON
1996
1997 EAAB A5 10 RELONE: LDA POKMSK ;ENABLE TRANSMIT DONE INTERRUPT
1998 EAAB 09 08 ORA #$08
1999 EAAB B5 10 STA POKMSK
2000 EAAB 0D 00 D2 STA IRGEN
2001
2002 EAAB 68 CHKDON: PLA
2003 EAAB A8 TAY ;RESTORE Y REG
2004 EAABC 68 PLA ;RETURN FROM INTERRUPT
ERR LINE ADDR  B1 B2 B3 B4  SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )  PAGE 47

2005 EABD  40  RTI
2006  ;
2007  ;
2008 EABE  A0  00  NOTEND:  LDY  #0
2009 EAC0  B1  32  LDA (BUFRLO),Y  ;PUT NEXT BYTE FROM BUFFER
2010 EAC2  8D  0D  D2  STA  SEROUT  ;INTO THE SERIAL OUTPUT REGISTER
2011  ;
2012 EAC5  18  CLC  ;ADD IT TO CHECKSUM
2013 EAC6  65  31  ADC  CHKSUM
2014 EAC8  67  00  ADC  #0
2015 EACA  85  31  STA  CHKSUM
2016  ;
2017 EACC  4C  BA  EA  JMP  CHKDON  ;GO RETURN
2018  ;
2019  ;
2020  ;
2021  ;
2022  ;
2023  ;
2024 ; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
2025  
2026 EACF  A5  3B  ISRTD:  LDA  CHKSN
2027 EAD1  F0  0B  BEQ  FOOEY  ;BRANCH IF CHECKSUM NOT YET SENT
2028  ;
2029 EAD3  85  3A  STA  XMTDON  ;OTHERWISE SET TRANSMISSION DONE FLAG
2030  ;
2031 EAD5  A5  10  LDA  POKMSK  ;DISABLE TRANSMIT DONE INTERRUPT
2032 EAD7  29  F7  AND  #F7
2033 EAD9  85  10  STA  POKMSK
2034 EADB  8D  0E  D2  STA  IRQEN
2035  ;
2036 EAEC  6B  FOOEY:  PLA  ;RETURN FROM INTERRUPT
2037 EADF  40  RTI
2038  ;
2039  ;
2040  ;
2041  ;
2042  ;
2043  ;
2044  ;
2045  ;
2046 ; RECEIVE SUBROUTINE
2047  
2048 EAE0  A9  00  RECEIV:  LDA  #0
2049  ;
2050 EAE2  AC  OF  03  LDY  CASFLG
2051 EAE5  D0  02  BNE  NOCLR  ;BRANCH IF CASSETTE
2052  ;
2053 EAE7  85  31  STA  CHKSUM  ;CLEAR CHKSUM
2054 EAE9  85  3B  NOCLR:  STA  BUFRFL  ;BUFFER FULL FLAG
2055 EAEB  85  39  STA  RCVDN  ;RECEIVE DONE FLAG
2056  ;
2057  ;
2058  ;
ERR LINE ADDR B1 B2 B3 B4  SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)  PAGE 48

2059 EAED A9 01  LDA  #SUCCES  ;SET GOOD STATUS FOR DEFAULT CASE.
2060 EAEE B5 30  STA  STATUS  ;DO RECEIVE ENABLE
2061 EAFC 20 1B EC  JSR  RECVEN  ;COMMAND FRAME HI COMMAND
2062 EAFA A9 3C  LDA  #NCOMHI  ;STORE IN PIA
2063 EAFA B5 03 D3  STA  PBCTL  ;PAGE 48
2064 EAFA A5 11  CHKTIM: LDA  BRKKEY  ;DO
2065 EAFA D0 03  BNE  NTBRKI  ;RECEIVE ENABLE
2066 EAFA 4C A0 ED  JMP  BROKE  ;JUMP IF BREAK KEY Pressed
2067 ;
2068 EA00 AD 17 03  NTBRKI: LDA  TIMFLG  ;NO,
2069 EA03 F0 05  BEQ  TOUT  ;IF TIMEOUT, GO SET ERROR STATUS
2070 EA05 A5 39  LDA  RECVDN  ;DONE?
2071 EA07 F0 F0  BEQ  CHKTIM  ;STORE
2072 EA09 60  GBACK: RTS  ;IN PIA
2073 EA0A A9 8A  TOUT: LDA  #TIMOUT  ;YES,
2074 EA0C B5 30  STA  STATUS  ;SET TIMEOUT STATUS
2075 ;
2076 ;
2077 ;
2078 ;
2079 ;
2080 ;
2081 EA0E 60  RRETRN: RTS  ;RETURN
2082 ;
2083 ;
2084 ;
2085 ;
2086 ;
2087 ;
2088 ;
2089 ;  ;SERIAL INPUT READY INTERRUPT SERVICE ROUTINE
2090 ;
2091 EA0F 98  ISRSIR: TYA
2092 EA10 48  PHA  ;SAVE Y REG ON STACK
2093 ;
2094 ;
2095 ;
2096 EA11 AD 0F D2  LDA  SKSTAT  ;RESET STATUS REGISTER
2097 EA14 8D 0A D2  STA  SKRES  ;****** THIS MAY NOT BE THE PLACE TO DO IT ******
2098 ;
2099 ;
2100 EA17 30 04  BMI  NTFRAM  ;BRANCH IF NO FRAMING ERROR
2101 ;
2102 EA19 A0 8C  LDY  #FRMERR  ;SET FRAME ERROR STATUS
2103 EA1B 84 30  STY  STATUS  ;
2104 ;
2105 EA1D 29 20  NTFRAM: AND  #$20
2106 EA1F D0 04  BNE  NTOVRN  ;BRANCH IF NO OVERRUN ERROR
2107 ;
2108 EA21 A0 8E  LDY  #OVRRUN  ;SET OVERRUN ERROR STATUS
2109 EA23 84 30  STY  STATUS  ;
2110 ;
2111 EA25 A5 38  NTOVRN: LDA  BUFRL  ;BRANCH IF BUFFER WAS NOT YET FILLED
2112 EA27 F0 13  BEQ  NOTYET  ;
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2113</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>THIS INPUT BYTE IS THE CHECKSUM</td>
</tr>
<tr>
<td>2114</td>
<td>EB29</td>
<td>AD</td>
<td>00</td>
<td>D2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2115</td>
<td>EB2C</td>
<td>C5</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2116</td>
<td>EB2E</td>
<td>F0</td>
<td>04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2118</td>
<td>EB30</td>
<td>A0</td>
<td>8F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2119</td>
<td>EB32</td>
<td>84</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2121</td>
<td>EB34</td>
<td>A9</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2122</td>
<td>EB36</td>
<td>85</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2124</td>
<td>EB38</td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2125</td>
<td>EB39</td>
<td>A8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2126</td>
<td>EB3A</td>
<td>68</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2127</td>
<td>EB3B</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2129</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2130</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2131</td>
<td>EB3C</td>
<td>AD</td>
<td>00</td>
<td>D2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2132</td>
<td>EB3F</td>
<td>A0</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2133</td>
<td>EB41</td>
<td>91</td>
<td>32</td>
<td></td>
<td></td>
<td>STORE INPUT REGISTER INTO BUFFER</td>
</tr>
<tr>
<td>2134</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2135</td>
<td>EB43</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2136</td>
<td>EB44</td>
<td>65</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2137</td>
<td>EB46</td>
<td>69</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2138</td>
<td>EB48</td>
<td>85</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2139</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2140</td>
<td>EB4A</td>
<td>E6</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2141</td>
<td>EB4C</td>
<td>D0</td>
<td>02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2142</td>
<td>EB4E</td>
<td>E6</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2143</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2144</td>
<td>EB50</td>
<td>A5</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2145</td>
<td>EB52</td>
<td>C5</td>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2146</td>
<td>EB54</td>
<td>A5</td>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2147</td>
<td>EB56</td>
<td>E5</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2148</td>
<td>EB58</td>
<td>90</td>
<td>DE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2149</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2150</td>
<td>EB5A</td>
<td>A5</td>
<td>3C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2151</td>
<td>EB5C</td>
<td>F0</td>
<td>06</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2152</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2153</td>
<td>EB5E</td>
<td>A9</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2154</td>
<td>EB60</td>
<td>85</td>
<td>3C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2156</td>
<td>EB62</td>
<td>F0</td>
<td>D0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2157</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2159</td>
<td>EB64</td>
<td>A9</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2160</td>
<td>EB66</td>
<td>85</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2161</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2162</td>
<td>EB68</td>
<td>D0</td>
<td>CE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2163</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2164</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2165</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2166</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LOAD BUFFER POINTERS SUBROUTINE

LOAD BUFFER POINTER WITH DCB BUFFER INFORMATION

```
; LOAD BUFFER POINTER SUBROUTINE

LDPNTR: CLC
LDPNTR: CLC
LDA DBUFLO
LDA DBUFLO
STA BUFRL0
STA BUFRL0
; ALSO SET BUFFER END + 1 ADDRESS

LDA DBUFHI
LDA DBUFHI
STA BUFRH0
STA BUFRH0

EB6A 18  ; LDPNTR: CLC
EB6B AD 04 03  ; LDA DBUFLO
EB6E 85 32  ; STA BUFRL0
EB70 6D 08 03  ; ADC DBYLO
EB73 85 34  ; STA BFENLO

EB75 AD 05 03  ; LDA DBUFHI
EB7B 85 33  ; STA BUFRH0
EB7A 6D 09 03  ; ADC DBYTHI
EB7D 85 35  ; STA BFENHI

EB7F 60  ; RTS ; RETURN

; CASSETTE HANDLING CODE

EB80 AD 03 03  ; CASENT: LDA DSTATS
EB83 10 2E  ; BPL CASRED ; BRANCH IF INPUT FROM CASSETTE

EB85 A9 CC  ; WRITE A RECORD
EB87 8D 04 D2  ; LDA #B600LO ; SET BAUD RATE TO 600
EB88 A9 05  ; STA AUDF3
EB8A 8D 06 D2  ; LDA #B600HI
EB8C 8D 06 D2  ; STA AUDF4

EB8F 20 F2 EB  ; JSR SENDEN ; TURN ON POKEY MARK TONE

EB92 A0 0F  ; LDY #WSIRG ; LOAD SHORT WRITE INTER RECORD GAP TIME
EB94 AD 08 03  ; LDA DAUX2
EB97 30 02  ; BMI SRTIRO ; BRANCH IF SHORT GAP IS DESIRED

EB99 A0 B4  ; LDY #WIROLO ; SET WRITE IRG TIME
EB9B A2 00  ; SRTIRO: LDX #WIROHI
EB9D 20 B9 ED  ; JSR SETVBX

EBA0 A9 34  ; LDA #MOTRGO
EBA2 8D 02 D3  ; STA PACTL ; TURN ON MOTOR

EBA5 AD 17 03  ; TIMIT: LDA TIMFLG ; LOOP UNTIL DONE
```
ERR LINE | ADDR B1 B2 B3 B4 | SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER) | PAGE 51
---|---|---|---
2221 | EBAB DO FB | | |
2222 | | | |
2223 | EBAA 20 6A EB | | |
2224 | | | |
2225 | EBBAD 68 EA | | |
2226 | | | |
2227 | EBBO 4C DF EB | | |
2228 | | | |
2229 | | | |
2230 | | | |
2231 | | | |
2232 | | | |
2233 | EBB3 A9 FF | | |
2234 | EBB5 8D 0F 03 | | |
2235 | | | |
2236 | EBBB A0 0A | | |
2237 | EBBBA 0D 08 03 | | |
2238 | EBBB3 30 02 | | |
2239 | | | |
2240 | EBBF A0 7B | | |
2241 | EBC1 A2 00 | | |
2242 | EBC3 20 89 ED | | |
2243 | | | |
2244 | EBC6 A9 34 | | |
2245 | EBC8 8D 02 D3 | | |
2246 | | | |
2247 | EBC8 AD 17 03 | | |
2248 | EBCE DO FB | | |
2249 | | | |
2250 | EBD0 20 6A EB | | |
2251 | | | |
2252 | EBD3 20 75 EC | | |
2253 | EBD6 20 89 ED | | |
2254 | | | |
2255 | EBD9 20 10 ED | | |
2256 | | | |
2257 | EBDC 20 E0 EA | | |
2258 | | | |
2259 | EBDF A0 0B 03 | | |
2260 | EBEE 30 05 | | |
2261 | | | |
2262 | EBE4 A9 3C | | |
2263 | EBE6 BD 02 D3 | | |
2264 | | | |
2265 | EBE9 4D 0D EA | | |
2266 | | | |
2267 | | | |
2268 | | | |
2269 | | | |
2270 | | | |
2271 | EBEA A9 00 | | |
2272 | 00EB | | |
2273 | 00EC | | |
2274 | EBBE 8D 17 03 | | |

**Note:** The code appears to be a fragment of assembly language, likely for a microcontroller or similar device, with comments indicating its purpose in handling serial bus I/O operations.
ERR LINE ADDR B1 B2 B3 B4 SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER) PAGE 52

2275 EBF1 60 RTS
2276 ;
2277 ;
2278 ;
2279 ;
2280 ;
2281 ;
2282 ;
2283 ;
2284 EBF2 A9 07 SENDEN: LDA #$07 iMASK OFF PREVIOUS SERIAL BUS CONTROL BITS
2285 EBF4 2D 32 02 AND SSKCTL iSET TRANSMIT MODE
2286 EBF7 09 20 ORA #$20 iSET TRANSMIT MODE
2287 ;
2288 EBF9 AC 00 03 LDY DDEVIC
2289 EBFc Co 40 CPY #CASSET
2290 EBFe D0 0c BNE NOTCAS iBRANCH IF NOT CASSETTE
2291 ;
2292 EC00 09 08 ORA #$08 iSET THE FSK OUTPUT BIT
2293 ;
2294 EC02 A0 07 LDY #LTONE iSET FSK TONE FREQUENCIES
2295 EC04 8c 02 D2 STY AUDF2
2296 EC07 A0 05 LDY #HTONE
2297 EC09 8c 00 D2 STY AUDF1
2298 ;
2299 EC0c 8d 32 02 NOTCAS: STA SSKCTL iSTORE NEW VALUE TO SYSTEM MASK
2300 EC0f 8d 0f D2 STA SKCTL iSTORE TO ACTUAL REGISTER
2301 ;
2302 EC12 A9 C7 LDA #$C7 iMASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
2303 EC14 25 10 AND POKMSK
2304 EC16 09 10 ORA #$10 iENABLE OUTPUT DATA NEEDED INTERRUPT
2305 ;
2306 ;
2307 EC1B 4C 31 EC JMP CONTIN iGO CONTINUE IN RECEIVE ENABLE SUBROUTINE
2308 ;
2309 ;
2310 ;
2311 ;
2312 ;
2313 ;
2314 ;
2315 ;
2316 ;
2317 ;
2318 ;
2319 ;
2320 EC1B A9 07 RECEVEN: LDA #$07 iMASK OFF PREVIOUS SERIAL BUS CONTROL BITS
2321 EC1D 2D 32 02 AND SSKCTL
2322 EC20 09 10 ORA #$10 iSTORE NEW VALUE TO SYSTEM MASK
2323 EC22 8d 32 02 STA SSKCTL
2324 EC25 8d 0f D2 STA SKCTL
2325 ;
2326 EC2B 8d 0a D2 STA SKRES iRESET SERIAL PORT/KEYBOARD STATUS REGISTER
2327 ;
2328 EC2B A9 C7 LDA #$C7 iMASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
ERR LINE  ADDR  B1 B2 B3 B4  SIO  (SERIAL BUS INPUT/OUTPUT CONTROLLER)  PAGE  53

2329  EC2D  25  10  AND  POKMSK
2330  EC2F  09  20  ORA  **20  ;ENABLE RECEIVE INTERRUPT
2331  EC31  B5  10  CONTIN: STA  POKMSK  ;STORE NEW VALUE TO SYSTEM MASK
2332  EC33  8D  0E  D2  STA  IRGEN  ;STORE TO ACTUAL REGISTER
2333
2334
2335  EC36  A9  28  LDA  **28  ;CLOCK CH.3 WITH 1.79 MHZ
2336  EC3B  8D  08  D2  STA  AUDCTL  ;CLOCK CH.4 WITH CH. 3
2337
2338  EC3B  A2  06  LDX  #6  ;SET PURE TONES, NO VOLUME
2339  EC3D  A9  A8  LDA  **A8
2340  EC3F  A4  41  LDY  SOUNDR  ;TEST QUIET I/O FLAG
2341  EC41  D0  02  BNE  NOISE1  ;LINE IS NORMAL (NOISY)
2342  EC43  A9  A0  LDA  **A0
2343  EC45  9D  01  D2  NOISE1: STA  AUDC1,X
2344  EC48  CA  DEX
2345  EC49  CA  DEX
2346  EC4A  10  F9  BPL  NOISE1
2347
2348  EC4C  A9  A0  LDA  **A0
2349  EC4E  8D  05  D2  STA  AUDC3  ;TURN OFF SOUND ON CHANNEL 3
2350  EC51  AC  00  03  LDY  DDEVIC
2351  EC54  C0  60  CPY  #CASET
2352  EC56  F0  06  BEQ  CAS31  ;BRANCH IF CASSETTE IS DESIRED
2353  EC58  8D  01  D2  STA  AUDC1  ;OTHERWISE TURN OFF CHANNELS 1 AND 2
2354  EC5B  8D  03  D2  STA  AUDC2
2355
2356
2357  EC5E  60  CAS31: RTS  ;RETURN
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368 ;DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
2369
2370  EC5F  EA  SENDDS: NOP
2371  EC60  A9  C7  RECVDs: LDA  **C7  ;MASK OFF SERIAL BUS INTERRUPTS
2372  EC62  25  10  AND  POKMSK
2373  EC64  B5  10  STA  POKMSK  ;STORE NEW VALUE TO SYSTEM MASK
2374  EC66  8D  0E  D2  STA  IRGEN  ;STORE TO ACTUAL REGISTER
2375
2376  EC69  A2  06  LDX  #6
2377  EC6B  A9  00  LDA  #0
2378  EC6D  9D  01  D2  ZERIT: STA  AUDC1,X
2379  EC70  CA  DEX
2380  EC71  CA  DEX
2381  EC72  10  F9  BPL  ZERIT  ;TURN OFF AUDIO VOLUME
2382
ERR LINE ADDR B1 B2 B3 B4 SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER) PAGE 54

2383 EC74 60 RTS \; \text{RETURN}

2384 \;
2385 \;
2386 \;
2387 \;
2388 \;
2389 \;
2390 \;
2391 \;
2392 \;
2393 \;
2394 \;
2395 \;
2396 EC75 AD 06 03 \text{SET DEVICE TIME OUT VALUES IN Y, X SUBROUTINE}

2397 EC7B 6A \;
2398 EC79 6A \;
2399 EC7A AA \;
2400 EC7B 29 3F \;
2401 EC7D AA \;
2402 \;
2403 EC7E 98 \;
2404 EC7F 6A \;
2405 EC80 29 CO \;
2406 EC82 AB \;
2407 \;
2408 EC83 60 RTS \;
2409 \;
2410 \;
2411 \;
2412 \;
2413 \;
2414 \;
2415 \;
2416 \;
2417 \;
2418 \;
2419 EC84 0F EB \;
2420 EC86 90 EA \;
2421 EC88 CF EA \;
2422 \;
2423 00EB \;
2424 000F \;
2425 00EA \;
2426 0090 \;
2427 00EA \;
2428 00CF \;
2429 \;
2430 \;
2431 \;
2432 \;
2433 \;
2434 \;
2435 \;
2436 EC8A A2 01 SENDIN: LDX #01

STTMOT: LDA DTIMLO \; \text{GET DEVICE TIME OUT IN 1 SECOND INCR}

RDR A \; 6 HI BITS IN X, LO 2 BITS IN Y

TAY \; \text{TEMP SAVE}

AND #$3F \; \text{MASK OFF 2 HI BITS}

TAX \; \text{THIS IS HI BYTE OF TIME OUT}

TYA \; \text{RESTORE}

RDR A \; \text{THIS IS HI BYTE OF TIME OUT}

AND #$C0 \; \text{MASK OFF ALL BUT 2 HI BITS}

TAY \; \text{THIS IS LO BYTE OF TIME OUT}

SEND A DATA FRAME TO AN INTELLIGENT PERIPHERAL SUBROUTINE
2437 EC8C A0 FF DELAY0: LDY #$FF
2438 EC8E B8 DELAY1: DEY
2439 EC8F D0 FD BNE DELAY1
2440 EC91 CA DEX
2441 EC92 D0 FB BNE DELAY0
2442 EC94 20 6B EA JSR SEND ; GO SEND THE DATA FRAME
2443 EC97 A0 02 LDY #$TIMLO ; SET ACK TIME OUT
2444 EC99 A2 00 LDX #$TIMHI
2445 EC9B 20 B9 ED WAITER: JSR SETVbx
2446 EC9E 20 1A EA JSR WAIT ; WAIT FOR ACK
2447 ECA1 98 TYA ; IF Y=0, A TIME OUT OR NACK OCCURED
2448 ECA2 60 RTS ; RETURN
2449 ECA3 BD 10 03 COMPUT: STA TIMER2
2450 ECA6 BC 11 03 STY TIMER2+1 ; SAVE FINAL TIMER VALUE
2451 ECA9 20 04 ED JSR ADJUST ; ADJUST VCOUNT VALUE
2452 ECAC BD 10 03 STA TIMER2 ; SAVE ADJUSTED VALUE
2453 ECAC AD 00 03 LDA TIMER1
2454 ECAC BD 04 ED JSR ADJUST ; ADJUST
2455 ECAC 8D 10 03 STA TIMER1 ; SAVE ADJUSTED TIMER1 VALUE
2456 ECAC AD 00 03 LDA TIMER2
2457 ECAC EC 38 SEC
2458 ECAC BD 04 ED STA TIMER1 ; FIND VCOUNT DIFFERENCE
2459 ECAC AD 12 03 LDA TIMER2 ; FIND VBLANK COUNT DIFFERENCE
2460 ECAC EC 38 SEC
2461 ECAC BD 04 ED STA TIMER1+1
2462 ECAC AD 11 03 LDA TIMER2+1
2463 ECAC DB 00 03 SBC TIMER1+1
2464 ECAC 38 ED SBC TIMER2
2465 ECAC A9 7D IF PALFLG
2466 ECAC A9 B3 LDA #$-9C
2467 ECAC 80 03 HITIMR: CLC
2468 ECAC 80 04 ADC #$9C
2469 ECAC 80 05 ENDIF
2470 ECAC A9 7D IF PALFLG
2471 ECAC A9 B3 LDA #$-83
2472 ECAC A9 B0 IF PALFLG-1
2473 ECAC A9 B3 LDA #$-9C
2491 ECCD 69 83 ADC #$83 ; ACCUMULATE MULTIPLICATION
2492 .ENDIF
2493 ECCF 88 DEY
2494 ECD0 10 FA BPL HITIMR ; DONE?
2495 ECD2 18 CLC
2496 ECD3 60 12 03 ADC TEMP1 ; TOTAL VCOUNT DIFFERENCE
2497 ECD6 AB FINDX: TAY ; SAVE ACCUM
2498 ECD7 4A LSR A
2499 ECD8 4A LSR A
2500 ECD9 4A ASL A
2501 ECDA 0A SEC
2502 ECDB 3B SEC
2503 ECDC E9 16 SBC #22 ; ADJUST TABLE INDEX
2504 ECDE AA TAX ; DIVIDE INTERVAL BY 4 TO GET TABLE INDEX
2505 ECDF 9B TYA ; RESTORE ACCUM
2506 ECEF 29 07 AND #7 ; PULL OFF 3 LO BITS OF INTERVAL
2507 ECE2 AB TAY
2508 ECE3 AF 5F LDA #$-11
2509 ECE5 18 DINTP: CLC
2510 ECE6 69 08 ADC #11 ; ACCUMULATE INTERPOLATION CONSTANT
2511 ECE8 88 DEY
2512 ECE9 10 FA BPL DINTP ; INTERPOLATION CONSTANT COMPUTATION DONE?
2513
2514 ECEB AO 00 ENINTP: LDY #0
2515 ECED 8C OE 03 STY ADDCOR ; CLEAR ADDITION CORRECTION FLAG
2516 ECF0 3B SEC
2517 ECF1 E9 07 SBC #7 ; ADJUST INTERPOLATION CONSTANT
2518 ECF3 10 03 BPL PLUS
2519 ECF5 CE OE 03 DEC ADDCOR
2520 ECF8 18 PLUS: CLC
2521 ECF9 7D 00 ED ADC POKTAB,X ; ADD CONSTANT TO LO BYTE TABLE VALUE
2522 ECFC AB TAY ; LO BYTE POKEY FREQ VALUE
2523 ECFD AD OE 03 LDA ADDCOR
2524 EDO0 7D D1 ED ADC POKTAB+1,X ; ADD CARRY TO HI BYTE TABLE VALUE
2525 ; HI BYTE POKEY FREQ VALUE
2526 EDO3 60 RTS
2527
2528 ; ROUTINE TO ADJUST VCOUNT VALUE
2529
2530
2531
2532 EDO4 C9 7C CMP #$7C
2533 EDO6 30 04 BMI ADJ1 ; LARGER THAN '7C'? 
2534 EDO8 3B SEC
2535 EDO9 E9 7C SBC #$7C
2536 EDOB 60 RTS
2537 ED0C 18 ADJ1: CLC
2538 .IF PALFLG
2539 ADC #$20
2540 .ENDIF
2541 .IF PALFLG-1
2542 EDOD 69 07 ADC #$7
2543 .ENDIF
2544 EDOF 60 RTS
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' HEX.

```
BEGIN: LDA BRKKEY
SNE NTBRK2
JMP IF BREAK KEY PRESSED

OKTIM1: LDA SKSTAT
AND #$10 ;READ SERIAL PORT
BNE BEGIN ;START BIT?
STA SAVIO ;SAVE SER. DATA IN
LDX VCOUNT ;READ VERTICAL LINE COUNTER
LDY RTCLOK+2 ;READ LO BYTE OF VBLANK CLOCK
STX TIMER1 ;SAVE INITIAL TIMER VALUE
STY TIMER1+1

LDX #1 ;SET MODE FLAG
STX TEMP3
LDY #$10 ;SET BIT COUNTER FOR 10 BITS
COUNT: LDA BRKKEY
BEG BROKE ;BRANCH IF BREAK KEY PRESSED

OKIMR: LDA SKSTAT
AND #$10 ;READ SERIAL PORT
BNE OKTIMR ;BRANCH IF NOT TIMED OUT
CLI

OKTIMR: LDA SKSTAT
AND #$10 ;READ SERIAL PORT
BNE OKTMR ;BRANCH IF NOT TIMED OUT
CLI

;DONE WITH BOTH MODES?
```
ERR LINE ADDR B1 B2 B3 B4  SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )  PAGE  58

2599  ED60  A4 14  LDY  RTCLOK+2  READ TIMER LO & HI BYTES
2600  ED62  20  A3 EC  JSR  COMPUT  NO, COMPUTE BAUD RATE
2601  ED65  BC EE 02  STY  CBAUDL
2602  ED68  BD EF 02  STA  CBAU DH  SET BAUD RATE INTO RAM CELLS
2603  ED6B  A0 09  LDY  #9  SET BIT COUNTER FOR 9 BITS
2604  ED6D  DO CC  BNE  COUNT
2605  
2606  ED6F  AD EE 02  QOREAD: LDA  CBAUDL
2607  ED72  8D 04 D2  STA  AUDF3
2608  ED75  AD EF 02  LDA  CBAUDH
2609  ED78  8D 06 D2  STA  AUDF4  SET POKEY FREQUENCY REGS FOR BAUD RATE
2610  ED7B  A9 00  LDA  #0
2611  ED7D  8D 0F D2  STA  SKSTAT
2612  ED80  AD 32 02  LDA  SSKCTL
2613  ED83  8D 0F D2  STA  SKSTAT  INIT. POKEY SERIAL PORT
2614  ED86  2A 32 02  LDA  (BUFRLO),Y  STORE '*55' AS FIRST RCV. BUFFER
2615  ED89  A9 32  STY  INY
2616  ED8C  C8 32 02  LDA  CBAUDL
2617  ED8F  A9 32  STA  (BUFRLO),Y
2618  ED92  85 31  STA  #AA
2619  ED95  18  STA  CHKSUM  STORE CHECKSUM FOR 2 BYTES OF '*AA'
2620  ED98  18  STA  CLC
2621  ED9B  85 32  LDA  BUFRLO
2622  ED9E  89 02  ADC  #2
2623  EDA2  85 32  STA  BUFRLO
2624  EDA5  A9 33  LDA  BUFRHI
2625  EDA8  89 00  ADC  #0
2626  EDAB  85 33  STA  BUFRHI  INCR. BUFFER POINTER BY 1
2627  EDAE  58  STA  CLC
2628  ED9F  60  RTS

2629  
2630  
2631  
2632  EDA0  20 5F EC  BROKE: JSR  SENDDS  BREAK KEY WAS Pressed, SO PREPARE
2633  EDA3  A9 3C  LDA  #MOTRST  TO RETURN
2634  EDA6  8D 02 D3  STA  PACTL  TURN OFF MOTOR
2635  EDA9  8D 03 D3  STA  PBCTL  RAISE NOT COMMAND LINE
2636  
2637  EDAD  A9 80  LDA  #BRKABT
2638  EDAD  85 30  STA  STATUS  STORE BREAK ABORT STATUS CODE
2639  
2640  ED9F  AE 18 03  LDX  STACKP  RESTORE STACK POINTER
2641  EDB2  9A  TXS  
2642  
2643  EDB3  C6 11  DEC  BRKKEY  SET BREAK KEY FLAG TO NONZERO
2644  EDB5  58  CLC  ALLOW IRQ'S
2645  
2646  EDB6  4C 0D EA  JMP  RETURN  GO RETURN
2647  
2648  
2649  
2650  
2651  
2652  EDB9  A9 EC  SETVBX: LDA  #JTADRL  STORE TIME OUT ROUTINE ADDRESS
ERR LINE ADDR B1 B2 B3 B4

SIO (Serial Bus Input/Output Controller)

<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2653</td>
<td>EDBB</td>
<td>80</td>
<td>26</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>2654</td>
<td>EDBE</td>
<td>A9</td>
<td>E8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2655</td>
<td>EDC0</td>
<td>80</td>
<td>27</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>2656</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2657</td>
<td>EDC3</td>
<td>A9</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2658</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2659</td>
<td>EDC5</td>
<td>78</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2660</td>
<td>EDC6</td>
<td>20</td>
<td>5C</td>
<td>E4</td>
<td></td>
</tr>
<tr>
<td>2661</td>
<td>EDC9</td>
<td>A9</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2662</td>
<td>EDC8</td>
<td>80</td>
<td>17</td>
<td>03</td>
<td></td>
</tr>
<tr>
<td>2663</td>
<td>EDCE</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2664</td>
<td>EDCF</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2665</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2666</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2667</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2668</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2669</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2670</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2671</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2672</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2673</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2674</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2675</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2676</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2677</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2678</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2679</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2680</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2681</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2682</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2683</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2684</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2685</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2686</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2687</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2688</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2689</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2690</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2691</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2692</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2693</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2694</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2695</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2696</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2697</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2698</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2699</td>
<td>EDDO</td>
<td>E8</td>
<td>03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2700</td>
<td>EDD2</td>
<td>43</td>
<td>04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2701</td>
<td>EDD4</td>
<td>9E</td>
<td>04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2702</td>
<td>EDD6</td>
<td>F9</td>
<td>04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2703</td>
<td>EDD8</td>
<td>54</td>
<td>05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2704</td>
<td>EDDA</td>
<td>AF</td>
<td>05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2705</td>
<td>EDDE</td>
<td>0A</td>
<td>06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2706</td>
<td>EDDE</td>
<td>65</td>
<td>06</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2653 EDBB 80 26 02 STA CDTMA1
2654 EDBE A9 E8 LDA #JTADRH
2655 EDC0 80 27 02 STA CDTMA1+1
2656      ; THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
2657 EDC3 A9 01 LDA #1 ; SET FOR TIMER 1
2658      ; ANY VBLANKS THAT OCCUR
2659 EDC5 78 SEI
2660 EDC6 20 5C E4 JSR SETVBV
2661 EDC9 A9 01 LDA #1 ; SET FOR TIMER 1
2662 EDC8 80 17 03 STA TIMFLG ; SET FLAG TO NOT TIMED OUT
2663 EDCE 58 CLI
2664 EDCF 60 RTS

‘VCOUNT’ INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE CONVERSION TABLE

THE VALUES STORED IN THE TABLE ARE ‘AUDF+7’.

THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:

\[
F_{\text{OUT}} = F_{\text{IN}} / (2 \times (\text{AUDF+M})) , \quad \text{WHERE } F_{\text{IN}} = 1.78979 \text{ MHZ.} \quad \& \quad M = 7
\]

FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY AN INTERVAL OF THE ‘VCOUNT’ TIMER:

\[
\text{AUDF+7} = (11.365167) \times T_{\text{OUT}}, \quad \text{WHERE } T_{\text{OUT}} = \# \text{ OF COUNTS (127 USEC.RESOLUTION) OF ‘VCOUNT’ FOR 1 CHARACTER TIME (10 BIT TIMES).}
\]

<table>
<thead>
<tr>
<th>AUDF+7</th>
<th>BAUD RATE</th>
<th>VCOUNT INTERVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>$27C</td>
<td>1407</td>
<td>96</td>
</tr>
<tr>
<td>$2D7</td>
<td>1231</td>
<td>64</td>
</tr>
<tr>
<td>$332</td>
<td>1094</td>
<td>72</td>
</tr>
<tr>
<td>$38D</td>
<td>985</td>
<td>90</td>
</tr>
<tr>
<td>$3E8</td>
<td>895</td>
<td>88</td>
</tr>
<tr>
<td>$443</td>
<td>820</td>
<td>96</td>
</tr>
<tr>
<td>$49E</td>
<td>757</td>
<td>104</td>
</tr>
<tr>
<td>$4F9</td>
<td>703</td>
<td>112</td>
</tr>
<tr>
<td>$554</td>
<td>656</td>
<td>120</td>
</tr>
<tr>
<td>$5AF</td>
<td>615</td>
<td>128</td>
</tr>
<tr>
<td>$60A</td>
<td>579</td>
<td>136</td>
</tr>
<tr>
<td>$665</td>
<td>547</td>
<td>144</td>
</tr>
</tbody>
</table>

POKTAB: WORD $3E8

"VCOUNT" INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE CONVERSION TABLE

THE VALUES STORED IN THE TABLE ARE 'AUDF+7'.

THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:

\[
F_{\text{OUT}} = F_{\text{IN}} / (2 \times (\text{AUDF+M})) , \quad \text{WHERE } F_{\text{IN}} = 1.78979 \text{ MHZ.} \quad \& \quad M = 7
\]

FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY AN INTERVAL OF THE ‘VCOUNT’ TIMER:

\[
\text{AUDF+7} = (11.365167) \times T_{\text{OUT}}, \quad \text{WHERE } T_{\text{OUT}} = \# \text{ OF COUNTS (127 USEC.RESOLUTION) OF ‘VCOUNT’ FOR 1 CHARACTER TIME (10 BIT TIMES).}
\]

<table>
<thead>
<tr>
<th>AUDF+7</th>
<th>BAUD RATE</th>
<th>VCOUNT INTERVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>$27C</td>
<td>1407</td>
<td>96</td>
</tr>
<tr>
<td>$2D7</td>
<td>1231</td>
<td>64</td>
</tr>
<tr>
<td>$332</td>
<td>1094</td>
<td>72</td>
</tr>
<tr>
<td>$38D</td>
<td>985</td>
<td>90</td>
</tr>
<tr>
<td>$3E8</td>
<td>895</td>
<td>88</td>
</tr>
<tr>
<td>$443</td>
<td>820</td>
<td>96</td>
</tr>
<tr>
<td>$49E</td>
<td>757</td>
<td>104</td>
</tr>
<tr>
<td>$4F9</td>
<td>703</td>
<td>112</td>
</tr>
<tr>
<td>$554</td>
<td>656</td>
<td>120</td>
</tr>
<tr>
<td>$5AF</td>
<td>615</td>
<td>128</td>
</tr>
<tr>
<td>$60A</td>
<td>579</td>
<td>136</td>
</tr>
<tr>
<td>$665</td>
<td>547</td>
<td>144</td>
</tr>
</tbody>
</table>

POKTAB: WORD $3E8

"VCOUNT" INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE CONVERSION TABLE

THE VALUES STORED IN THE TABLE ARE 'AUDF+7'.

THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:

\[
F_{\text{OUT}} = F_{\text{IN}} / (2 \times (\text{AUDF+M})) , \quad \text{WHERE } F_{\text{IN}} = 1.78979 \text{ MHZ.} \quad \& \quad M = 7
\]

FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY AN INTERVAL OF THE ‘VCOUNT’ TIMER:

\[
\text{AUDF+7} = (11.365167) \times T_{\text{OUT}}, \quad \text{WHERE } T_{\text{OUT}} = \# \text{ OF COUNTS (127 USEC.RESOLUTION) OF ‘VCOUNT’ FOR 1 CHARACTER TIME (10 BIT TIMES).}
\]
ERR LINE  ADDR  B1  B2  B3  B4  SIO (SERIAL BUS INPUT/OUTPUT CONTROLLER)  PAGE  60

<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>WORD</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2707</td>
<td>EDE0</td>
<td>C0</td>
<td>06</td>
<td></td>
<td></td>
<td>$6C0</td>
<td>518</td>
</tr>
<tr>
<td>2708</td>
<td>EDE2</td>
<td>1A</td>
<td>07</td>
<td></td>
<td></td>
<td>$71A</td>
<td>492</td>
</tr>
<tr>
<td>2709</td>
<td>EDE4</td>
<td>75</td>
<td>07</td>
<td></td>
<td></td>
<td>$775</td>
<td>469</td>
</tr>
<tr>
<td>2710</td>
<td>EDE6</td>
<td>D0</td>
<td>07</td>
<td></td>
<td></td>
<td>$7D0</td>
<td>447</td>
</tr>
<tr>
<td>2711</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$826</td>
<td>428</td>
</tr>
<tr>
<td>2712</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$8B1</td>
<td>410</td>
</tr>
<tr>
<td>2713</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$93C</td>
<td>394</td>
</tr>
<tr>
<td>2714</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$997</td>
<td>379</td>
</tr>
<tr>
<td>2715</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$9F2</td>
<td>365</td>
</tr>
<tr>
<td>2716</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$A4D</td>
<td>352</td>
</tr>
<tr>
<td>2717</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$AA8</td>
<td>339</td>
</tr>
<tr>
<td>2718</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$B03</td>
<td>328</td>
</tr>
<tr>
<td>2719</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$B03</td>
<td>319</td>
</tr>
<tr>
<td>2720</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>192</td>
</tr>
<tr>
<td>2721</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>176</td>
</tr>
<tr>
<td>2722</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>160</td>
</tr>
<tr>
<td>2723</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>152</td>
</tr>
<tr>
<td>2724</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>168</td>
</tr>
<tr>
<td>2725</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>184</td>
</tr>
<tr>
<td>2726</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>192</td>
</tr>
<tr>
<td>2727</td>
<td>0014</td>
<td>02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>208</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>216</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>224</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>232</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>240</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>248</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256</td>
</tr>
</tbody>
</table>

2724  EDE8  CRNTP3  *=14
2725  0014  02  SIOSPR: BYTE DSKORG-CRNTP3 ;^GSIOL IS TOO LONG
.TITLE 'DISK ***** DISKP.SRC ***** 3/9/79 ***** 4:00:00 P.M.'

STATVH = DVSTAT/256
STATVL = (-256)*STATVH+DVSTAT ;STATUS POINTER

CONSTANT EQUATES

DISKID = $31 ;SERIAL BUS DISK I.D.
PUTSEC = $50 ;DISK PUT SECTOR DCB COMMAND
READ = $52 ;DISK GET SECTOR DCB COMMAND
WRITE = $57 ;DISK PUT SECTOR WITH READ CHECK DCB COMMAND
STATC = $53 ;DISK STATUS DCB COMMAND
FOMAT = $21 ;DISK FORMAT DCB COMMAND ******
NODAT = 0 ;SIO COMMAND FOR "NO DATA" OPERATION
GETDAT = $40 ;SIO COMMAND FOR "DATA FROM DEVICE"
PUTDAT = $80 ;SIO COMMAND FOR "DATA TO DEVICE"

VECTORS

**$E450

JMP DINIT ;DISK INIT. VECTOR

JMP DSKIF ;DISK INTERFACE ENTRY POINT

CONSTANTS

**DSKORG

;***************************************************************************************

;DISK INTERFACE ROUTINE STARTS HERE

;***************************************************************************************
DISK INTERFACE INITIALIZATION ROUTINE

DISK INTERFACE ENTRY POINT

DISK INTERFACE ENTRY POINT
ERR LINE ADDR B1 B2 B3 B4 DISK ***** DISKP.SRC ***** 3/9/79 ***** 4:00:00 PAGE 63

2836 EE4F C8 TWICE: INY
2837 EE50 C8 INY
2838 EE51 B1 15 RDBAD: LDA (BUFADR), Y ; READ LD BYTE BAD SECTOR DATA
2839 EE53 C9 FF CMP #$FF
2840 EE55 D0 F8 BNE TWICE ; IS IT "FF" ?
2841 EE57 C8 INY
2842 EE58 B1 15 LDA (BUFADR), Y ; READ HI BYTE BAD SECTOR DATA
2843 EE5A C8 INY
2844 EE5B C9 FF CMP #$FF
2845 EE5D D0 F2 BNE RDBAD ; IS IT "FF" ?
2846 EE5F 88 DEY
2847 EE60 88 DEY ; YES,
2848 EE61 BC 08 03 STY DBYTHO ; PUT BAD SECTOR BYTE COUNT INTO DCB
2849 EE64 A9 00 LDA #$0
2850 EE66 8D 09 03 STA DBYTHI
2851 EE69 AC 03 03 ENDDIF: LDY DSTATS
2852 EE6C 60 RTS
2853
2854
2855
2856
2857 ; S U B R O U T I N E S
2858
2859
2860
2861
2862 EE6D AD 04 03 PUTADR: LDA DBUFLO
2863 EE70 B5 15 STA BUFADR
2864 EE72 AD 05 03 LDA DBUFHI
2865 EE75 B5 16 STA BUFADR+1 ; PUT BUFFER ADDR IN TEMP REG
2866 EE77 60 RTS
2867 ;******************************************************************************
2868
2869
2870 ; SPARE BYTE OR MODULE TOO LONG FLAG
2871
2872 EE7B CRNTP4 = *
2873
2874 **$14
2875 0014 00 DSKSPR: .BYTE PRNORG-CRNTP4 ; DISKP TOO LONG
2876
DEVICE NUMBER OR CODE EQUATES

OPNOUT = $2 ; IOC B OPEN FOR OUTPUT COMMAND
NBUFSZ = 40 ; PRINT NORMAL BUFFER SIZE
DBUFSZ = 20 ; PRINT DOUBLE BUFFER SIZE
PDEVN = $40 ; PRINTER DEVICE NUMBER
STATC = $53 ; DCB STATUS COMMAND CODE
WRITEC = $57 ; DCB WRITE COMMAND
SPACE = $20 ; ASCII SPACE CHAR.
N = $4E ; ASCII "N" CHAR.
D = $44 ; ASCII "D" CHAR.
S = $53 ; ASCII "S" CHAR.

**=$E30

PHOPEN-1 ; PRINTER HANDLER OPEN
PHCLOS-1 ; PH CLOSE
BADST-1 ; PH READ
PHPRINT-1 ; PH WRITE
PHSTAT-1 ; PH STATUS
BADST-1 ; PH SPECIAL
PHINIT ; PH INIT.
0 ; ROM FILLER

**=PRNORG

PRINTER HANDLER INITIALIZATION ROUTINE
ERR LINE  ADDR  B1  B2  B3  B4  PRINTER  *****  PRINTP.SRC  *****  3/9/79  *****  4:00  PAGE  65

2931
2932  EE7B  A9  1E  PHINIT:  LDA  #30
2933  EE7A  85  1C  STA  PTIMOT  ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
2934  EE7C  60  RTS
2935
2936
2937
2938
2939  EE7D  EA  02  ; PRINTER HANDLER CONSTANTS
2940  EE7F  CO  03
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954  EE81  A9  04  ; PRINTER HANDLER STATUS ROUTINE
2955  EE83  85  1E  PHSTAT:  LDA  #4
2956  EE85  AE  7D  EE  STA  PBUSZ  ;SET BUFFER SIZE TO 4 BYTES
2957  EE86  AE  7E  EE  LDX  PHSTLO
2958  EE87  AC  53  LDY  PHSTLO+1  ;SET POINTER TO STATUS BUFFER
2959  EE88  BD  02  03  LDA  #STATC  ;SET COMMAND TO "STATUS"
2960  EE89  BD  0A  03  STA  DCOMND  ;SET STATUS COMMAND
2961  EE8A  80  0A  03  STA  DAUX1
2962  EE8B  20  EE  JSR  SETDCB  ;00 SETUP DCB
2963  EE8C  20  59  EE  JSR  SIOV  ;SEND STATUS COMMAND
2964  EE8D  30  03  BMI  BADST  ;GO IF ERROR
2965  EE8E  20  14  EF  JSR  PHPUT  ;YES, PUT STATUS INTO GLOBAL BUFFER.
2966  EE90  A9  00  BADST:  RTS
2967
2968
2969
2970  EE9F  20  B1  EE  PHOPEN:  JSR  PHSTAT  ;DO STATUS COMMAND TO SIO
2971
2972  EEA2  A9  00  LDA  #0
2973  EEA4  85  1D  STA  PBUNTS  ;CLEAR PRINT BUFFER POINTER
2974  EEA6  60  RTS
2975
2976
2977
2978
2979
2980
2981
2982  EEA7  85  1F  PHWRIT:  STA  PTEMP  ;SAVE ACCUM
2983  EEA9  20  1A  EF  JSR  PRMODE  ;GO DETERMINE PRINT MODE
2984  EEAC  A6  1D  LDX  PBUNTS

**********

ERR LINE  ADDR  B1 B2 B3 B4  PRINTER ***** PRINTP. SRC ***** 3/9/79 *****  4:00   PAGE  66

2985  EEA5  A5 1F   LDA PTEMP ; GET CHAR. SENT BY CIO
2986  EEB0  9D C0 03  STA PRNBUF,X ; PUT CHAR. IN PRINT BUFFER
2987  EE03  EB   INX ; INCR. BUFFER POINTER
2988  EEB4  E4 1E   CPX PBUFSZ ; BUFFER POINTER=BUFFER SIZE?
2989  EEB6  F0 13   BEG BUFFUL
2990  EEBB  B6 1D   STX PBPTNT ; SAVE BUFFER POINTER
2991  EEBA  C9 98   CMP #CR ; IS CHAR. = EOL ?
2992  EEBC  F0 03   BEG BLFILL ; IF YES, GO DO BLANK FILL.
2993  EEBE  A0 01   LDY #SUCCES ; PUT GOOD STATUS IN Y REG FOR CIO.
2994  EEC0  60   RTS
2995  EEC1  A9 20   BLFILL: LDA #SPACE ; PUT BLANK IN ACCUM.
2996  EEC3  9D C0 03  FillBF: STA PRNBUF,X ; STORE IT IN PRINT BUFFER.
2997  EEC6  EB   INX
2998  EEC7  E4 1E   CPX PBUFSZ
2999  EEC9  D0 F8   INCR. BUFFER POINTER
3000  EECB  A9 00   BUFFUL: LDA #0
3001  EECB  85 1D   STA PBPTNT ; CLEAR PRINT BUFFER POINTER
3002  EECF  AE 7F EE   LDX PHCHLO
3003  EED2  AC 80 EE   LDY PHCHLO+1 ; SET POINTER TO PRINT BUFFER
3004  EED5  20 E6 EE   JSR SETDCB ; GO SETUP DCB
3005  EED8  20 59 E4   JSR SIOV ; SEND PRINT COMMAND
3006  EEBE  60   RTS
3007
3008
3009
3010
3011
3012
3013  EEDC  20 1A EF   PHCLOS: JSR PRMODE ; GO DETERMINE PRINT MODE
3014  EEDF  A6 1D   LDX PBPTNT
3015  EEE1  D0 DE   BNE BLFILL
3016  EEE3  A0 01   LDY #SUCCES
3017  EEE5  60   RTS
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034  EEE6  BE 04 03  SETDCB: STX DBUFLO
3035  EEE9  BC 05 03  STY DBUFHI ; SET BUFFER POINTER
3036  EEEC  A9 40   LDA #PDEVN
3037  EEED  8D 00 03   STA DDEVIC ; SET PRINTER BUS I.D. FOR DCB
3038  EEF1  A9 01   LDA #1

SUBROUTINES

SET UP DCB TO CALL SIO
ERR LINE ADDR B1 B2 B3 B4 PRINTER ***** PRINTP.SRC ***** 3/9/79 ***** 4:00 PAGE 67

<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>PRINTER</th>
<th>***** PRINTP.SRC ***** 3/9/79 ***** 4:00 PAGE 67</th>
</tr>
</thead>
</table>

```
3039 EEF3 BD 01 03 STA DUNIT ; SET UNIT NUMBER TO 1
3040 EEF6 A9 80 LDA #$BO ; DEVICE WILL EXPECT DATA
3041 EEF8 AE 02 03 LDX DCOMND ; STATUS COMMAND?
3042 EEF8 E0 53 CPX #$50 ; EXPECT DATA FROM DEVICE
3043 EEFD D0 02 BNE PSIOC ; SET SID MODE COMMAND.
3044 EEF7 A9 40 LDA #$40 ; EXPECT DATA FROM DEVICE
3045 EF01 BD 03 03 PSIOC: STA DSTATS ; SET SID MODE COMMAND.
3046 EF04 A5 1E LDA PBUSZ
3047 EF06 BD 08 03 STA DBYTL0 ; SET LO BYTE COUNT
3048 EF09 A9 00 LDA 0
3049 EF0B BD 09 03 STA DBYTHI ; SET HI BYTE COUNT
3050 EF0E A5 1C LDA PTimot
3051 EF10 BD 06 03 STA DTIMLO ; SET DEVICE TIMEOUT COUNT
3052 EF13 60 RTS
3053
3054
3055
3056
3057
3058
3059 EF14 AD EC 02 PHPUT: LDA DVSTAT+2 ; GET DEVICE TIMEOUT FROM STATUS & SAVE IT
3060 EF17 BC 1C STA PTIMOT ; SAVE DEVICE TIMEOUT
3061 EF19 60 RTS
3062
3063
3064
3065
3066
3067
3068
3069 EF1A A0 57 PRMODE: LDY #WRITEC ; PUT WRITE COMMAND IN Y REG
3070 EF1C A5 2B LDA ICAX2Z ; READ PRINT MODE
3071 EF1E C9 4E CMODE: CMP #N ; PRINT NORMAL?
3072 EF20 D0 04 BNE CDUBL ; PRINT NORMAL?
3073 EF22 A2 2B LDX #NBUSIZ ; YES, SET NORMAL CHAR. BUFFER SIZE
3074 EF24 D0 0E BNE SETBSZ ; SIDEWAYS?
3075 EF26 C9 44 CDUBL: CMP #D ; PRINT DOUBLE?
3076 EF28 D0 04 BNE CSIDE ; PRINT NORMAL?
3077 EF2A A2 14 LDX #DBUSIZ ; YES, SET DOUBLE CHAR. BUFFER SIZE
3078 EF2C D0 06 BNE SETBSZ ; SIDEWAYS?
3079 EF2E C9 53 CSIDE: CMP #S ; PRINT DOUBLE?
3080 EF30 D0 08 BNE GOERR ; IF NOT, GO TO ERROR ROUTINE
3081 EF32 A2 1D LDX #SBUSIZ ; YES, SET SIDEWAYS BUFFER SIZE
3082 EF34 B6 1E SETBSZ: STX PBUSIZ ; STORE PRINT BUFFER SIZE
3083 EF36 BC 02 03 STY DCOMND ; STORE DCB COMMAND
3084 EF39 BD 0A 03 STA DAUXI ; STORE DCB AUXI PRINT MODE
3085 EF3C 60 RTS
3086 EF3D A9 4E GOERR: LDA #N ; SET DEFAULT PRINT MODE TO NORMAL
3087 EF3F D0 DD BNE CMODE
3088
3089
3090
3091
3092
```

; SPARE BYTE OR MODULE TOO LONG FLAG
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>PRINTER</th>
<th>3/9/79</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3093</td>
<td>EF41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRNTP5  = *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3094</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>**=14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3095</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3096</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3097</td>
<td>0014</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td>PRNSPR: .BYTE CASORQ-CRNTP5 ; ^GPRINTP TOO LONG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3098</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
.PAGE
.TITLE 'CASSET HANDLER 3/12 (DK1:CASCV)'

CBUFH = CASBUF/256
CUBL = (-256)*CBUFH+CASBUF
SRSTA = $40 :SIO READ STATUS
SWSTA = $80 :SIO WRITE STATUS
MOTRGO = $34
MOTRST = $3C

OOFD
CBUFL
(-256)*CBUFH+CASBUF

SRSTA
$40
iSIO
READ
STATUS

SWSTA
$80
iSIO
WRITE
STATUS

MOTRGO
$34

MOTRST
$3C

DTA = $FC :DATA RECORD TYPE BYTE
DTI = $FA :LAST DATA RECORD
EDT = $FE :END OF TAPE
HDR = $FB :HEADER
TONE1 = 2 :CHANGE TO RECORD MODE TONE
TONE2 = 1 :PRESS PLAY TONE

OOFB
HDR $FB

0002 TONE1 = 2 iCHANGE TO RECORD MODE TONE
0001 TONE2 = 1 iPRESS PLAY TONE

*==CASETV

WORD OPENC-1,CLOSEC-1,GBYTE-1,PBYTE-1,STATU-1,SPECIAL-1

BYTE 0 iROM FILLER BYTE

*==RBLOKV

JMP RBLOK

*==CSOPIV

JMP OPINP

*==CASORG

INIT ROUTINE

INIT: LDA #$CC

STA CBAUOL
LOA #$05 STA CBAUDH

SET CASSETTE BAUD RATE TO 600

THATS ALL FOLKS

SPECIAL: RTS

USED IN MONITP FOR CASSETTE BOOT

*==RBLOKV

JMP RBLOK

*==CSOPIV

JMP OPINP

*==CASORG

INIT ROUTINE

INIT: LDA #$CC

STA CBAUOL
LOA #$05 STA CBAUDH

SET CASSETTE BAUD RATE TO 600

THATS ALL FOLKS

SPECIAL: RTS
OPEN FUNCTION - WITH NO TIMING ADJUST

OPENC: LDA ICAX2Z ; GET AX2
        STA FTYPE ; SAVE IT FOR FUTURE REFERENCE
        LDA ICAX1Z
        AND #$0C ; IN AND OUT BITS
        CMP #$04
        BEQ OPINP ; SEE IF OPEN FOR OUTPUT
        CMP #$08
        BEQ OPOUT
        RTS ; IF ALREADY OPEN, RETURN LEAVING STATUS=$84

OPINP: LDA #0
        STA WMODE ; SET READ MODE
        STA FEOF ; NO EOF YET

SFH: LDA #TONE2 ; TONE FOR PRESS PLAY
        JSR BEEP ; GO BEEP
        BMI OPNRTN ; IF ERROR DURING BEEP
        LDA #MOTRGO
        STA PACTL ; TURN MOTOR ON

        .IF PALFLG
        LDY #$EO
        LDX #1
        .ENDIF

        .IF PALFLG-1
        LDY #$40 ; 5-31-79 9 SEC READ LEADER
        LDX #2
        .ENDIF

WAITM: LDA CDTMF3
        STA CDTMF3
        BNE WAITTM ; WAIT FOR MOTOR TO COME UP TO SPEED

        .IF PRK ABRT
        LDY #BRKABT ; BREAK KEY ABORT STATUS
        DEC BRKKEY ; RESET BREAK KEY
        ORA #0
        STA BMODE ; CLEAR WRITE MODE FLAG
        RTS ; AND EXIT.

PBKR: LDY #BRKABT ; BREAK KEY ABORT STATUS
        DEC BRKKEY ; RESET BREAK KEY
        ORA #0 ; CLEAR WRITE MODE FLAG
        RTS ; AND EXIT.

OPNRTN: LDA #0 ; CLEAR WRITE MODE FLAG
        STA WMODE ; SET WRITE MODE
        RTS ; AND EXIT.

OPEN FOR OUTPUT

OPOUT: LDA #$80 ; SET WRITE MODE
        STA WMODE ; SET WRITE MODE
        RTS ; AND EXIT.

EF4C A5 2B
EF4E B5 3E
EF50 A5 2A
EF52 29 0C
EF54 C9 04
EF56 F0 05
EF58 C9 08
EF5A F0 39
EF5C 60
EF5D A9 00
EF5F 8D B9 02
EF62 85 3F
EF64 A9 01
EF66 20 58 F0
EF69 30 24
EF6B A9 34
EF6D 8D 02 D3
EF70 A0 40
EF72 A2 02
EF74 A9 03
EF76 8D 2A 02
EF79 20 5C E4
EF7C AD 2A 02
EF7F D0 F8
EF81 A9 80
EF83 85 3D
EF85 8D 8A 02
EF88 4C D3 EF
EF8A A0 80
EF8B C6 11
EF8D A9 00
EF91 8D B9 02
EF94 60
EF95 A9 80
EF97 8D B9 02
EF9A A9 02
EF9C 20 5B F0
EF9F 30 EE
EFA1 A9 CC
EFA3 8D 04 D2
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>CASSET HANDLER 3/12 (DK1:CASCV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3201</td>
<td>EFA6</td>
<td>A9</td>
<td>05</td>
<td></td>
<td></td>
<td>LDA #$05 ; FOR SOME OBSCURE REASON</td>
</tr>
<tr>
<td>3202</td>
<td>EFA8</td>
<td>BD</td>
<td>06</td>
<td>D2</td>
<td></td>
<td>STA AUDF4</td>
</tr>
<tr>
<td>3203</td>
<td>EFA8</td>
<td>A9</td>
<td>60</td>
<td></td>
<td></td>
<td>LDA #$60</td>
</tr>
<tr>
<td>3204</td>
<td>EFA9</td>
<td>BD</td>
<td>00</td>
<td>03</td>
<td></td>
<td>STA DDEVIC</td>
</tr>
<tr>
<td>3205</td>
<td>EFBO</td>
<td>20</td>
<td>08</td>
<td>E4</td>
<td></td>
<td>JSR SENDEV ; TELL POKEY TO WRITE MARKS</td>
</tr>
<tr>
<td>3206</td>
<td>EFBB</td>
<td>A9</td>
<td>34</td>
<td></td>
<td></td>
<td>LDA #MOTRGO ; WRITE 5 SEC BLANK TAPE</td>
</tr>
<tr>
<td>3207</td>
<td>EFBD</td>
<td>BD</td>
<td>02</td>
<td>D3</td>
<td></td>
<td>STA PACTL</td>
</tr>
<tr>
<td>3208</td>
<td>EFBE</td>
<td>A9</td>
<td>03</td>
<td></td>
<td></td>
<td>LDA #3</td>
</tr>
<tr>
<td>3209</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.IF PALFLG</td>
</tr>
<tr>
<td>3210</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LDX #$3</td>
</tr>
<tr>
<td>3211</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LDY #$CO</td>
</tr>
<tr>
<td>3212</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.ENDIF</td>
</tr>
<tr>
<td>3213</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.IF PALFLG-1</td>
</tr>
<tr>
<td>3214</td>
<td>EFBA</td>
<td>A2</td>
<td>04</td>
<td></td>
<td></td>
<td>LDX #4 ; 5/30/79 20 SEC LEADER</td>
</tr>
<tr>
<td>3215</td>
<td>EFBC</td>
<td>A0</td>
<td>80</td>
<td></td>
<td></td>
<td>LDY #$80</td>
</tr>
<tr>
<td>3216</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.ENDIF</td>
</tr>
<tr>
<td>3217</td>
<td>EFBE</td>
<td>20</td>
<td>5C</td>
<td>E4</td>
<td></td>
<td>JSR SETVBV</td>
</tr>
<tr>
<td>3218</td>
<td>EFCA</td>
<td>A9</td>
<td>FF</td>
<td></td>
<td></td>
<td>LDA #$FF</td>
</tr>
<tr>
<td>3219</td>
<td>EFC3</td>
<td>BD</td>
<td>2A</td>
<td>02</td>
<td></td>
<td>STA CDMF3</td>
</tr>
<tr>
<td>3220</td>
<td>EFC6</td>
<td>A8</td>
<td>11</td>
<td></td>
<td></td>
<td>WDLR: LDA BRKKEY</td>
</tr>
<tr>
<td>3221</td>
<td>EFC8</td>
<td>F0</td>
<td>C1</td>
<td></td>
<td></td>
<td>BEQ PBRK ; IF BREAK DURING WRITE LEADER</td>
</tr>
<tr>
<td>3222</td>
<td>EFCA</td>
<td>AD</td>
<td>2A</td>
<td>02</td>
<td></td>
<td>LDA CDMF3</td>
</tr>
<tr>
<td>3223</td>
<td>EFCD</td>
<td>D0</td>
<td>F7</td>
<td></td>
<td></td>
<td>BNE WDLR</td>
</tr>
<tr>
<td>3224</td>
<td>EFCF</td>
<td>A9</td>
<td>00</td>
<td></td>
<td></td>
<td>LDA #0 ; INIT BUFFER POINTER</td>
</tr>
<tr>
<td>3225</td>
<td>EFD1</td>
<td>BD</td>
<td>3D</td>
<td></td>
<td></td>
<td>STA BPTR</td>
</tr>
<tr>
<td>3226</td>
<td>EFD3</td>
<td>A0</td>
<td>01</td>
<td></td>
<td></td>
<td>OPDK: LDY #$SUCCES</td>
</tr>
<tr>
<td>3227</td>
<td>EFD5</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>RTS</td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4 CASSET HANDLER 3/12 (DK1:CASCV) PAGE 72

3228 PAGE
3229 ; GET BYTE
3230 ;
3231 ;

3232 EFDA A5 3F GBYTE: LDA FEOF ; IF AT EOF ALREADY
3233 EFDB 30 33 BMI ISEOF ; RETURN EOF STATUS
3234 EFDA A6 3D CPX BLIM ; IF END OF BUFFER
3235 EFDC EC BA 02 LDX BPTR ; BUFFER POINTER
3236 EFDF F0 08 BEQ RBLK ; READ ANOTHER BLOCK
3237 EFED BD 00 04 LDA CASBUF+3,X ; GET NEXT BYTE
3238 EFE4 E6 3D INC BPTR ; BUMP POINTER
3239 EFE6 A0 01 LDY #SUCCES ; OK STATUS
3240 EFE8 60 |
3241 EFE9 A9 52 |
3242 EFEF 30 F7 |
3243 EFE9 98 |
3244 EFEF 30 F7 |
3245 EFF1 A9 00 |
3246 EFF3 85 3D |
3247 EFF5 A2 80 |
3248 EFF7 AD FF 03 |
3249 EFFA C9 FE |
3250 EFFE C9 FA |
3251 EFFE C9 FA |
3252 F000 D0 03 |
3253 F002 AE 7F 04 |
3254 F005 BE BA 02 |
3255 F008 4C D6 EF |
3256 F008 4C D6 EF |
3257 F00D A0 88 |
3258 F00F 60 |

3233 ; IF AT EOF ALREADY
3234 ; RETURN EOF STATUS
3235 ; IF END OF BUFFER
3236 ; BUFFER POINTER
3237 ; READ ANOTHER BLOCK
3238 ; GET NEXT BYTE
3239 ; BUMP POINTER
3240 |
3241 |
3242 |
3243 |
3244 |
3245 |
3246 |
3247 |
3248 |
3249 |
3250 |
3251 |
3252 |
3253 |
3254 |
3255 |
3256 |
3257 |
3258 |
; PAGE
; PUT BYTE TO BUFFER
;
; PUT BYTE TO BUFFER
LDA PBYTE
LDX BPTR
STAB CASBUF+3,X
INC BPTR
LDY #SUCCES
BEQ +3
STA CASBUF+3,X
RTEX

; WRITE OUT THE BUFFER
LDA #DTA
JSR WSIOSB
LDA 0
STA BPTR
RTEX
ERR LINE ADDR B1 B2 B3 B4 CASSET HANDLER 3/12 (DK1:CASCV)

3276 .PAGE
3277
3278
3279
3280 F028 A0 01 STATU: LDY #SUCCES
3281 F02A 60 RTS

PAGE 74
ERR LINE ADDR B1 B2 B3 B4 CASSET HANDLER 3/12 (DK1: CASCV)

3282 .PAGE
3283 ;
3284 ; CLOSE
3285 ;
3286 F02B AD 89 02 CLOSEC: LDA WMODE ; SEE IF WRITING
3287 F02E 30 08 BMI CLWRT ; GO CLOSE FOR WRITE
3288 ; CLOSE FOR READ - FLAG CLOSED
3289 F030 A0 01 LDY #SUCCES SUCCESSFUL
3290 F032 A9 3C FCAX: LDA #MOTRST STOP THE MOTOR IN CASE WAS SHORT IRG MODE
3291 F034 8D 02 D3 STA PACTL
3292 F037 60 RTS
3293 F038 A6 3D CLWRT: LDX BPTR BUFFER POINTER
3294 F03A F0 0A BEQ WTLR IF NO DATA BYTES IN BUFFER, NO DT1 REC
3295 F03C BE 7F 04 STX CASBUF+130 WRITE TO LAST RECORD
3296 F03F A9 FA LDA #DT1 REC TYPE
3297 F041 20 D2 F0 STA PACTL
3298 F044 30 EC JSR WSIOSB WRITE OUT USER BUFFER
3299 F046 A2 7F BMI FCAX GO IF ERROR
3300 F048 A9 00 WTLR: LDX #127 ZERO BUFFER
3301 F04A 9D 00 04 LDA #0
3302 F04C CA DEX
3303 F04E 10 FA STA CASBUF+3,X
3304 F050 A9 FE LDA #EOT WRITE EOT RECORD
3305 F052 20 D2 F0 JSR WSIOS8
3306 F055 4C 32 F0 JMP FCAX FLAG CLOSED AND EXIT
ERR LINE ADDR B1 B2 B3 B4 
3307 ; .PAGE
3308 ; ; SUBROUTINES
3309 ; ; BEEP - GENERATE TONE ON KEYBOARD SPEAKER
3310 ; ON ENTRY A= FREG
3311 ;
3312 F05B 85 40 BEEP: STA FREG
3313 F05A A5 14 BEEP1: LDA RTCLK+2 ; CURRENT CLOCK
3314 F05C 18 CLC
3315 .IF PALFLG
3316 .ADC #25
3317 .ENDIF
3318 .IF PALFLG-1
3319 .ADC #30 ; 1 SEC TONE
3320 .ENDIF
3321 F05D 69 1E
3322 F05F AA TAX
3323 F060 A9 FF WFL: LDA #$FF
3324 F062 8D 1F DO STA CONSOL ; TURN ON SPEAKER
3325 F065 A9 00 LDA #0
3326 F067 A0 F0 LDY #$F0
3327 F069 8B DEY
3328 F06A D0 FD BNE *-1
3329 F06C BD 1F DO STA CONSOL ; TURN OFF SPEAKER
3330 F06F A0 F0 LDY #$F0
3331 F071 8B DEY
3332 F072 D0 FD BNE *-1
3333 F074 E4 14 CPX RTCLK+2 ; SEE IF 1 SEC IS UP YET
3334 F076 D0 EB BNE WFL
3335 F078 C6 40 DEC FREQ ; COUNT BEEPS
3336 F07A F0 08 BEG WFAK ; IF ALL DONE GO WAIT FOR KEY
3337 F07C 8A TXA
3338 F07D 1B CLC
3339 .IF PALFLG
3340 .ADC #8
3341 .ENDIF
3342 .IF PALFLG-1
3343 .ADC #10 .ENDIF
3344 F07E 69 0A
3345 F080 AA ; TAX
3346 F081 E4 14 CPX RTCLK+2
3347 F083 D0 FC BNE *-2
3348 F085 F0 D3 BEG BEEP1 ; UNCOND GO BEEP AGAIN
3349 F087 20 8C F0 WFAK: JSR WFAK1 ; USE SIMULATED "JMP (KGETCH)"
3350 F08A 9B TYA
3351 F08B 60 RTS
3352 F08C AD 25 E4 WFAK1: LDA KEYBDV+5
3353 F08F 48 PHA
3354 F090 AD 24 E4 LDA KEYBDV+4 ; SIMULATE "JMP (KGETCH)"
3355 F093 48 PHA
3356 F094 60 RTS
3357 ;
3358 ; SIOSB - CALL SIO ON SYSTEM BUFFER
ERR LINE ADDR B1 B2 B3 B4 CASSET HANDLER 3/12 (DK1: CASCV) PAGE 77

3361  F095  8D  02  03  SIOSB: STA  DCOMND ;SAVE COMMAND
3362  F09B  A9  00  LDA  #0
3363  F09A  8D  09  03  STA  D8YTHI ;SET BUFFER LENGTH
3364  F09D  A9  83  LDA  #131
3365  F09F  8D  08  03  STA  D8YTL0
3366  F0A2  A9  03  LDA  #CBUFL
3367  F0A4  8D  05  03  STA  D8UFHI ;SET BUFFER ADDRESS
3368  F0A7  A9  FD  LDA  #CBUFL
3369  F0A9  8D  04  03  STA  D8UFLO
3370  F0AC  A9  60  CSIO: LDA  #60 ;CASSET PSEUDO DEVICE
3371  F0AE  BD  00  03  STA  DDEVIC
3372  F0B1  A9  00  LDA  #0
3373  F0B3  8D  01  03  STA  DUNIT
3374  F0B6  A9  23  LDA  #35 ;DEVICE TIMEOUT (5/30/79)
3375  F0BB  8D  06  03  STA  D8IMLO
3376  F0BB  AD  02  03  LDA  DCOMND ;GET COMMAND BACK
3377  F0BE  A0  40  LDY  #SRSTA ;SIO READ STATUS COMMAND
3378  F0C0  C9  52  CMP  #R
3379  F0C2  F0  02  BEG  ++4
3380  F0C4  A0  80  LDY  #SWSTA ;SIO WRITE STATUS COMMAND
3381  F0C6  BC  03  03  STY  DSTATS ;SET STATUS FOR SIO
3382  F0C9  A5  3E  LDA  FTYPE
3383  F0CB  8D  08  03  STA  DAUX2 ;INDICATE IF SHORT IRG MODE
3384  F0CE  20  59  E4  JSR  SIOV ;GO CALL SIO
3385  F0D1  60  RTS
3386  ; WSIOSB - WRITE SIO SYSTEM BUFFER
3387  ;
3388  F0D2  8D  FF  03  WSIOSB: STA  CASBUF+2 ;STORE TYPE BYTE
3389  F0D5  A9  55  LDA  #$55
3390  F0D7  8D  FD  03  STA  CASBUF+0
3391  F0DA  8D  FE  03  STA  CASBUF+1
3392  F0DD  A9  57  LDA  #'H ;WRITE
3393  F0DF  20  95  F0  JSR  SIOSB ;CALL SIO ON SYSTEM BUFFER
3394  F0E2  60  RTS  AND ;RETURN
3395  F0E3  CRNTP6 ==
3396  ;***$14
3397  0014  00  CASSPR: .BYTE  MONORO-CRNTP6 ;^GCASCV IS TOO LONG
.TITLE 'MONITOR ***** MONITP.SRC ***** 3/9/79 ***** 4:00:00 P

CONSTANT EQUATES

BUFFH = (CASBUF+3)/256
BUFFL = (-256)*BUFFH+CASBUF+3 ;BUFFER POINTER

THE FOLLOWING EQUATES ARE IN THE CARTRIDGE ADDRESS SPACE.

"B" CARTIDGE ADDR'S ARE B000-9FFF (36K CONFIG. ONLY)
"A" CART. ADDR'S ARE AO00-BFFF (36K CONFIG. ONLY)
"A" CART. ADDR'S ARE BO00-BFFF (48K CONFIG. ONLY)

**BFFA

CARTRIDGE FLAG ACTION DEFINITIONS

BIT ACTION IF SET

SPECIAL -- DON'T POWER-UP, JUST RUN CARTIDGE
NONE
RUN CARTRIDGE
NONE
BOOT DOS
1. IF BIT2 IS O, GOTO BLACKBOARD MODE.
2. IF BIT0 SET, THE DISK WILL BE BOOTED BEFORE ANY OTHER ACTION.

POWER-UP VECTOR

ENTRY POINT VECTOR

BLACK BOARD VECTOR

WARM START VECTOR

COLD START VECTOR (9000 FOR RAM VECTOR WRITING)

TO HANDLE RAM VECTOR WRITING
3507 ; HANDLER TABLE ENTRIES
3508 ;
3509 TBLLEN = IDENT-TBLENT-1 HANDLER TABLE LENGTH. "MOVED TO LINE 8"
3510
3511 IDENT = .BYTE CLS, 'ATARI COMPUTER - MEMO PAD', CR
3512
3513 IDENTH = IDENT/256
3514 IDENTL = (-256)*IDENTH+IDENT ;SYSTEM I.D. MSG POINTER
3515
3516 TBLLEN = IDENT-TBLENT-1 ;HANDLER TABLE LENGTH
3517 DERR5: .BYTE 'BOOT ERROR', CR
3518
3519 DERRH = DERR5/256
3520 DERRL = (-256)*DERRH+DERR5 ;DISK ERROR MSG POINTER
3521
3522 OPNEDT: .BYTE 'E:', CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
3523
3524 OPNH = OPNEDT/256
3525 OPNL = (-256)*OPNH+OPNEDT ;SCREEN EDITOR OPEN POINTER
RESET BUTTON ROUTINE STARTS HERE

; **DISABLE IRQ INTERRUPTS**
LDA COLDST ; WERE WE IN MIDDLE OF COLDSTART?
BNE PWRUP ; YES, GO TRY IT AGAIN
LDA #FF
BNE PWRUP1 ; SET WARM START FLAG

POWER UP ROUTINES START HERE

; **DISABLE IRQ INTERRUPTS**
PWRUP: LDA #0 ; CLEAR WARMSTART FLAG

ZERORM: LDA #0
LDY #WARMST
STA RAMLO ; INITIALIZE RAM POINTER

CLRRAM: STA (RAMLO),Y ; CLEAR MEMORY LOC.
CPY #0 ; AT END OF PAGE?
BNE CLRRAM ; NO.

INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
LDA BLKBDV+1
STA DOSVEC ; USE BLACKBOARD VECTOR

CLEAR OS RAM (FOR WARMSTART)
ZOSRAM: LDX #0

CLEAR PAGES 2 AND 3
ZOSRM2: STA $200,X
STA $300,X

; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
LDA BLKBDV+1
STA DOSVEC ; USE BLACKBOARD VECTOR

CLEAR OS RAM (FOR WARMSTART)
ZOSRAM: LDX #0

CLEAR PAGES 2 AND 3
ZOSRM2: STA $200,X
STA $300,X
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>MONITOR</th>
<th>MONITP.SRC</th>
<th>3/9/79</th>
<th>4:00</th>
<th>PAGE</th>
<th>82</th>
</tr>
</thead>
<tbody>
<tr>
<td>3615</td>
<td>F16A</td>
<td>D0</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
<td>BNE</td>
<td>ZDSRM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3616</td>
<td>F16C</td>
<td>A2</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>LDX</td>
<td>#INTZBS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3617</td>
<td>F16E</td>
<td>95</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td>ZDSRM3:</td>
<td>STA 0, X</td>
<td>CLEAR ZERO PAGE LOCATIONS INTZBS-7F</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3618</td>
<td>F170</td>
<td>E8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3619</td>
<td>F171</td>
<td>10</td>
<td>FB</td>
<td></td>
<td></td>
<td></td>
<td>BPL</td>
<td>ZDSRM3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3620</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3621</td>
<td>F173</td>
<td>A9</td>
<td>02</td>
<td></td>
<td></td>
<td></td>
<td>; ESTABLISH SCREEN MARGINS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3622</td>
<td>F175</td>
<td>B5</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td>ESTSCM:</td>
<td>LDA #LEDGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3623</td>
<td>F177</td>
<td>A9</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td>STA</td>
<td>LMRGN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3624</td>
<td>F179</td>
<td>B5</td>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td>LDA</td>
<td>#REDGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3625</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STA</td>
<td>RMARGN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3626</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3627</td>
<td>F17B</td>
<td>A2</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td>; MOVE VECTOR TABLE FROM ROM TO RAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3628</td>
<td>F17D</td>
<td>BD</td>
<td>80</td>
<td>E4</td>
<td></td>
<td></td>
<td>OPSYS:</td>
<td>LDX #$25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3629</td>
<td>F180</td>
<td>9D</td>
<td>00</td>
<td>02</td>
<td></td>
<td></td>
<td>MOVVEC:</td>
<td>LDA VCTABL, X</td>
<td>ROM TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3630</td>
<td>F183</td>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STA</td>
<td>INTAB, X</td>
<td>TO RAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3631</td>
<td>F184</td>
<td>10</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
<td>DEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3632</td>
<td>F186</td>
<td>20</td>
<td>BA</td>
<td>F2</td>
<td></td>
<td></td>
<td>JSR</td>
<td>OSRAM</td>
<td>DO O.S. RAM SETUP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3633</td>
<td>F189</td>
<td>58</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLI</td>
<td></td>
<td>ENABLE IRQ INTERRUPTS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3634</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3635</td>
<td>F18A</td>
<td>A2</td>
<td>0E</td>
<td></td>
<td></td>
<td></td>
<td>NXTENT:</td>
<td>LDA TBLLEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3636</td>
<td>F18B</td>
<td>BD</td>
<td>E3</td>
<td>F0</td>
<td></td>
<td></td>
<td>STA</td>
<td>HATABS, X</td>
<td>PUT IN TABLE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3637</td>
<td>F18D</td>
<td>9D</td>
<td>1A</td>
<td>03</td>
<td></td>
<td></td>
<td>DEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3638</td>
<td>F192</td>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BPL</td>
<td>NXTENT</td>
<td>DONE WITH ALL ENTRIES?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3639</td>
<td>F193</td>
<td>10</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3640</td>
<td>F195</td>
<td>A2</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td>; INTERROGATE CARTRIDGE ADDR. SPACE TO SEE WHICH CARTRIDGES THERE ARE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3641</td>
<td>F197</td>
<td>B6</td>
<td>07</td>
<td></td>
<td></td>
<td></td>
<td>LDX</td>
<td>#0</td>
<td>CLEAR &quot;B&quot; CART. FLAG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3642</td>
<td>F199</td>
<td>B6</td>
<td>06</td>
<td></td>
<td></td>
<td></td>
<td>STX</td>
<td>TSTDAT</td>
<td>CLEAR &quot;A&quot; CART. FLAG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3643</td>
<td>F19B</td>
<td>AE</td>
<td>E4</td>
<td>02</td>
<td></td>
<td></td>
<td>LDX</td>
<td>RAMSIZ</td>
<td>RAM IN &quot;B&quot; CART. SLOT?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3644</td>
<td>F19E</td>
<td>E0</td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td>CPX</td>
<td>#90</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3645</td>
<td>F1A0</td>
<td>B0</td>
<td>0A</td>
<td></td>
<td></td>
<td></td>
<td>BCS</td>
<td>ENDBCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3646</td>
<td>F1A2</td>
<td>AD</td>
<td>FC</td>
<td>9F</td>
<td></td>
<td></td>
<td>BPL</td>
<td>CART-$2000</td>
<td>NO.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3647</td>
<td>F1A5</td>
<td>DO</td>
<td>05</td>
<td></td>
<td></td>
<td></td>
<td>BNE</td>
<td>CARTR. PLUGGED INTO &quot;B&quot; SLOT?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3648</td>
<td>F1A7</td>
<td>E6</td>
<td>07</td>
<td></td>
<td></td>
<td></td>
<td>INC</td>
<td>TSTDAT</td>
<td>YES. SET &quot;B&quot; CART. FLAG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3649</td>
<td>F1A9</td>
<td>20</td>
<td>3C</td>
<td>F2</td>
<td></td>
<td></td>
<td>JSR</td>
<td>CBINI</td>
<td>INITIALIZE CARTRIDGE &quot;B&quot;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3650</td>
<td>F1AC</td>
<td>AE</td>
<td>E4</td>
<td>02</td>
<td></td>
<td></td>
<td>CPX</td>
<td>#80</td>
<td>RAM IN &quot;A&quot; CART. SLOT?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3651</td>
<td>F1AF</td>
<td>E0</td>
<td>B0</td>
<td></td>
<td></td>
<td></td>
<td>BCS</td>
<td>ENDBCK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3652</td>
<td>F1B1</td>
<td>B0</td>
<td>0A</td>
<td></td>
<td></td>
<td></td>
<td>BCS</td>
<td>ENDACK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3653</td>
<td>F1B3</td>
<td>AE</td>
<td>FC</td>
<td>8F</td>
<td></td>
<td></td>
<td>BDX</td>
<td>CART</td>
<td>NO.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3654</td>
<td>F1B6</td>
<td>D0</td>
<td>05</td>
<td></td>
<td></td>
<td></td>
<td>BNE</td>
<td>CARTR. PLUGGED INTO &quot;A&quot; SLOT?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3655</td>
<td>F1B8</td>
<td>E6</td>
<td>06</td>
<td></td>
<td></td>
<td></td>
<td>INC</td>
<td>TRAMSZ</td>
<td>YES. SET &quot;A&quot; CART. FLAG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
; OPEN SCREEN EDITOR

; ENDACK: LDA \#3

; BPL SCRNOK: BR IF NO ERROR

; JMP PWRUP: RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)

; SCRNOK: INY SCREEN OK, SO WAIT FOR VBLANK TO

; BEG NOCART: NEITHER CARTRIDGE LIVES

; NOA2: AND \#1 DOES EITHER CART WANT BOOT?

; JSR CSBOOT: CHECK, BOOT, AND INIT

; JSR CIOV: GO TO CIO

; DO CASSETTE BOOT

; JSR CSBOOT: CHECK, BOOT, AND INIT

; JSR CIOV: GO TO CIO

; CHECK TO SEE IF EITHER CARTRIDGE WANTS DISK BOOT

; LDA TRAMSZ: CHECK BOTH CARTRIDGES

; ORA TSTDAT: NEITHER CARTRIDGE LIVES

; BEG NOCART: "A" CART?

; LDA TRAMSZ: "B" CART?

; BEG NOB1: ADD OTHER FLAG

; AND \#1 DOES EITHER CART WANT BOOT?

; JSR CARTCS: RUN "A" CARTRIDGE

; JSR CARTCS: RUN "B" CARTRIDGE

; JSR NOCAR2: NO
ERR LINE ADDR B1 B2 B3 B4               MONITOR ***** MONITP.SRC ***** 3/9/79 ***** 4:00  PAGE 84
3723 F216 AD FD 9F                LDA CARTFG-$2000 ;GET "B" MODE FLAG
3724 F219 29 04                AND #4 ;DOES IT WANT TO RUN?
3725 F21B F0 DF                BEQ NOCART ;NO
3726 F21D 6C FA 9F                JMP (CARTCS-$2000) ;RUN "B" CARTRIDGE
3727 ;
3728 ; NO CARTRIDGES, OR NEITHER WANTS TO RUN,
3729 ; SO GO TO DOSVEC (DOS, CASSETTE, OR BLACKBOARD)
3730 F220 6C 0A 00                NOCART2: JMP (DOSVEC)
3731 ;
3732 ; PRINT SIGN-ON MESSAGE
3733 F223 A2 F2                SIGNON: LDX #IDENTL
3734 F225 A0 F0                LDY #IDENTH
3735 F227 20 85 F3                JSR PUTLIN ;GO PUT SIGN-ON MSG ON SCREEN
3736 ;
3737 ;
3738 ;
3739 ; BLACKBOARD ROUTINE
3740 F22A 20 30 F2                BLACKB: JSR BLKB2 ;"JSR EGETCH"
3741 F22D 4C 2A F2                JMP BLACKB ;FOREVER
3742 F230 AD 05 E4                BLKB2: LDA EDITRV+5 ;HIGH BYTE
3743 F233 48                PHA
3744 F234 AD 04 E4                LDA EDITRV+4 ;LOW BYTE
3745 F237 48                PHA
3746 F238 60                RTS ;SIMULATES "JMP (EDITRV)"
3747 ;
3748 ;
3749 ; CARTRIDGE INITIALIZATION INDIRECT JUMPS
3750 F239 6C FE BF                CAINI: JMP (CARTAD)
3751 F23C 6C FE 9F                CBINI: JMP (CARTAD-$2000)
SUBROUTINES

; CHECK FOR HOW MUCH RAM & SPECIAL CARTRIDGE CASE.
; IF SPECIAL CARTRIDGE CASE, DON'T GO BACK -- GO TO CART.

SPECL:  LDA CART ; CHECK FOR RAM OR CART
BNE ENSPE2 ; GO IF NOTHING OR MAYBE RAM
INC CART ; NOW DO RAM CHECK
LDA CART ; IS IT ROM?
BNE ENSPEC ; NO
LDA CARTFG ; YES,
BPL ENSPEC ; BIT SET?
JMP (CARTAD) ; YES, GO RUN CARTRIDGE

; CHECK FOR AMOUNT OF RAM

ENSPEC: DEC CART ; RESTORE RAM IF NEEDED
ENSPEC2: LDY #0
STY RAMLO+1
LDA #$10
STA TRAMSZ ; SET RAM POINTER TO 4K.

HOWMCH: LDA (RAMLO+1),Y ; READ RAM LOCATION
EOR #$FF ; INVERT IT.
STA (RAMLO+1),Y ; WRITE INVERTED DATA.
CMP (RAMLO+1),Y ; READ RAM AGAIN
BNE ENDRAM
EOR #$FF ; CONVERT IT BACK
STA (RAMLO+1),Y ; RESTORE ORIGINAL RAM DATA
LDA TRAMSZ
ERR LINE ADDR B1 B2 B3 B4 MONITOR ***** MONITP.SRC ***** 3/9/79 ***** 4:00 PAGE B6

3806 F26F 18 CLC
3807 F270 69 10 ADC #10
3808 F272 85 06 STA TRAMSZ ; INCR. RAM POINTER BY 4K.
3809 F274 D0 E9 BNE HOWMCH ;GO FIND HOW MUCH RAM.
3810 F276 60 ENDRAM: RTS
3811
3812
3813
3814
3815
3816
3817
3818 F277 A9 00 HARDI: LDA #0
3819 F279 AA Tax
3820 F27A 9D 00 D0 CLRCHP: STA $D000, X
3821 F27D 9D 00 D4 STA $D400, X
3822 F280 9D 00 D2 STA $D200, X
3823 F283 9D 00 D3 STA $D300, X
3824 F286 E8 INX
3825 F287 D0 F1 BNE CLRCHP
3826 F289 60 RTS
3827
3828
3829
3830
3831 F28A C6 11 OSRAM: DEC BRKKEY ; TURN OFF BREAK KEY FLAG
3832 F28C A9 54 LDA #.LOW.BRKKY2
3833 F28E 8D 36 02 STA BRKKEY
3834 F291 A9 E7 LDA #.HIGH.BRKKY2
3835 F293 8D 37 02 STA BRKKY+1
3836 F296 A5 06 LDA TRAMSZ ; READ RAM SIZE IN TEMP. REG.
3837 F298 8D E4 02 STA RAMSZ ; SAVE IT IN RAM SIZE.
3838 F29B 8D E6 02 STA MEMTOP+1 ; INIT. MEMTOP ADDR HI BYTE
3839 F29E A9 00 LDA #0
3840 F2A0 8D E5 02 STA MEMTOP ; INIT. MEMTOP ADDR LO BYTE
3841 F2A3 A9 00 LDA #INIMLL
3842 F2A5 8D E7 02 STA MEMLO
3843 F2A8 A9 07 LDA #INIMLH
3844 F2AA 8D E8 02 STA MEMLO+1 ; INITIALIZE MEMLO ADDR vector
3845 F2AD 20 0C E4 JSR EDITRV+**C ; EDITOR INIT.
3846 F2B0 20 1C E4 JSR SCREENV+**C ; SCREEN INIT.
3847 F2B2 20 2C E4 JSR KEYBDV+**C ; KEYBOARD INIT.
3848 F2B6 20 3C E4 JSR PRINTV+**C ; PRINTER HANDLER INIT
3849 F2B9 20 4C E4 JSR CASETV+**C ; CASSETTE HANDLER INIT
3850 F2BC 20 6E E4 JSR CIDINV ; CIO INIT.
3851 F2BF 20 65 E4 JSR SIDINV ; SIO INIT.
3852 F2C2 20 6B E4 JSR INTINV ; INTERRUPT HANDLER INIT.
3853 F2C5 AD 1F D0 LDA CONSOL
3854 F2CB 29 01 AND #1
3855 F2CA D0 02 BNE NOKEY ; GAME START KEY DEPRESSED?
3856 F2C2 E6 4A INC CKEY ; YES, SET KEY FLAG.
3857 F2CE 60 NOKEY: RTS
3858
3859
3860
; DO BOOT OF DISK

LDA WARMST
BEG NOWARM
; WARM START?

LDA BOOT?
BEG NOINIT
; VALID BOOT?

JSR DINI
; YES, RE-INIT. DOS SOFTWARE

NOINIT: RTS
NOWARM: LDA #1

STA DUNIT
; ASSIGN DISK DRIVE NO.

LDA #STATC
STA DCOMND
; SET UP STATUS COMMAND

JSR DSKINV
; DO DISK STATUS

BPL DOBOOT
; IS STATUS FROM SIG GOOD?

RTS
; NO, GO BACK WITH BAD BOOT STATUS

DOBOOT: LDA #0

STA DAUX2

LDA #1

STA DAUX1
; SET SECTOR # TO 1.

LDA #BUFFL

STA DBUFL0

LDA #BUFFH

STA DBUFHI

SECT1: JSR GETSEC
; SET SECTOR

BPL ALLSEC
; STATUS O.K.?

JSR DSKRDE
; NO, GO PRINT DISK READ ERROR

LDA CASSIB

BEQ DOBOOT
; CASSETTE BOOT?

RTS
; YES, QUIT

ALLSEC: LDX #3

RDBYTE: LDA CASBUF+3,X
; READ A BUFFER BYTE

STA DFLAGS,X
; STORE IT

F316 CA

DEX

BPL RDBYTE
; DONE WITH 4 BYTE TRANSFER?

LDA BOOTAD
; YES.

STA RAMLO

LDA BOOTAD+1

STA RAMLO+1
; PUT BOOT ADDR INTO Z. PAGE RAM

LDA CASBUF+7

STA DOSINI7
; ESTABLISH DOS INIT ADDRESS

LDA CASBUF+8

STA DOSINI+1

LDA DOSINI+1

MVBUFF: LDY #87F
; YES, SET BYTE COUNT

LDA CASBUF+3,Y
; MOVE A BYTE FROM SECTOR BUFFER TO BOOT ADDR

MVNXB: STA (RAMLO)+Y

DEY

LDA RAMLO

BPL MVNXB
; DONE?

CLC

clc

LDA RAMLO

ADC #80

STA RAMLO

STA RAMLO

LDA RAMLO+1
3914 F340 69 00
3915 F342 85 05
3916 F344 CE 41 02
3917 F347 F0 11
3918 F349 EE 0A 03
3919 F34C 20 9D F3
3920 F34F 10 DC
3921 F351 20 B1 F3
3922 F354 A5 48
3923 F356 D0 AE
3924 F358 F0 F2
3925 F35A A5 4B
3926 F35C F0 03
3927 F35E 20 9D F3
3928 F361 20 6C F3
3929 F364 B0 A0
3930 F366 20 7E F3
3931 F369 E6 09
3932 F36B 60
3933 F36C 1B
3934 F36D AD 42 02
3935 F370 69 06
3936 F372 85 04
3937 F374 AD 43 02
3938 F377 69 00
3939 F379 B5 05
3940 F37B 6C 04 00
3941 F37E 6C OC 00
3942 ;
3943 ;
3944 ;
3945 ;
3946 ;
3947 ;
3948 F381 A2 OD
3949 F383 A0 F1
3950 ;
3951 ;
3952 ;
3953 ;
3954 ;
3955 ;
3956 ;
3957 ;
3958 F385 8A
3959 F386 A2 00
3960 F38B 9D 44 03
3961 F388 98
3962 F38C 9D 45 03
3963 F38F A9 09
3964 F391 9D 42 03
3965 F394 A9 FF
3966 F396 9D 48 03
3967 F399 20 56 E4

; INCR BOOT LOADER BUFFER POINTER.
; DEC DBSECT ; DEC # OF SECTORS.
; YES, INCR SECTOR #
; JSR GETSEC ; GO GET SECTOR.
; LDA XSSBT ; STATUS O.K.?
; BEQ SECX ; IF DISK, TRY SECTOR AGAIN.
; INC BOOTAD ; SHOW BOOT SUCCESS
; JSR GETSEC ; YES, GET EOF RECORD, BUT DON'T USE IT.
; JSR BLOAD ; GO EXECUTE BOOT LOADER
; JMP (RAMLO)
; JSR (DOSINI)
; LDA #SEX
; STA ICBAH,X ; SET UP ADDR OF BEGIN OF LINE
; LDA #ICBL,X ; SET BUFFER LENGTH
; JSR CIOV ; PUT LINE ON SCREEN
ERR LINE ADDR B1 B2 B3 B4
3968 F39C 60 RTS
3969
3970
3971
3972
3973
3974
3975 F39D A5 4B GETSEC: LDA CASSBT
3976 F39F F0 03 BEG DISKM: ;CASSETTE BOOT ?
3977 F3A1 4C 7A E4 JMP RBLOKV ;YES, GO TO READ BLOCK ROUTINE
3978 F3A4 A9 52 DISKM: LDA #READ
3979 F3A6 8D 02 03 STA DCOMND ;SET READ SECTOR COMMAND
3980 F3A9 A9 01 LDA #1
3981 F3AB 8D 01 03 STA DUNIT ;SET DRIVE NO. TO DRIVE 0
3982 F3AE 20 53 E4 JSR DSKINV ;GET SECTOR
3983 F3B1 60 RTS
3984
3985
3986
3987
3988
3989 F3B2 A5 08 ;DO CHECK FOR CASSETTE BOOT & IF SO, DO BOOT
3990 F3B4 F0 0A CSBOOT: LDA WARMST ;WARMSTART?
3991 F3B6 A5 09 BEQ CSBOOT2 ;NO
3992 F3B8 29 02 LDA BOOT? ;GET BOOT FLAG
3993 F3BA F0 03 AND #2 ;WAS CASSETTE BOOT SUCCESFULL?
3994 F3BC 20 E1 F3 BEQ NOCSD2 ;NO
3995 F3BF 60 JSR CINI ;YES, INIT CASSETTE SOFTWARE
3996
3997 F3C0 A5 4A
3998 F3C2 F0 1C CSBOOT2: LDA CKEY
3999 F3C4 A9 80 BEQ NOCSD2 ;"C" KEY FLAG SET ?
4000 F3C6 85 3E LDA #80 ;YES,
4001 F3CB E4 4B STA FTYPE ;SET LONG IRG TYPE
4002 F3CA 20 7D E4 JSR CSOPIV ;OPEN CASSETTE FOR INPUT
4003 F3CD 20 01 F3 JSR SECT1 ;DO BOOT & INIT.
4004 F3D0 A9 00 LDA #0
4005 F3D2 85 4B STA CASSBT ;RESET CASSETTE BOOT FLAG
4006 F3D4 85 4A STA CKEY ;CLEAR KEY FLAG
4007 F3D6 06 09 ASL BOOT? ;SHIFT BOOT FLAG (NOW=2 IF SUCCESS)
4008 F3DB A5 0C LDA DOSINI
4009 F3DA 85 02 STA CASINI ;MOVE INIT ADDRESS FOR CASSETTE
4010 F3DC A5 0D LDA DOSINI+1
4011 F3DE 85 03 STA CASINI+1
4012 F3E0 60 NOCSBT: RTS
4013
4014 F3E1 6C 02 00
4015 CINI: JMP (CASINI) ;INIT CASSETTE
4016 ;*******************************************************
4017 ; SPARE BYTE OR MODULE TOO LONG FLAG
4018 ;*******************************************************
ERR LINE  ADDR  B1  B2  B3  B4  MONITOR  *****  MONITP.SRC  *****  3/9/79  *****  4:00  PAGE  90

4022  4023  0014  00

MONSPR:  .BYTE  KBDORG-CRNP7  ;^MONITP TOO LONG
4025  PAGE
4026  TITLE 'DISPLAY HANDLER -- 10-30-78 -- DISPLC'
4027  ; HANDLER DEPENDENT EQUATES
4028
4029
4030  007D  CLRCOD = $7D ; CLEAR SCREEN ASCI CODE
4031  009F  CNTL1 = $9F ; POKEY KEY CODE FOR ^1
4032
4033  0068  FRMADR = SAVADR
4034  0066  TOADR = MLTTMP
4035
ERR LINE ADDR B1 B2 B3 B4  DISPLAY HANDLER -- 10-30-78 -- DISPLC  PAGE 92

4036                   PAGE
4037                   ;
4038                   ; *=EDITRV
4039                   ; SCREEN EDITOR HANDLER ENTRY POINT
4040                   ;
4041                   ;
4042                   ;
4043 E400 FB F3  EDITOR: .WORD EOPEN-1
4044 E402 33 F6 .WORD RETUR1-1 ; (CLOSE)
4045 E404 3D F6 .WORD EGETCH-1
4046 E406 A3 F6 .WORD EOUTH-1
4047 E408 33 F6 .WORD RETUR1-1 ; (STATUS)
4048 E40A 3C F6 .WORD NOFUNC-1 ; (SPECIAL)
4049 E40C 4C E4 F3 JMP PWRONA
4050 E40F 00 .BYTE 0 ; ROM FILLER BYTE
4051                   ;
4052                   ; *=SCREEN
4053                   ; DISPLAY HANDLER ENTRY POINT
4054                   ;
4055                   ;
4056 E410 F5 F3  DISPLA: .WORD DOPEN-1
4057 E412 33 F6 .WORD RETUR1-1 ; (CLOSE)
4058 E414 92 F5 .WORD GETCH-1
4059 E416 B6 F5 .WORD EOUTH-1
4060 E418 33 F6 .WORD RETUR1-1 ; (STATUS)
4061 E41A FB FC .WORD DRAW-1 ; (SPECIAL)
4062 E41C 4C E4 F3 JMP PWRONA
4063 E41F 00 .BYTE 0 ; ROM FILLER BYTE
4064                   ;
4065                   ; *=KEYBDV
4066                   ;
4067                   ; KEYBOARD HANDLER ENTRY POINT
4068                   ;
4069                   ;
4070 E420 33 F6  KBDHND: .WORD RETUR1-1
4071 E422 33 F6 .WORD RETUR1-1 ; (CLOSE)
4072 E424 E1 F6 .WORD KGGETCH-1
4073 E426 3C F6 .WORD NOFUNC-1 ; (OUTCH)
4074 E428 33 F6 .WORD RETUR1-1 ; (STATUS)
4075 E42A 3C F6 .WORD NOFUNC-1 ; (SPECIAL)
4076 E42C 4C E4 F3 JMP PWRONA
4077 E42F 00 .BYTE 0 ; ROM FILLER BYTE
4078                   ;
4079                   ;
4080                   ; INTERRUPT VECTOR TABLE ENTRY
4081                     **VCTABL-I#TABS+VKEYBD
4082 E488 BE FF  .WORD PIRGS ; KEYBOARD IRQ INTERRUPT VECTOR
<table>
<thead>
<tr>
<th>ERROR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4083</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>***KBDORG</td>
</tr>
<tr>
<td>4084</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4085</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4086</td>
<td>F3E4</td>
<td>A9</td>
<td>FF</td>
<td></td>
<td></td>
<td>PWRON: LDA $FF</td>
</tr>
<tr>
<td>4087</td>
<td>F3E6</td>
<td>BD</td>
<td>FC</td>
<td>02</td>
<td></td>
<td>STA CH</td>
</tr>
<tr>
<td>4088</td>
<td>F3E9</td>
<td>AD</td>
<td>E6</td>
<td>02</td>
<td></td>
<td>LDA MEMTOP+1</td>
</tr>
<tr>
<td>4089</td>
<td>F3EC</td>
<td>29</td>
<td>F0</td>
<td></td>
<td></td>
<td>AND #$F0</td>
</tr>
<tr>
<td>4090</td>
<td>F3EE</td>
<td>B5</td>
<td>6A</td>
<td></td>
<td></td>
<td>INSURE 4K PAGE BOUNDARY</td>
</tr>
<tr>
<td>4091</td>
<td>F3F0</td>
<td>A9</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4092</td>
<td>F3F2</td>
<td>BD</td>
<td>BE</td>
<td>02</td>
<td></td>
<td>LDA #$40</td>
</tr>
<tr>
<td>4093</td>
<td>F3F5</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>DEFAULT TO UPPER CASE ALPHA AT PWRON</td>
</tr>
<tr>
<td>4094</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RTS: POWER ON COMPLETED
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

4094
4095
4096
4097
4098
4099 F3F6 A5 2B DOPEN: LDA ICAX2I ;GET AUX 2 BYTE
4100 F3FB 29 0F AND $0F
4101 F3FA D0 08 BNE OPNCOM ;IF MODE ZERO, CLEAR ICAX1Z
4102 F3FC A5 2A EOPEN: LDA ICAX1Z ;CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
4103 F3FE 29 0F AND $0F
4104 F400 B5 2A STA ICAX1Z
4105 F402 A9 00 STA $0
4106 F404 B5 57 STA DINDEX
4107 F406 A9 E0 LDA $#EO ;INITIALIZE GLOBAL VBLANK RAM
4108 F408 BD F4 02 STA CHBAS
4109 F40B A9 02 LDA $2
4110 F40D BD F3 02 STA CHACT
4111 F410 BD 2F 02 STA SDMCTL ;TURN OFF DMA NEXT VBLANK
4112 F413 A9 01 LDA #$SUCCES
4113 F415 B5 4C STA DSTAT ;CLEAR STATUS
4114 F417 A9 C0 LDA #$CO ;DO IRGEN
4115 F419 05 10 ORA POKMSK
4116 F41B 85 10 STA POKMSK
4117 F41D 8D 2F 02 STA SDMCTL
4118 F41F AO OE LDY #$14 ;CLEAR TAB STOPS
4119 F421 A9 01 LDA #$1 ;INIT TAB STOPS TO EVERY 8 CHARACTERS
4120 F423 BD 73 02 STA TINDEX ;TEXT INDEX MUST ALWAYS BE 0
4121 F425 B5 64 STA ADDRES
4122 F427 B5 7B STA SWPFL0
4123 F429 BD F0 02 STA CRSINH ;TURN CURSOR ON AT OPEN
4124 F42B A0 0E LDY #$14 ;CLEAR TAB STOPS
4125 F42D A9 01 LDA #$1 ;INIT TAB STOPS TO EVERY 8 CHARACTERS
4126 F430 99 A3 02 CLRTBS: STA TABMAP,Y
4127 F433 8B DEY
4128 F434 10 FA BPL CLRTBS
4129 F436 A2 04 LDX #$4 ;LOAD COLOR REGISTERS
4130 F438 BD C1 FE DOPENB: LDA COLORB,X
4131 F43A 9D C4 02 STA COLORO,X
4132 F43C CA DEY
4133 F43E F3 10 F7 BPL DOPENB
4134 F440 A4 6A LDY RAMTOP ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
4135 F442 8B DEY
4136 F444 9C 95 02 STY TXTMSC+1
4137 F446 A9 60 LDA #$60 ;TXTRSC
4138 F448 BD 94 02 STA TXTMSC
4139 F44A 85 57 LDY DINDEX
4140 F44C BD 69 FE STA ANCONV,X ;CONVERT IT TO ANTIC CODE
4141 F44E A9 91 OPNERR: LDA #BADMOD ;IF ZERO, IT IS ILLEGAL
4142 F450 B5 4C STA DSTAT
4143 F452 B5 51 DOPENA: STA HOLDI ;SET UP AN INDIRECT POINTER
4144 F454 A5 6A LDA RAMTOP
4145 F456 BD 63 STA ADDRESS+1
4146 F458 BC 45 FE LDY ALOCAT,X ;ALLOCATE N BLOCKS OF 40 BYTES
4147 F460 A9 2B DOPEN1: LDA #40
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-76 -- DISPLCG

4148 F462 20 21 F9 JSR DBSUB
4149 F465 86 DEY
4150 F466 D0 F8 BNE DOPEN1
4151 F468 AD 6F 02 LDA GPRIOR ;CLEAR GTIA MODES
4152 F469 29 3F AND #$3F
4153 F46D 85 67 STA OPNTMP+1
4154 F46F A8 TAY
4155 F470 88 DEV
4156 F472 90 17 BCC NOTB
4157 F474 8A TXA
4158 F475 6A ROR A
4159 F476 6A ROR A
4160 F477 6A ROR A
4161 F478 29 C0 AND #$C0 ;NOW 2 TOP BITS
4162 F47A 05 67 ORA OPNTMP+1
4163 F47B A8 TAY
4164 F47D A9 10 LDA #16 ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
4165 F47F 20 21 F9 JSR DBSUB
4166 F482 E0 00 CPX #8 ;TEST IF 320X1
4167 F484 D0 05 BNE NOTB ;IF MODE = 11
4168 F486 A9 06 LDA #6 ;PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
4169 F488 BD CB 02 STA COLOR4
4170 F48B BC 6F 02 NOTB: STY GPRIOR ;STORE NEW PRIORITY
4171 F48E A5 64 LDA ADRESS ;SAVE MEMORY SCAN COUNTER ADDRESS
4172 F490 A5 58 STA SAVMSC
4173 F492 A5 65 LDA ADRESS+1
4174 F494 A5 59 STA SAVMSC+1
4175 F496 AD OB 0A VBLANK: LDA VCOUNT ;WAIT FOR NEXT VBLANK BEFORE MESSING
4176 F499 C9 7A CMP #$7A ;WITH THE DISPLAY LIST
4177 F49B D0 F9 BNE VBWAIT
4178 F49D 20 1F F9 JSR DDDEC ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
4179 F4A0 BD 75 FE LDA PAGEBT,X ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
4180 F4A3 F0 06 BEQ NOMOD ;OF CROSSING A 256 BYTE PAGE BOUNDARY
4181 F4A5 A9 FF LDA:#FF ;IF SO, DROP DOWN A PAGE
4182 F4A7 B5 64 STA ADRESS
4183 F4A9 C6 65 DEC ADRESS+1
4184 F4AB A5 64 NOMOD: LDA ADRESS ;SAVE END OF DISPLAY LIST FOR LATER
4185 F4AD B5 68 STA SAVADR
4186 F4AF A5 65 LDA ADRESS+1
4187 F4B1 B5 69 STA SAVADR+1
4188 F4B3 20 13 F9 JSR DDDEC ;(DOUBLE BYTE DOUBLE DECREMENT)
4189 F4B6 A9 41 LDA #$41 ;(ANTIC) WAIT FOR VBLANK AND JMP TO TOP
4190 F4BB 20 17 F9 JSR STORE
4191 F4BB B6 66 STX OPNTMP
4192 F4BD A9 1B LDA #24 ;INITIALIZE BOTSCR
4193 F4BF BD BF 02 STA BOTSCR
4194 F4C2 A5 57 LDA DINDEX ;DISALLOW MIXED MODE IF MODE.GE.9
4195 F4C4 C9 09 CMP #$9
4196 F4C6 B0 2D BCS NOTMIXD
4197 F4C8 A5 2A LDA ICAX1Z ;TEST MIXED MODE
4198 F4CA 29 10 AND #$10
4199 F4CC F0 27 BEQ NOTMIXD
4200 F4CE A9 04 LDA #4
4201 F4D0 BD BF 02 STA BOTSCR
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>4202</td>
<td>F4D3</td>
<td>A2</td>
<td>02</td>
<td></td>
<td></td>
<td>LDX #2</td>
</tr>
<tr>
<td>4203</td>
<td>F4D5</td>
<td>A9</td>
<td>02</td>
<td></td>
<td></td>
<td>DOPEN2: JSR STORE</td>
</tr>
<tr>
<td>4204</td>
<td>F4D7</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4205</td>
<td>F4DA</td>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4206</td>
<td>F4DB</td>
<td>10</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4207</td>
<td>F4DD</td>
<td>A4</td>
<td>6A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4208</td>
<td>F4DF</td>
<td>BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4209</td>
<td>F4E0</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4210</td>
<td>F4E1</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4211</td>
<td>F4E4</td>
<td>A9</td>
<td>60</td>
<td></td>
<td></td>
<td>JSR STORE</td>
</tr>
<tr>
<td>4212</td>
<td>F4E6</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4213</td>
<td>F4E9</td>
<td>A9</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4214</td>
<td>F4EB</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4215</td>
<td>F4EE</td>
<td>1B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4216</td>
<td>F4EF</td>
<td>A9</td>
<td>0C</td>
<td></td>
<td></td>
<td>LDA #MXDMDE-NUMDLE; POINT X AT MIXED MODE TABLE</td>
</tr>
<tr>
<td>4217</td>
<td>F4F1</td>
<td>65</td>
<td>66</td>
<td></td>
<td></td>
<td>DOPEN3: LDA HOLD1; STORE N DLE'S</td>
</tr>
<tr>
<td>4218</td>
<td>F4F3</td>
<td>B5</td>
<td>66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4219</td>
<td>F4F5</td>
<td>A4</td>
<td>66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4220</td>
<td>F4F7</td>
<td>BE</td>
<td>51</td>
<td>FE</td>
<td></td>
<td>LDX NUMDLE,Y; GET NUMBER OF DISPLAY LIST ENTRIES</td>
</tr>
<tr>
<td>4221</td>
<td>F4FA</td>
<td>A5</td>
<td>51</td>
<td></td>
<td></td>
<td>DOPEN4: LDA HOLD1; DO REMAINING DLE'S</td>
</tr>
<tr>
<td>4222</td>
<td>F4FC</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4223</td>
<td>F4FF</td>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4224</td>
<td>F500</td>
<td>DO</td>
<td>F8</td>
<td></td>
<td></td>
<td>BNE DOPEN4; DO THE MESSY 320X1 PROBLEM</td>
</tr>
<tr>
<td>4225</td>
<td>F502</td>
<td>A5</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4226</td>
<td>F504</td>
<td>C9</td>
<td>08</td>
<td></td>
<td></td>
<td>DOPEN5: LDA #73; GET REMAINING NUMBER OF DLE'S</td>
</tr>
<tr>
<td>4227</td>
<td>F506</td>
<td>90</td>
<td>1C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4228</td>
<td>F508</td>
<td>A2</td>
<td>5D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4229</td>
<td>F50A</td>
<td>A5</td>
<td>6A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4230</td>
<td>F50C</td>
<td>3B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4231</td>
<td>F50D</td>
<td>E9</td>
<td>10</td>
<td></td>
<td></td>
<td>SBC #10; RELOAD MEMORY SCAN COUNTER</td>
</tr>
<tr>
<td>4232</td>
<td>F50F</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4233</td>
<td>F512</td>
<td>A9</td>
<td>00</td>
<td></td>
<td></td>
<td>JSR STORE</td>
</tr>
<tr>
<td>4234</td>
<td>F514</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4235</td>
<td>F517</td>
<td>A9</td>
<td>4F</td>
<td></td>
<td></td>
<td>DOPEN6: LDA HOLD1; (ANTIC) RELOAD MSC CODE</td>
</tr>
<tr>
<td>4236</td>
<td>F519</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4237</td>
<td>F51C</td>
<td>A5</td>
<td>51</td>
<td></td>
<td></td>
<td>DOPEN7: LDA HOLD1; DO REMAINING DLE'S</td>
</tr>
<tr>
<td>4238</td>
<td>F51E</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4239</td>
<td>F521</td>
<td>CA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4240</td>
<td>F522</td>
<td>DO</td>
<td>F8</td>
<td></td>
<td></td>
<td>BNE DOPEN7; POLISH OFF DISPLAY LIST</td>
</tr>
<tr>
<td>4241</td>
<td>F524</td>
<td>A5</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4242</td>
<td>F526</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4243</td>
<td>F529</td>
<td>A5</td>
<td>5B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4244</td>
<td>F52B</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4245</td>
<td>F52E</td>
<td>A5</td>
<td>51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4246</td>
<td>F530</td>
<td>09</td>
<td>40</td>
<td></td>
<td></td>
<td>ORA #40; 24 BLANK LINES</td>
</tr>
<tr>
<td>4247</td>
<td>F532</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4248</td>
<td>F535</td>
<td>A9</td>
<td>70</td>
<td></td>
<td></td>
<td>JSR STORE</td>
</tr>
<tr>
<td>4249</td>
<td>F537</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4250</td>
<td>F53A</td>
<td>A9</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4251</td>
<td>F53C</td>
<td>20</td>
<td>17</td>
<td>F9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4252</td>
<td>F53F</td>
<td>A5</td>
<td>64</td>
<td></td>
<td></td>
<td>LDA ADDRESS; SAVE DISPLAY LIST ADDRESS</td>
</tr>
<tr>
<td>4253</td>
<td>F541</td>
<td>BD</td>
<td>30</td>
<td>02</td>
<td></td>
<td>STA SDLSTL</td>
</tr>
<tr>
<td>4254</td>
<td>F544</td>
<td>A5</td>
<td>65</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4255</td>
<td>F546</td>
<td>BD</td>
<td>31</td>
<td>02</td>
<td></td>
<td>STA SDLSTL+1</td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC PAGE 97

4256 F549 A9 70  LDA #$70  ; ADD LAST BLANK LINE ENTRY
4257 F54B 20 17 F9  JSR STORE  ; POSITION ADDRES=SDLSTL-1
4258 F54E A5 64  LDA ADDRES  ; STORE NEW MEMTOP
4259 F550 8D E5 02  STA MEMTOP
4260 F553 A5 65  LDA ADDRES+1
4261 F555 8D E6 02  STA MEMTOP+1
4262 F558 A5 68  STA SAVADR
4263 F55A 85 64  LDA ADDRESS
4264 F55C 8D E5 02  STA MEMTOP
4265 F55E A5 65  LDA SAVADR+1
4266 F560 AD 31 02  LDA SDLSTL+1
4267 F563 20 17 F9  JSR STORE
4268 F566 AD 30 02  LDA SDLSTL
4269 F568 20 17 F9  JSR STORE
4270 F56A A5 4C  LDA DSTAT  ; IF ERROR OCCURRED ON ALLOCATION, OPEN THE ED
4271 F56E 10 07  BPL DOPEN9  ; SAVE STATUS
4272 F570 4B  PHA  ; OPEN THE EDITOR
4273 F571 20 FC F3  JSR EOPEN  ; OPEN THE EDITOR
4274 F574 6B  PLA  ; RESTORE STATUS
4275 F575 A8  TAY  ; AND RETURN IT TO CIO
4276 F576 60  RTS
4277 F577 A5 2A  DOPEN9: LDA ICAX1Z  ; TEST CLEAR INHIBIT BIT
4278 F579 29 20  AND #$20
4279 F57B D0 0B  BNE DOPEN7  ; CLEAR SCREEN
4280 F57D 20 B9 F7  JSR CLRSOR  ; AND HOME TEXT CURSOR (AC IS ZERO)
4281 F580 BD 90 02  LDA TXTROW  ; EVERYTHING ELSE IS SET UP
4282 F583 A5 52  STA LMARGN
4283 F585 BD 91 02  LDA TXTCOL
4284 F588 A9 22  DOPEN7: LDA #$22
4285 F58A 00 2F 02  ORA SDMCTL  ; SO TURN ON DMACTL
4286 F58D BD 2F 02  STA SDMCTL
4287 F590 4C 21 F6  JMP RETUR2
4288
4289
4290 F593 20 9E FA  GETCH: JSR RANGE  ; GETCH DOES INCRSR, GETPLT DOESN'T
4291 F596 20 A2 F5  JSR GETPLT
4292 F599 20 32 F8  JSR INATAC  ; CONVERT INTERNAL CODE TO ATASCII
4293 F59C 20 D4 F9  JSR INCRSB
4294 F59F 4C 34 F6  JMP RETUR1
4295 F5A2 20 47 F9  GETPLT: JSR CONVRT  ; CONVERT ROW/COLUMN TO ADDRESS
4296 F5A5 B1 64  LDA (ADDRESS),Y
4297 F5A7 2D A0 02  AND DMASK
4298 F5AA 46 6F  SHIFTD: LSR SHFAMT  ; SHIFT DATA DOWN TO LOW BITS
4299 F5AC 80 03  BCS SHIFT1
4300 F5AE 44 4A  LSR A
4301 F5AF 10 F9  BPL SHIFTD  ; (UNCONDITIONAL)
4302 F5B1 BD FA 02  SHIFT1: STA CHAR
4303 F5BA C9 00  CMP #0
4304 F5B6 60  RTS
4305
4306
4307
4308 F5B7 BD FB 02  OUTCH: STA ATACHR
4309 F5BA 20 9E FA  JSR RANGE
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

4310 F5BD AD FB 02 ; TEST FOR CLEAR SCREEN
4311 F5C0 C9 7D LDA ATACHR
4312 F5C2 D0 06 CMP #CLRCOD
4313 F5C4 20 B9 F7 BNE OUTCHB
4314 F5C7 4C 21 F6 JMP RETUR2
4315 F5CA AD FB 02 ; TEST FOR CARRIAGE RETURN
4316 F5CD C9 98 LDA ATACHR
4317 F5CF D0 06 CMP #CR
4318 F5D0 20 30 FA BNE OUTCHB
4319 F5D1 10 F9 JSR CLRSCR
431A 4C 21 F6 JMP RETUR2
431B F5DA 20 08 F9 JSR INCRSR
431C F5DD 4C 21 F6 JMP RETUR2
431D
431E
431F F5EE AD FF 02 ; *****LOOP HERE IF START/STOP FLAG IS NON-O
4320 F5F1 A8 LDA SSFLAG
4321 F5F2 2A TAY
4322 F5F3 2A X
4323 F5F4 2A ROL A
4324 F5F5 2A ROL A
4325 F5F6 2A ROL A
4326 F5F7 2A AND #3
4327 F5F8 AA TAX
4328 F5F9 98 TYA
4329 F5FA 29 9F AND #$9F
432A F5FB 1D F6 FE ORA ATATINDEX
432B F5FC 8D FA 02 STA ATACHR
432C F5FD 20 47 F9 JSR CONVRT
432D F5FE AD FA 02 LDA ATACHR
432E F5FF 46 6F SHIFTU: LSR SHFAMT
4330 F600 B0 04 BCS SHIFT2
4331 F601 0A ASL A
4332 F602 4C 08 F6 JMP SHIFTU
4333 F603 20 A0 02 SHIFT2: AND DMASK
4334 F604 B5 50 STA TMPCHR
4335 F605 AD AO 02 LDA DMASK
4336 F606 49 FF EOR #$FF
4337 F607 31 64 AND (ADDRESS),Y
4338 F608 05 50 ORA TMPCHR
4339 F609 91 64 STA (ADDRESS),Y
433A F60A 60 RTS
433B
433C F610 20 A2 F5 RETUR2: JSR GETPLT
433D F611 89 SD STA OLDCHR
433E F612 A6 57 LDX DINDEX
433F
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>CODE</th>
<th>OP</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>4364</td>
<td>F62B</td>
<td>D0</td>
<td>0A</td>
<td></td>
<td></td>
<td>BNE</td>
<td>RETU1</td>
<td></td>
</tr>
<tr>
<td>4365</td>
<td>F62A</td>
<td>AE</td>
<td>F0</td>
<td>02</td>
<td></td>
<td>LDX</td>
<td>CRShNH</td>
<td></td>
</tr>
<tr>
<td>4366</td>
<td>F62D</td>
<td>D0</td>
<td>05</td>
<td></td>
<td></td>
<td>BNE</td>
<td>RETU1</td>
<td></td>
</tr>
<tr>
<td>4367</td>
<td>F62F</td>
<td>49</td>
<td>B0</td>
<td></td>
<td></td>
<td>EOR</td>
<td>##80</td>
<td></td>
</tr>
<tr>
<td>4368</td>
<td>F631</td>
<td>20</td>
<td>FF</td>
<td>F5</td>
<td></td>
<td>JSR</td>
<td>OUTF2</td>
<td></td>
</tr>
<tr>
<td>4369</td>
<td>F634</td>
<td>A4</td>
<td>4C</td>
<td></td>
<td></td>
<td>RETU1</td>
<td>0E64</td>
<td></td>
</tr>
<tr>
<td>4370</td>
<td>F636</td>
<td>A9</td>
<td>01</td>
<td></td>
<td></td>
<td>LDA</td>
<td>#SUSCE</td>
<td></td>
</tr>
<tr>
<td>4371</td>
<td>F638</td>
<td>B9</td>
<td>4C</td>
<td></td>
<td></td>
<td>STA</td>
<td>DSTAT</td>
<td></td>
</tr>
<tr>
<td>4372</td>
<td>F63A</td>
<td>AD</td>
<td>FB</td>
<td>02</td>
<td></td>
<td>LDA</td>
<td>ATACHR</td>
<td></td>
</tr>
<tr>
<td>4373</td>
<td>F63D</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>NOFUNC: RTS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

; TEST CURSOR INHIBIT
; TOGGLE MSB
; DISPLAY IT
; RETURN TO CIO WITH STATUS IN Y
; SET STATUS= SUCCESSFUL COMPLETION
; PUT ATACHR IN AC FOR RETURN TO CIO
; (NON-EXISTENT FUNCTION RETURN POINT)

; END OF DISPLAY HANDLER
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

4379
4380
4381
4382
4383
4384 F63E 20 B3 FC EGETCH: JSR SWAP
4385 F641 20 B8 FA JSR ERANGE
4386 F644 A5 6B LDA BUFCNT ; ANYTHING IN THE BUFFER?
4387 F646 D0 34 BNE EGETCH3 ; YES
4388 F648 A5 54 LDA ROWCRS ; NO, SO SAVE BUFFER START ADDRESS
4389 F64A B5 6C STA BUFSTR
4390 F64C A5 55 LDA COLCRS
4391 F64E B5 6D STA BUFSTR+1
4392 F650 20 E2 F6 EGETCH1: JSR KOETCH ; LET'S FILL OUR BUFFER
4393 F653 B4 4C STY DSTAT ; SAVE KEYBOARD STATUS
4394 F655 AD FB 02 LDA ATACHR ; TEST FOR CR
4395 F658 C9 9B CMP #CR
4396 F65A F0 12 BEG EGETCH2
4397 F65C 20 AD F6 JSR DOSI ; NO, SO PRINT IT
4398 F65F 20 B3 FC JSR SWAP ; JSR DOSI DID SWAP SO SWAP BACK
4399 F662 A5 63 LDA LOGCOL ; BEEP IF NEARING LOGICAL COL 120
4400 F664 C9 71 CMP #113
4401 F666 D0 03 BNE EGETCH6
4402 F66B 20 0A F9 JSR BELL
4403 F66B 4C 50 F6 EGETCH6: JMP EGETCH1
4404 F66E 20 E4 FA EGETCH2: JSR OFFCRS ; GET BUFFER COUNT
4405 F671 20 00 FC JSR DOBUFC
4406 F674 A5 6C LDA BUFSTR ; RETURN A CHARACTER
4407 F676 B5 54 STA ROWCRS
4408 F678 A5 6D LDA BUFSTR+1
4409 F67A B5 55 STA COLCRS
4410 F67C A5 6B EGETCH3: LDA BUFNT
4411 F67E F0 11 BEG EGETCH5
4412 F680 C6 6B EGETCH7: BEG EGETCH5 
4413 F682 F0 0D BEG EGETCH5
4414 F684 A5 4C LDA DSTAT ; IF ERR, LOOP ON EGETCH7 UNTIL BUFR IS EMPTIE
4415 F686 30 F8 BMI EGETCH7
4416 F688 20 93 F5 JSR GETCH
4417 F68B 8D FB 02 STA ATACHR
4418 F68E 4C 83 FC JMP SWAP ; AND RETURN WITHOUT TURNING CURSOR BACK ON
4419 F691 20 30 FA EGETCH5: JSR DOCRWS ; DO REAL CARRIAGE RETURN
4420 F694 A9 9B LDA #CR ; AND RETURN EDL
4421 F696 8D FB 02 STA ATACHR
4422 F699 20 B1 F6 JSR RETURN1 ; TURN ON CURSOR THEN SWAP
4423 F69C B4 4C STY DSTAT ; SAVE KEYBOARD STATUS
4424 F69E 4C 83 FC JMP SWAP ; AND RETURN THROUGH RETURN1
4425
4426 F6A1 6C 64 00 JSRIND: JMP (ADDRESS) ; JSR TO THIS CAUSES JSR INDIRECT
4427
4428 F6A4 8D FB 02 EOUCHT: STA ATACHR ; SAVE ATASCII VALUE
4429 F6A7 20 B3 FC JSR SWAP
4430 F6AA 20 B8 FA JSR ERANGE
4431 F6AD 20 E4 FA DOSI: JSR OFFCRS ; TURN OFF CURSOR
4432 F6B0 20 BD FC JSR TSTCTL ; TEST FOR CONTROL CHARACTERS (Z=1 IF CTL)
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC PAGE 101

4433 F6B3 F0 09 BEG EOUTC5
4434 F6B5 OE A2 02 EOUTC6: ASL ESCFLG ; ESCFLG ONLY WORKS ONCE
4435 F6BB 20 CA F5 JSR OUTCHE
4436 F6BB 4C B3 FC ERETN: JMP SWAP ; AND RETURN THROUGH RETUR1
4437 F6BE AD FE 02 EOUTC5: LDA DSPFLG ; DO DSPFLG AND ESCFLG
4438 F6C1 OD A2 02 ORA ESCFLG
4439 F6C4 D0 EF BNE EOUTC6 ; IF NON-O DISPLAY RATHER THAN EXECUTE IT
4440 F6C6 OE A2 02 ASL ESCFLG
4441 F6C9 E8 INX ; PROCESS CONTROL CHARACTERS
4442 F6CA BD C6 FE LDA CNTRLS, X ; GET DISPLACEMENT INTO ROUTINE
4443 F6CD B5 64 STA ADDRESS
4444 F6CF BD C7 FE LDA CNTRLS+1, X ; GET HIGH BYTE
4445 F6D2 B5 65 STA ADDRESS+1
4446 F6D4 20 A1 F6 JSR JSRIND ; DO COMPUTED JSR
4447 F6D7 20 21 F6 JSR RETUR2 ; DO CURSOR
4448 F6DA 4C B3 FC JMP SWAP ; ALL DONE SO RETURN THROUGH RETUR1
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461 F6DD A9 FF KGETC2: LDA #$FF
4462 F6DF BD FC 02 STA CH
4463 F6E2 A3 2A KGETCH: LDA ICAX1Z ; TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MOD
4464 F6E4 4A LSR A
4465 F6E5 B0 62 BCS GETOUT
4466 F6E7 A9 B0 LDA #BRKABT
4467 F6E9 A6 11 LDX BRKKEY ; TEST BREAK
4468 F6EB F0 5B BEQ K7 ; IF BREAK, PUT BRKABT IN DSTAT AND CR IN ATA
4469 F6ED AD FC 02 LDA CH
4470 F6F0 C9 FF CMP #$FF
4471 F6F2 F0 EE BEG KGETCH
4472 F6F4 B5 7C STA HOLDCH ; SAVE CH FOR SHIFT LOCK PROC
4473 F6F6 A2 FF LDX #$FF ; "CLEAR" CH
4474 F6FB 8E FC 02 STX CH
4475 F6FB 20 D8 FC JSR CLICK ; DO KEYBOARD AUDIO FEEDBACK (A IS OK)
4476 F6FE AA KGETC3: TAX ; DO ASCCON
4477 F6FF E0 C0 CPX #$CO ; TEST FOR CTL & SHIFT TOGETHER
4478 F701 90 02 BCC ASCCO1
4479 F703 A2 03 LDX #$3 ; BAD CODE
4480 F705 BD FE FE ASCCO1: LDA ATABCI.X ; DONE
4481 F70B BD FB 02 STA ATACHR
4482 F70B C9 80 CMP #$80 ; DO NULLS
4483 F70D F0 CE BEG KGETC2
4484 F70F C9 B1 CMP #$B1 ; CHECK ATARI KEY
4485 F711 D0 08 BNE KGETC1
4486 F713 AD B6 02 LDA INVFLG
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

4487 F716 49 80 EOR #$80
4488 F718 BD B6 02 STA INVFLG
4489 F718 4C DD F6 JMP KGETC2 ; DONT RETURN A VALUE
4490 F71E C9 B2 KGETC1: CMP #$82 ; CAPS/LOWER.
4491 F720 D0 07 BNE K1
4492 F722 A9 00 LDA #0 ; CLEAR SHFLOK
4493 F724 BD BE 02 STA SHFLOK
4494 F727 F0 B4 BEQ KGETC2
4495 F729 C9 83 K1: CMP #$83 ; SHIFT CAPS/LOWER
4496 F72B D0 07 BNE K2
4497 F72D A9 40 LDA #$40
4498 F72F BD BE 02 STA SHFLOK ; SHIFT BIT
4499 F732 D0 A9 BNE KGETC2
4500 F734 C9 B4 K2: CMP #$84 ; CNTL CAPS/LOWER
4501 F736 D0 07 BNE K3
4502 F738 A9 80 LDA #$80 ; CNTL BIT
4503 F73A BD BE 02 STA SHFLOK
4504 F73D D0 9E BNE KGETC2
4505 F73F C9 B5 K3: CMP #$85 ; DO EOF
4506 F741 D0 DA BNE K6
4507 F743 A9 88 LDA #EOFERR
4508 F745 B5 4C K7: STA DSTAT
4509 F747 B5 11 STA BRKKEY ; RESTORE BREAK
4510 F749 A9 9B GETOUT: LDA #CR ; PUT CR IN ATACHR
4511 F74B D0 26 BNE K8 ; (UNCONDITIONAL)
4512 F74D A9 7C LDA HOLDCH ; PROCESS SHIFT LOCKS
4513 F74F C9 40 CMP #40 ; REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
4514 F751 BD 15 BCS K5 ; OVER LOCK
4515 F753 AD FB 02 LDA ATACHR ; TEST FOR ALPHA
4516 F756 C9 61 CMP #$61 ; LOWER CASE A
4517 F758 90 0E BCC K5 ; NOT ALPHA IF LT
4518 F75A C9 7B CMP #$7B ; LOWER CASE Z+1
4519 F75C BD 0A BCS K5 ; NOT ALPHA IF GE
4520 F75E AD BE 02 LDA SHFLOK ; DO SHIFT/CONTROL LOCK
4521 F761 F0 09 BEQ K5 ; IF NO LOCK, DONT RE-DO IT
4522 F763 DD 7C BRA HOLDCH
4523 F765 4C FE F6 JMP KGETC3 ; DO RETRY
4524 F768 20 BD FC K5: JSR TSTCTL ; DONT INVERT MSB OF CONTROL CHARACTERS
4525 F76B F0 09 BEQ K4
4526 F76D AD FB 02 LDA ATACHR
4527 F770 4D B6 02 EOR INVFLG
4528 F773 BD FB 02 K8: STA ATACHR
4529 F776 4C 34 F6 K4: JMP RETURN ALL DONE
4530
4531
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

PAGE 103

CONTROL CHARACTER PROCESSORS

ESCAPE: LDA #$80 ; SET ESCAPE FLAG
STA ESCLFLG

UPDNCM: STX ROWCRS
CMP BOTSRCR ; WRAPAROUND

COMRET1: JMP DOLCOL iCOLVERT ROW AND COL TO LOGCOL AND RETURN
COMRE1: JMP DOLCOL iCOLVERT ROW AND COL TO LOGCOL AND RETURN

CLRSC2: STA (ADDRESS),Y ; (AC IS ZERO)

CLRSC3: STA LQGMAP,Y ; (Y IS ZERO AFTER CLRSC2 LOOP)

CLEAN UP LOGICAL LINE BIT MAP

HOME: JSR COLCR ; PLACE COLCRS AT LEFT EDGE

PLACE COLCRS AT LEFT EDGE
ERR LINE  ADDR  B1  B2  B3  B4  DISPLAY HANDLER  --  10-30-78  --  DISPLC  PAGE 104

4586  F7DF  85  54  STA  ROWCRS
4587  F7E1  85  56  STA  COLCRS+1
4588  F7E3  85  6C  STA  BUFSTR
4589  F7E5  60  RTS
4590  
4591  F7E6  A5  63  BS:  LDA  LOGCOL  :BACKSPACE
4592  F7EB  C5  52  CMP  LMRGN  :LEFT EDGE?
4593  F7EA  F0  21  BEQ  BS1
4594  F7EC  A5  55  BSA:  LDA  COLCRS  :LEFT EDGE?
4595  F7EE  C5  52  CMP  LMRGN  :LEFT EDGE?
4596  F7E3  85  6C  STA  BUFSTR
4597  F7E5  60  RTS
4598  F7E6  A5  63  BS:  LDA  LOGCOL  :BACKSPACE
4599  F7EB  C5  52  CMP  LMRGN  :LEFT EDGE?
4600  F7EA  F0  21  BEQ  BS1
4601  F7EC  A5  55  BSA:  LDA  COLCRS  :LEFT EDGE?
4602  F7EE  C5  52  CMP  LMRGN  :LEFT EDGE?
4603  F7E3  85  6C  STA  BUFSTR
4604  F802  20  7F  F7  JSR  CRSRT  BEGIN SEARCH
4605  F805  A9  20  BS2:  LDA  #$20  :MAKE BACKSPACE DESTRUCTIVE
4606  F807  8D  FB  02  STA  ATACHR
4607  F80A  20  E0  F5  JSR  OUTPLT
4608  F80D  4C  DD  FB  BS1:  JMP  DOLCOL  :AND RETURN
4609  F810  20  A1  F7  TAB:  JSR  CRSTAB  BEGIN SEARCH
4610  F813  A5  55  LDA  COLCRS  :TEST FOR NEW LINE
4611  F815  C5  52  CMP  LMRGN  :LEFT EDGE?
4612  F817  D0  0A  BNE  TAB1  :NO
4613  F819  20  34  FA  JSR  DOCR  .DD CARRIAGE RETURN
4614  F81C  20  20  FB  JSR  LOGGET  .CHECK IF END OF LOGICAL LINE
4615  F81F  90  02  BCC  TAB1  .NO, CONTINUE
4616  F821  B0  07  BCS  TAB2  .(UNCONDITIONAL)
4617  F823  A5  63  TAB1:  LDA  LOGCOL  :CHECK FOR TAB STOP
4618  F825  20  25  FB  JSR  BITGET  .NO; SO KEEP LOOKING
4619  F828  90  E6  BCC  TAB  .NO; SO KEEP LOOKING
4620  F82A  4C  DD  FB  TAB2:  JMP  DOLCOL  :COLVERT ROW AND COL TO LOGCOL AND RETURN
4621  F82D  A5  63  SETTAB:  LDA  LOGCOL
4622  F82F  4C  06  FB  JMP  BITSET  :SET BIT IN MAP AND RETURN
4623  F832  A5  63  CLRTAB:  LDA  LOGCOL
4624  F834  4C  12  FB  JMP  BITCLR  :CLEAR " " " "
4625  F837  20  9D  FC  INSCHR:  JSR  PHACRS
4626  F83A  20  A2  F5  JSR  GETPLT  .GET CHARACTER UNDER CURSOR
4627  F83D  B5  7D  STA  INSDAT
4628  F83F  A9  00  LDA  #0
4629  F841  8D  BB  02  STA  SCRFGL
4630  F844  20  FF  F5  INSCH4:  JSR  OUTCH2  :STORE DATA
4631  F847  A5  63  LDA  LOGCOL  :SAVE LOGCOL: IF AFTER INCRSA LOGCOL IS
4632  F849  4B  PHA  #0
4633  F84A  20  DC  F9  JSR  INCRSA  :SPECIAL INCRSR ENTRY POINT
4634  F84D  6B  PLA
4635  F84E  C5  63  CMP  LOGCOL
4636  F850  8D  0C  BCS  INSCH3  :QUIT
4637  F852  A5  7D  INSCH1:  LDA  INSDAT  :KEEP GOING
4638  F854  4B  PHA
4639  F855  20  A2  F5  JSR  GETPLT

ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC PAGE 105

4640 F858 85 7D STA INSDAT
4641 F85A 68 PLA INSCH4
4642 F85B 4C 44 FB JMP INSCH3: JSR PLACRS
4643 F85E 20 A8 FC INSC6: JMP DOLCOL
4644 F861 CE BB 02 DEC SCRFGLG
4645 F864 30 04 BMI INSCH5 IF SCROLL OCCURRED
4646 F866 CA 34 DEC ROWCRS MOVE CURSOR UP
4647 F868 DD F7 BNE INSCH6 (UNCOND) CONTINUE UNTIL SCRFGLG IS MINUS
4648 F86A 4C DD FB INSCH5: JMP DOLCOL COLVERT ROW AND COL TO LOGCOL AND RETURN
4649
4650
4651 F86D 20 9D FC DELCHR: JSR PHACRS GET DATA TO THE RIGHT OF THE CURSOR
4652 F870 20 47 F9 DELCH1: JSR CONVRT GET DATA TO THE RIGHT OF THE CURSOR
4653 F873 A5 64 LDA ADRESS
4654 F875 B5 68 STA SAVADR SAVE ADDRESS TO KNOW WHERE TO PUT DATA
4655 F877 A5 65 LDA ADDRESS+1
4656 F879 B5 69 STA SAVADR+1
4657 F87B A5 63 LDA LOGCOL
4658 F87D 48 PHA
4659 F87E 20 D4 F9 JSR INCRSB PUT CURSOR OVER NEXT CHARACTER
4660 F881 68 PLA
4661 F882 C3 63 CMP LOGCOL TEST NEW LOGCOL AGAINST OLD LOGCOL
4662 F884 B0 10 BCS DELCH2 IF OLD. GE. NEW THEN QUIT
4663 F886 A5 54 LDA ROWCRS IS ROW OFF SCREEN?
4664 F888 CD BF 02 CMP DBCSR
4665 F88B B0 09 BCS DELCH2 YES, SO QUIT
4666 F88D 20 A2 F5 JSR GETPLT GET DATA UNDER CURSOR
4667 F890 A0 00 LDY #$0
4668 F892 91 68 STA (SAVADR), Y PUT IT IN PREVIOUS POSITION
4669 F894 FO DA BEQ DELCH1 AND LOOP (UNCONDITIONAL)
4670 F896 A0 00 LDY #$0
4671 F898 98 TYA
4672 F899 91 68 STA (SAVADR), Y CLEAR THE LAST POSITION
4673 F89B 20 68 FC JSR DELTIA TRY TO DELETE A LINE
4674 F89E 20 A8 FC JSR PLACRS AND RETURN
4675 F8A1 4C DD FB JMP DOLCOL AND RETURN
4676 F8A4 38 INSLIN: SEC NORMAL INSLIN PUTS "1" INTO BIT MAP
4677 F8A5 20 78 FB INSLIA: JSR EXTEND ENTRY POINT FOR C=0
4678 F8AA A5 52 STA LMARON DO CARRIAGE RETURN (NO LF)
4679 F8AC B5 55 STA COLORUS
4680 F8AC 20 47 F9 JSR CONVRT GET ADDRESS
4681 F8AF A5 64 LDA ADDRESS SET UP TO=40+FROM (FROM = CURSOR)
4682 F8B1 B5 68 STA FRMADR
4683 F8B3 18 CLC
4684 F8B4 69 28 ADC #40
4685 F8B6 B5 66 STA TOADR
4686 F8BA A5 65 LDA ADDRESS+1
4687 F8BA B5 69 STA FRMADR+1
4688 F8BC 69 00 ADC #0
4689 F8BE B5 67 STA TOADR+1
4690 F8C0 A6 54 LDX ROWCRS SET UP LOOP COUNTER
4691 F8C2 E0 17 CPX #23
4692 F8C4 F0 08 BEQ INSLL2
4693 F8C6 20 4E FB INSLL1: JSR MOVLIN
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

4694 FBC9 E8 INX
4695 FBCA E0 17 CPX #23
4696 FBCC D0 FB BNE INSLL1
4697 FBCE 20 9B F8 CPX INSLL2: JSR CLRLLN CLEAR CURRENT LINE
4698 FBDA 4C DD F8 JMP DOLCOL COLVERT ROW AND COL TO LOGCOL AND RETURN
4699 FBDA 20 DD F8 DELLIN: JSR DOLCOL GET BEGINNING OF LOG LINE (HOLD1)
4700 FBDA A4 91 DELLIA: LDY HOLD1 SQUEEZE BIT MAP
4701 FBDA B4 54 STY ROWCRS PUT CURSOR THERE
4702 FBDA A4 54 DELLIB: LDY ROWCRS
4703 FBDD 98 DELL1: TYA
4704 FBDE 3B SEC
4705 FBD0 20 23 F8 JSR LOGGET GET NEXT BIT
4706 FBE2 08 PHP
4707 FBE3 9B TYA
4708 FBE4 18 CLC
4709 FBE5 A9 7B ADC #120
4710 FBE7 28 PLP
4711 FBE8 20 04 F8 JSR BITPUT WRITE IT OVER PRESENT BIT
4712 FBE8 C8 INY
4713 FBEA C0 18 CPY #24
4714 FBEA D0 ED BNE DELL1: LOOP
4715 FBF0 AD B4 02 LDA LOGMAP+2 SET LSB #1
4716 FBF3 09 01 ORA #1
4717 FBF5 8D B4 02 STA LOGMAP+2
4718 FBF8 A5 52 DELL2: LDA LMARGN DELETE LINE OF DATA USING PART OF SCROLL
4719 FBF9 B5 55 STA COLCRS CR NO LF
4720 FBFC 20 47 F9 JSR CONVRT
4721 FBF0 20 B7 FB JSR SCROL1
4722 F902 20 20 F8 JSR LOGGET TEST NEXT LINE FOR CONTINUATION
4723 F905 9D 44 ; IS IT A NEW LOG LINE?
4724 F907 4C DD F8 BCC DELLIB AND SO DELETE ANOTHER
4725 F907 4C DD F8 JMP DOLCOL YES SO DOLCOL AND RETURN
4726 F90A A0 20 BELL: LDY #20
4727 F90C 20 28 FC BELL1: JSR CLICK
4728 F90E 88 DEY
4729 F910 10 FA BPL BELL1
4730 F912 60 RTS
.PAGE

; ROUTINES

; DOUBLE BYTE DECREMENT OF INDIRECT POINTER
; INCLUDING DB SUBTRACT AND DB DOUBLE DECREMENT

DBDDEC: LDA #2
BNE DBSUB  ; (UNCONDITIONAL)

; STORE DATA INDIRECT AND DECREMENT POINTER
; (PLACED HERE TO SAVE JMP DBDDEC AFTER STORE)
STORE: LDY DSTAT  ;RETURN ON ERROR

; STORE: STA (ADRESS),Y
; JMP DBDDEC DECREMENT AND RETURN

DBDEC: LDA #1

; DBSUB: STA SUBTMP
Dbsub: LDA DSTAT  ;RETURN ON ERROR

; LDA DBSUB1: MAKE SURE NOTHING EVER OVERWRITES APPMHI
Dbsub1: LDA APPMHI+1

; CONVRT: LDA ROWCRS  ;SAVE CURSOR
CONVRT: LDA ROWCRS
PHA

; LDA COLCRS
PHA

; COLCRS+1
PHA

; JSR PUTMSC
JSR PUTMSC

; PUT 10*ROWCRS INTO MLTTMP
LDA ROWCRS
STA MLTTMP
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC PAGE 108

4785 F957 A9 00 LDA #0
4786 F959 B5 67 STA MLTTMP+1
4787 F95B A5 66 LDA MLTTMP \QUICK X8
4788 F95D 0A ASL A
4789 F95E 26 67 ROL MLTTMP+1
4790 F960 B5 51 STA HOLD1 \(SAVE 2X VALUE\)
4791 F962 A4 67 LDY MLTTMP+1 \"
4792 F964 8C 9F 02 STY HOLD2 \"
4793 F967 0A ASL A
4794 F968 26 67 ROL MLTTMP+1
4795 F969 0A ASL A
4796 F96B 26 67 ROL MLTTMP+1
4797 F96D 18 CLC \ADD IN 2X
4798 F96E 65 51 ADC HOLD1
4799 F970 B5 66 STA MLTTMP
4800 F972 A5 67 LDA MLTTMP+1
4801 F974 6D 9F 02 ADC HOLD2
4802 F976 B5 67 STA MLTTMP+1
4803 F979 A6 57 LDX DINDEX \NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINIS
4804 F97B BC B1 FE LDY DHLINE.X \MULTIPLY
4805 F97E 88 CONVR1: DEY \LOOP N TIMES
4806 F97F 30 07 BMI CONVR2
4807 F981 06 66 ASL MLTTMP
4808 F983 26 67 ROL MLTTMP+1
4809 F985 4C 7E F9 JMP CONVR1
4810 F98B BC A5 FE CONVR2: LDY DIVTB.X \NOW DIVIDE HCRSR TO ACCOUNT FOR PARTIAL BYT
4811 F98B A5 55 LDA COLORS
4812 F98D A2 07 LDX #7 \* TRICKY *
4813 F98F 88 CONVR3: DEY
4814 F990 30 0A BMI CONVR4
4815 F992 CA DEX
4816 F993 46 56 LSR COLCRS+1
4817 F995 6A ROR A
4818 F996 6E A1 02 ROR TMPLBT \SAVE LOW BITS FOR MASK
4819 F999 4C BF F9 JMP CONVR3
4820 F99C CB CONVR4: INY \SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
4821 F99D 18 CLC
4822 F99E 65 66 ADC MLTTMP \ADD SHIFTED COLCRS TO MLTTMP
4823 F9A0 B5 66 STA MLTTMP
4824 F9A2 90 02 BCC CONVR5
4825 F9A4 E6 67 INC MLTTMP+1
4826 F9A6 38 CONVR5: SEC \* TRICKY *
4827 F9A7 6E A1 02 CONVR6: ROR TMPLBT \SLIDE A "1" UP AGAINST LOW BITS (CONTINUE T
4828 F9AA 18 CLC
4829 F9AB CA DEX \AND FINISH SHIFT SO LOW BITS ARE
4830 F9AC 10 9F BPL CONVR6 \RIGHT JUSTIFIED.
4831 F9AE AE A1 02 LDX TMPLBT \TMPLBT IS NOW THE INDEX INTO DMASKTB
4832 F9B1 A5 66 LDA MLTTMP \PREPARE FOR RETURN
4833 F9B3 18 CLC
4834 F9B4 65 64 ADC ADDRESS
4835 F9B6 B5 64 STA ADDRESS
4836 F9BB 85 5E STA OLDADR \REMEMBER THIS ADDRESS FOR CURSOR
4837 F9BA A5 67 LDA MLTTMP+1
4838 F9Bc 65 65 ADC ADDRESS+1
ERR LINE  ADDR  B1  B2  B3  B4
4839  F9BE  85  65
4840  F9C0  85  5F
4841  F9C2  BD  B1  FE
4842  F9C5  BD  A0  02
4843  F9CB  85  6F
4844  F9CA  68
4845  F9CB  85  56
4846  F9CD  68
4847  F9CE  85  55
4848  F9DD  68
4849  F9D1  85  54
4850  F9D3  60
4851
4852
4853
4854
4855  F9D4  A9  00
4856  F9D6  F0  02
4857  F9DB  A9  9B
4858  F9DA  85  7D
4859  F9DC  E6  63
4860  F9DE  E6  55
4861  F9E0  D0  02
4862  F9E2  E6  56
4863  F9E4  A5  55
4864  F9E6  A6  57
4865  F9EB  DD  8D  FE
4866  F9EB  FO  0B
4867  F9ED  E0  00
4868  F9EF  DO  06
4869  F9F1  C5  53
4870  F9F3  FO  02
4871  F9F5  B0  01
4872  F9F7  60
4873  F9FB  E0  0B
4874  F9FA  90  04
4875  F9FC  A5  56
4876  F9FE  FO  F7
4877  FA00  A5  57
4878  FA02  DO  30
4879  FA04  A5  63
4880  FA06  C9  51
4881  FA08  90  0A
4882  FA0A  A5  7D
4883  FA0C  FO  26
4884  FA0E  20  30  FA
4885  FA11  4C  77  FA
4886  FA14  20  34  FA
4887  FA17  A5  54
4888  FA19  1B
4889  FA1A  69  7B
4890  FA1C  20  25  FB
4891  FA1F  90  0B
4892  FA21  A5  7D

DISPLAY HANDLER -- 10-30-78 -- DISPLC

STA     ADDRESS+1
STA     OLDADR+1
LDA     DMASK, X
STA     DMASK
STA     SHFAMT
PLA
STA     COLCRS+1
PLA
STA     COLCRS
PLA
STA     ROWCRS
RTS

INCREMENT CURSOR AND DETECT BOTH END OF LINE AND END OF SCREEN

INCRSB: LDA   #0   ;NON-EXTEND ENTRY POINT
BEQ     INCRC

INCRSR: LDA   #9B   ;SPECIAL CASE ELIMINATOR
STA     INSAT
INC     LOGCOL    ;(INSCHR ENTRY POINT)

INCRS2: LDA     COLCRS   ;TEST END OF LINE
LDX    DINDEX
CMP     COLUMN, X   ;TEST TABLED VALUE FOR ALL SCREEN MODES
BEQ     INC2A   ;DO CR IF EQUAL

INCED   E0  00
CPX    #0   ;MODE 0?
BNE     INCRS3  ;IF NOT, JUST RETURN

INCFS   DO  06
BNE     INCRS3  ;IF NOT, JUST RETURN

INCFS   CS  53
CMP     RMARGN   ;TEST AGAINST RMARGN
BNE     INCRS3  ;EQUAL IS OK

INCRS3: RTC   INC2A   ;IF GREATER THAN, DO CR

INCRS3: R10

INCRS3: R10

INC2A: CPX    #8   ;CHECK MODE
BCC     DOCR1   ;NOT 320X1 SO DO IT
LDA     COLCRS+1   ;TEST MSD
BEQ     INCRS3  ;ONLY AT 64 SO DON'T DO IT

DOCRA: LDA    DINDEX   ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
BEQ     INCRS3  ;TEST LINE OVERRUN

DOCR1: LDA     DINDEX   ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
BNE     DOCR    ;ONLY AT 64 SO DON'T DO IT
LDA     LOGCOL
BNE     DOCR    ;TEST LINE OVERRUN

DOCR1B: JSR     DOCRWS   ;LOG LINE OVERFLOW IS SPECIAL CASE
JSR     INCRS1  ;RETURN
JSR     R10    ;GET IT OVER WITH

DOCR    ;TEST LOGICAL LINE BIT MAP

BNE     DOCR    ;ONLY DO LOG LINE OVERFLOW IF INSAT <>0
JSR     DOCRWS   ;LOG LINE OVERFLOW IS SPECIAL CASE

INCRS1: JSR     INCRS1  ;RETURN
JSR     R10    ;GET IT OVER WITH

INCRSI: LDA     RMARGN   ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L

INCRSA: INC
BNE     INC2A   ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L

DOCRWS: LDA     INSAT   ;DON'T EXTEND IF INSAT IS ZERO
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>DISPLC</th>
<th>B1 B2 B3 B4</th>
<th>DISPLAY</th>
<th>HANDLER</th>
<th>PAGE 110</th>
</tr>
</thead>
<tbody>
<tr>
<td>4893</td>
<td>FA23</td>
<td>BEG</td>
<td>F0 04</td>
<td>DOCR1A</td>
<td>10-30-78</td>
<td></td>
</tr>
<tr>
<td>4894</td>
<td>FA25</td>
<td>CLC</td>
<td>18</td>
<td>INSRIA</td>
<td>(INSCHR SPECIAL CASE)</td>
<td></td>
</tr>
<tr>
<td>4895</td>
<td>FA26</td>
<td>JSR</td>
<td>20 A5 F8</td>
<td>DOLCS1</td>
<td>INSERT &quot;0&quot; INTO BIT MAP</td>
<td></td>
</tr>
<tr>
<td>4896</td>
<td>FA29</td>
<td>JMP</td>
<td>4C DD F8</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4897</td>
<td>FA32</td>
<td>JMP</td>
<td>A9 00</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4898</td>
<td>FA3E</td>
<td>BEG</td>
<td>F0 02</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4899</td>
<td>FA30</td>
<td>LDA</td>
<td>A9 9B</td>
<td>DOLCS1</td>
<td>(UNCONDITIONAL)</td>
<td></td>
</tr>
<tr>
<td>4900</td>
<td>FA32</td>
<td>STA</td>
<td>85 7D</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4901</td>
<td>FA34</td>
<td>JSR</td>
<td>20 E4 FC</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4902</td>
<td>FA37</td>
<td>STA</td>
<td>A9 00</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4903</td>
<td>FA39</td>
<td>STA</td>
<td>B8 56</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4904</td>
<td>FA3B</td>
<td>INC</td>
<td>E6 54</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4905</td>
<td>FA3D</td>
<td>LDX</td>
<td>A6 77</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4906</td>
<td>FA3F</td>
<td>LDY</td>
<td>#24</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4907</td>
<td>FA41</td>
<td>BPL</td>
<td>24 7B</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4908</td>
<td>FA43</td>
<td>BNE</td>
<td>10 05</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4909</td>
<td>FA45</td>
<td>LDY</td>
<td>#4</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4910</td>
<td>FA47</td>
<td>TYA</td>
<td>98</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4911</td>
<td>FA48</td>
<td>BNE</td>
<td>D0 03</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4912</td>
<td>FA4A</td>
<td>LDA</td>
<td>BD 99 FE</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4913</td>
<td>FA4D</td>
<td>CMP</td>
<td>C5 54</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4914</td>
<td>FA4F</td>
<td>BNE</td>
<td>D0 26</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4915</td>
<td>FA51</td>
<td>STY</td>
<td>8C 9D 02</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4916</td>
<td>FA54</td>
<td>TXA</td>
<td>8A</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4917</td>
<td>FA55</td>
<td>BNE</td>
<td>D0 20</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4918</td>
<td>FA57</td>
<td>LDA</td>
<td>A5 7D</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4919</td>
<td>FA59</td>
<td>BEG</td>
<td>F0 1C</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4920</td>
<td>FA5B</td>
<td>LDA</td>
<td>C9 9B</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4921</td>
<td>FA5D</td>
<td>LDA</td>
<td>3B</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4922</td>
<td>FA5E</td>
<td>BEQ</td>
<td>F0 01</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4923</td>
<td>FA60</td>
<td>CLC</td>
<td>18</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4924</td>
<td>FA61</td>
<td>JSR</td>
<td>20 AC F8</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4925</td>
<td>FA64</td>
<td>INC</td>
<td>EE BB 02</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4926</td>
<td>FA67</td>
<td>DEC</td>
<td>C6 6C</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4927</td>
<td>FA69</td>
<td>DEC</td>
<td>CE 9D 02</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4928</td>
<td>FA6C</td>
<td>LDA</td>
<td>AD B2 02</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4929</td>
<td>FA6F</td>
<td>SEC</td>
<td>3B</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4930</td>
<td>FA70</td>
<td>LDA</td>
<td>1D EF</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4931</td>
<td>FA72</td>
<td>LDA</td>
<td>AD 9D 02</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4932</td>
<td>FA75</td>
<td>STA</td>
<td>BS 54</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4933</td>
<td>FA77</td>
<td>JMP</td>
<td>4C DD F8</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4934</td>
<td>FA83</td>
<td>JP</td>
<td>10-30-78</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4935</td>
<td>FA85</td>
<td>STA</td>
<td>95 71</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4936</td>
<td>FA87</td>
<td>STA</td>
<td>60</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4937</td>
<td>FA7A</td>
<td>SEC</td>
<td>3B</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4938</td>
<td>FA7B</td>
<td>LDA</td>
<td>B5 70</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4939</td>
<td>FA7D</td>
<td>SBC</td>
<td>E5 74</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4940</td>
<td>FA7F</td>
<td>STA</td>
<td>95 70</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4941</td>
<td>FA81</td>
<td>LDA</td>
<td>B3 71</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4942</td>
<td>FA83</td>
<td>SBC</td>
<td>E3 75</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4943</td>
<td>FA85</td>
<td>STA</td>
<td>95 71</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4944</td>
<td>FA87</td>
<td>RTS</td>
<td>60</td>
<td>DOLCS1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The document appears to be a machine code listing, likely for a computer or microcontroller, with instructions and comments in English.
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4947</td>
<td></td>
<td></td>
<td>; RANGERING: LDA $B0$SCR $; IF$ $B0$SCR$=4$</td>
<td></td>
</tr>
<tr>
<td>4948</td>
<td></td>
<td></td>
<td>; RANGE: DO CURSOR RANGE TEST. IF ERROR, POP STACK TWICE AND JMP RETURN</td>
<td></td>
</tr>
<tr>
<td>4950</td>
<td></td>
<td></td>
<td>; (ERRANGE IS EDITOR ENTRY POINT AND TEST IF EDITOR IS OPEN.</td>
<td></td>
</tr>
<tr>
<td>4951</td>
<td></td>
<td></td>
<td>; IF IT ISN'T IT OPENS THE EDITOR AND CONTINUES</td>
<td></td>
</tr>
<tr>
<td>4952</td>
<td></td>
<td></td>
<td>; RANGERING: LDA $B0$SCR $; IF$ $B0$SCR$=4$</td>
<td></td>
</tr>
<tr>
<td>4953</td>
<td>FAB8</td>
<td>AD BF 02</td>
<td>ERANGERT: LDA $B0$SCR $; IF$ $B0$SCR$=4$</td>
<td></td>
</tr>
<tr>
<td>4954</td>
<td>FAB8</td>
<td>C9 04</td>
<td>CMP $#4$</td>
<td></td>
</tr>
<tr>
<td>4955</td>
<td>FABD</td>
<td>F0 07</td>
<td>BEQ RANGERT: THEN IT IS IN MIXED MODE AND OK</td>
<td></td>
</tr>
<tr>
<td>4956</td>
<td>FA98</td>
<td>A5 57</td>
<td>LDA DINDEX $; IF$ DINDEX $=0$</td>
<td></td>
</tr>
<tr>
<td>4957</td>
<td>FA91</td>
<td>F0 03</td>
<td>BEQ RANGERT: THEN IT IS IN EDITOR MODE AND OK</td>
<td></td>
</tr>
<tr>
<td>4958</td>
<td>FA93</td>
<td>20 FC F3</td>
<td>JSR EOPEN $; IF$ NOT OPEN EDITOR</td>
<td></td>
</tr>
<tr>
<td>4959</td>
<td>FABA</td>
<td>A9 27</td>
<td>RANGE: LDA $#39$ $; <em><strong><strong>$ RANGE CHECK $RMARGN$ $</strong></strong></em>$ SET UP AC</td>
<td></td>
</tr>
<tr>
<td>4960</td>
<td>FAB9</td>
<td>C5 53</td>
<td>CMP $RMARGN$ $; <em><strong><strong>$ RANGE CHECK $RMARGN$ $</strong></strong></em>$ COMPARE</td>
<td></td>
</tr>
<tr>
<td>4961</td>
<td>FA9A</td>
<td>B0 02</td>
<td>BCS RANGE3: RANGE CHECK $RMARGN$ $*****$ BRANCH GE</td>
<td></td>
</tr>
<tr>
<td>4962</td>
<td>FA9C</td>
<td>B5 53</td>
<td>STA $RMARGN$ $; <em><strong><strong>$ RANGE CHECK $RMARGN$ $</strong></strong></em>$ BAD SO STORE</td>
<td></td>
</tr>
<tr>
<td>4963</td>
<td>FA9E</td>
<td>A6 57</td>
<td>RANGE3: LDX DINDEX</td>
<td></td>
</tr>
<tr>
<td>4964</td>
<td>FAAO</td>
<td>BD 99 FE</td>
<td>LDA NOROWS,X $; CHECK$ ROWS</td>
<td></td>
</tr>
<tr>
<td>4965</td>
<td>FAAC</td>
<td>C5 54</td>
<td>CMP $ROCRS$</td>
<td></td>
</tr>
<tr>
<td>4966</td>
<td>FAAA</td>
<td>90 2A</td>
<td>BCC RNGERR $; (ERROR$ IF TABLE.GE.ROWCRS)</td>
<td></td>
</tr>
<tr>
<td>4967</td>
<td>FA7</td>
<td>F0 28</td>
<td>BEQ RNGERR</td>
<td></td>
</tr>
<tr>
<td>4968</td>
<td>FA9A</td>
<td>E0 08</td>
<td>CPX $#8$ $; CHECK FOR 320X1</td>
<td></td>
</tr>
<tr>
<td>4969</td>
<td>FAAB</td>
<td>D0 0A</td>
<td>BNE RANGE1: SPECIAL CASE IT</td>
<td></td>
</tr>
<tr>
<td>4970</td>
<td>FAAD</td>
<td>A5 56</td>
<td>LDA COLCRS+1</td>
<td></td>
</tr>
<tr>
<td>4971</td>
<td>FAAA</td>
<td>F0 13</td>
<td>BEQ RNGOK $; IF$ HIGH BYTE IS 0, COL IS OK</td>
<td></td>
</tr>
<tr>
<td>4972</td>
<td>FAB1</td>
<td>C9 01</td>
<td>CMP $#1$</td>
<td></td>
</tr>
<tr>
<td>4973</td>
<td>FABA</td>
<td>D0 1C</td>
<td>BNE RNGERR $; IF$ &gt;1, BAD</td>
<td></td>
</tr>
<tr>
<td>4974</td>
<td>FAB5</td>
<td>F0 04</td>
<td>BEQ RANGE2: IF 1, GO CHECK LOW BYTE</td>
<td></td>
</tr>
<tr>
<td>4975</td>
<td>FAB7</td>
<td>A5 56</td>
<td>RANGE1: LDA COLCRS+1 $; FOR$ OTHERS, NON-ZERO HIGH BYTE IS BAD</td>
<td></td>
</tr>
<tr>
<td>4976</td>
<td>FAB9</td>
<td>D0 16</td>
<td>BNE RNGERR</td>
<td></td>
</tr>
<tr>
<td>4977</td>
<td>FABB</td>
<td>BD BD FE</td>
<td>RANGE2: LDA COLUMN,X $; CHECK$ LOW BYTE</td>
<td></td>
</tr>
<tr>
<td>4978</td>
<td>FABE</td>
<td>C5 55</td>
<td>CMP $COLCRS$</td>
<td></td>
</tr>
<tr>
<td>4979</td>
<td>FAC0</td>
<td>90 0F</td>
<td>BCC RNGERR</td>
<td></td>
</tr>
<tr>
<td>4980</td>
<td>FAC2</td>
<td>F0 0D</td>
<td>BEQ RNGERR</td>
<td></td>
</tr>
<tr>
<td>4981</td>
<td>FAC4</td>
<td>A9 01</td>
<td>RNGOK: LDA $#SUCES$ $; SET$ STATUS OK</td>
<td></td>
</tr>
<tr>
<td>4982</td>
<td>FAC6</td>
<td>85 4C</td>
<td>STA DSTAT</td>
<td></td>
</tr>
<tr>
<td>4983</td>
<td>FAC8</td>
<td>A9 80</td>
<td>LDA $#BRKABT$ $; PREPARE$ BREAK ABORT STATUS</td>
<td></td>
</tr>
<tr>
<td>4984</td>
<td>FAC9</td>
<td>A6 11</td>
<td>LDX BRKKEY $; CHECK$ BREAK KEY FLAG</td>
<td></td>
</tr>
<tr>
<td>4985</td>
<td>FAC9</td>
<td>B5 11</td>
<td>STA BRKKEY $; 'CLEAR' BREAK</td>
<td></td>
</tr>
<tr>
<td>4986</td>
<td>FACE</td>
<td>F0 06</td>
<td>BEQ RNGERR2 $; IF$ BREAK, QUIT IMMEDIATELY AND RETURN TO CI</td>
<td></td>
</tr>
<tr>
<td>4987</td>
<td>FADO</td>
<td>BD 60</td>
<td>RTS</td>
<td></td>
</tr>
<tr>
<td>4988</td>
<td>FADF</td>
<td>20 D6 F7</td>
<td>RNGERRR: JSR HOME $; ON$ RANGE ERROR, BRING CURSOR BACK</td>
<td></td>
</tr>
<tr>
<td>4989</td>
<td>FAD4</td>
<td>A9 BD</td>
<td>LDA $#CRSRO$ $; SHOW$ CURSOR OVER RANGE ERROR</td>
<td></td>
</tr>
<tr>
<td>4990</td>
<td>FADD</td>
<td>85 4C</td>
<td>RNGERR2: STA DSTAT</td>
<td></td>
</tr>
<tr>
<td>4991</td>
<td>FADD</td>
<td>BD 68</td>
<td>RNGERRR: JSR HOME $; ON$ RANGE ERROR, BRING CURSOR BACK</td>
<td></td>
</tr>
<tr>
<td>4992</td>
<td>FADD</td>
<td>68</td>
<td>PLA $; RESTORE$ STACK (THIS ROUTINE IS ALWAYS 1 Lev</td>
<td></td>
</tr>
<tr>
<td>4993</td>
<td>FABA</td>
<td>A5 78</td>
<td>LDA $SWPFL0$ $; IF$ SWAPPED, SWAP BACK</td>
<td></td>
</tr>
<tr>
<td>4994</td>
<td>FADC</td>
<td>C0 03</td>
<td>BPL RETUR3</td>
<td></td>
</tr>
<tr>
<td>4995</td>
<td>FADE</td>
<td>20 B9 FC</td>
<td>JSR SWAPA $; AND$ DON'T DO RETUR1</td>
<td></td>
</tr>
<tr>
<td>4996</td>
<td>FAE1</td>
<td>4C 34 F6</td>
<td>RETUR3: JMP RETUR1 $; RETURN$ TO CI</td>
<td></td>
</tr>
<tr>
<td>4997</td>
<td></td>
<td></td>
<td>; OFFCRS: RESTORE OLD DATA UNDER CURSOR SO IT CAN BE MOVED</td>
<td></td>
</tr>
<tr>
<td>4998</td>
<td></td>
<td></td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>4999</td>
<td></td>
<td></td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td></td>
<td></td>
<td>;</td>
<td></td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC PAGE 112

5001
5002 FAE4 A0 00 ; OFFCRS: LDY #0
5003 FAE6 A5 5D ; LDA OLDCHR
5004 FAEB 91 5E ; STA (OLDADR),Y
5005 FAEA 60 ; RTS
5006
5007
5008
5009
5010
5011
5012 ; BITMAP Routines:
5013
5014 ; BITCON: PUT MASK IN BITMSK AND INDEX IN X
5015
5016 ; BITPUT: PUT CARRY INTO BITMAP
5017 ; BITROL: ROL CARRY INTO BOTTOM OF BITMAP (SCROLL)
5018 ; BITSET: SET PROPER BIT
5019 ; BITCLR: CLEAR PROPER BIT
5020 ; BITGET: RETURN CARRY SET IF BIT IS THERE
5021 ; LOGGET: DO BITGET FOR LOGMAP INSTEAD OF TABMAP
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032 ; BITCON: PHA
5033
5034
5035
5036
5037
5038 ; BITROL: ROL LOGMAP+2
5039 ; BITPUT: BCC BITCLR ;AND RETURN
5040 ; OTHERWISE FALL THROUGH TO BITSET AND RETURN
5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054 ; LOGGET: LDA ROWCRS
5055 ; LOGGET: CLC
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5055</td>
<td>FB23</td>
<td>69 78</td>
<td>LOG2GET: ADC #120</td>
</tr>
<tr>
<td>5056</td>
<td>FB25</td>
<td>20 EB FA</td>
<td>BITGET: JSR BITCON</td>
</tr>
<tr>
<td>5057</td>
<td>FB28</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>5058</td>
<td>FB29</td>
<td>BD A3 02</td>
<td></td>
</tr>
<tr>
<td>5059</td>
<td>FB2C</td>
<td>25 6E</td>
<td></td>
</tr>
<tr>
<td>5060</td>
<td>FB2E</td>
<td>F0 01</td>
<td></td>
</tr>
<tr>
<td>5061</td>
<td>FB30</td>
<td>38 SEC</td>
<td></td>
</tr>
<tr>
<td>5062</td>
<td>FB31</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>5063</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5064</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5065</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5066</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5067</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5068</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5069</td>
<td>FB32</td>
<td>AD FA 02</td>
<td></td>
</tr>
<tr>
<td>5070</td>
<td>FB35</td>
<td>A4 57</td>
<td></td>
</tr>
<tr>
<td>5071</td>
<td>FB37</td>
<td>C0 03</td>
<td></td>
</tr>
<tr>
<td>5072</td>
<td>FB39</td>
<td>B0 0F</td>
<td></td>
</tr>
<tr>
<td>5073</td>
<td>FB3B</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>5074</td>
<td>FB3C</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>5075</td>
<td>FB3D</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>5076</td>
<td>FB3E</td>
<td>2A</td>
<td></td>
</tr>
<tr>
<td>5077</td>
<td>FB3F</td>
<td>29 03</td>
<td></td>
</tr>
<tr>
<td>5078</td>
<td>FB41</td>
<td>AA</td>
<td></td>
</tr>
<tr>
<td>5079</td>
<td>FB42</td>
<td>AD FA 02</td>
<td></td>
</tr>
<tr>
<td>5080</td>
<td>FB45</td>
<td>29 9F</td>
<td></td>
</tr>
<tr>
<td>5081</td>
<td>FB47</td>
<td>1D FA FE</td>
<td></td>
</tr>
<tr>
<td>5082</td>
<td>FB4A</td>
<td>8D FB 02</td>
<td></td>
</tr>
<tr>
<td>5083</td>
<td>FB4D</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>5084</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5085</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5086</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5087</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5088</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5089</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5090</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5091</td>
<td>FB4E</td>
<td>A9 02</td>
<td></td>
</tr>
<tr>
<td>5092</td>
<td>FB50</td>
<td>B5 05</td>
<td></td>
</tr>
<tr>
<td>5093</td>
<td>FB52</td>
<td>A9 47</td>
<td></td>
</tr>
<tr>
<td>5094</td>
<td>FB54</td>
<td>B5 64</td>
<td></td>
</tr>
<tr>
<td>5095</td>
<td>FB56</td>
<td>A0 27</td>
<td></td>
</tr>
<tr>
<td>5096</td>
<td>FB58</td>
<td>B1 66</td>
<td></td>
</tr>
<tr>
<td>5097</td>
<td>FB5A</td>
<td>B5 50</td>
<td></td>
</tr>
<tr>
<td>5098</td>
<td>FB5C</td>
<td>B1 68</td>
<td></td>
</tr>
<tr>
<td>5099</td>
<td>FB5E</td>
<td>91 66</td>
<td></td>
</tr>
<tr>
<td>5100</td>
<td>FB60</td>
<td>A5 50</td>
<td></td>
</tr>
<tr>
<td>5101</td>
<td>FB62</td>
<td>91 64</td>
<td></td>
</tr>
<tr>
<td>5102</td>
<td>FB64</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>5103</td>
<td>FB65</td>
<td>10 F1</td>
<td></td>
</tr>
<tr>
<td>5104</td>
<td>FB67</td>
<td>A5 63</td>
<td></td>
</tr>
<tr>
<td>5105</td>
<td>FB69</td>
<td>B5 69</td>
<td></td>
</tr>
<tr>
<td>5106</td>
<td>FB6B</td>
<td>A5 64</td>
<td></td>
</tr>
<tr>
<td>5107</td>
<td>FB6D</td>
<td>B5 68</td>
<td></td>
</tr>
<tr>
<td>5108</td>
<td>FB6F</td>
<td>1B</td>
<td></td>
</tr>
</tbody>
</table>

; INATAC: INTERNAL(CHAR) TO ATASCII(ATCHR) CONVERSION

; INATAC: LDA CHAR
; LDY DINDEX ;IF GRAPHICS MODES
; CPY #3
; BCS INATAC ;THEN DON'T CHANGE CHAR
; ROL A
; ROL A
; ROL A
; ROL A
; AND #3
; TAX
; LDA CHAR
; AND #9F
; ORA INTATA,X
; STA ATACHR
; RTS

; MOVLIN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOADR
; DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
; BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40

; MOVLIN: LDA #LINBUF/256 ;SET UP ADDRESS=LINBUF=$247
; STA ADRESS+1
; LDA #LINBUF. AND. #$FF
; STA ADRESS
; LDY #39
; MOVLIN: LDA (TOADR), Y ;SAVE TO DATA
; STA TMPCHR
; LDA (FRMADR), Y ;STORE DATA
; STA (TOADR), Y
; LDA TMPCHR
; LDA (ADDRESS), Y
; DEY
; BPL MOVLIN
; LDA ADRESS+1 ;SET UP FRMADR=LAST LINE
; STA FRMADR+1
; LDA ADRESS
; STA FRMADR
; CLC ;ADD 40 TO TOADR
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5109</td>
<td>F870</td>
<td>A5</td>
<td>66</td>
<td>A5</td>
<td>66</td>
<td>LDA TOADR</td>
</tr>
<tr>
<td>5110</td>
<td>F872</td>
<td>69</td>
<td>28</td>
<td>69</td>
<td>28</td>
<td>ADC #40</td>
</tr>
<tr>
<td>5111</td>
<td>F874</td>
<td>85</td>
<td>66</td>
<td>85</td>
<td>66</td>
<td>STA TOADR</td>
</tr>
<tr>
<td>5112</td>
<td>F876</td>
<td>90</td>
<td>02</td>
<td>90</td>
<td>02</td>
<td>BCC MOVI2</td>
</tr>
<tr>
<td>5113</td>
<td>F878</td>
<td>E6</td>
<td>67</td>
<td>E6</td>
<td>67</td>
<td>INC TOADR+1</td>
</tr>
<tr>
<td>5114</td>
<td>F87A</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>MOVLI2: RTS</td>
</tr>
<tr>
<td>5115</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5116</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5117</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5118</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>; EXTEND: EXTEND BIT MAP FROM ROWCRS (EXTEND LOGICAL LINE)</td>
</tr>
<tr>
<td>5119</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>5120</td>
<td>F87B</td>
<td>08</td>
<td></td>
<td></td>
<td></td>
<td>EXTEND: PHP ; SAVE CARRY</td>
</tr>
<tr>
<td>5121</td>
<td>F87C</td>
<td>A0</td>
<td>17</td>
<td>A0</td>
<td>17</td>
<td>EXEN1: TYA</td>
</tr>
<tr>
<td>5122</td>
<td>F87E</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5123</td>
<td>F87F</td>
<td>20</td>
<td>22</td>
<td>20</td>
<td>22</td>
<td>JSR L01GET</td>
</tr>
<tr>
<td>5124</td>
<td>F880</td>
<td>08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5125</td>
<td>F883</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5126</td>
<td>F884</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5127</td>
<td>F885</td>
<td>69</td>
<td>79</td>
<td>69</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>5128</td>
<td>F887</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5129</td>
<td>F88B</td>
<td>20</td>
<td>04</td>
<td>20</td>
<td>04</td>
<td>JSR BITPUT</td>
</tr>
<tr>
<td>5130</td>
<td>F888</td>
<td>88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5131</td>
<td>F88C</td>
<td>30</td>
<td>04</td>
<td>30</td>
<td>04</td>
<td>BMI EXEN4</td>
</tr>
<tr>
<td>5132</td>
<td>F88E</td>
<td>C4</td>
<td>54</td>
<td>C4</td>
<td>54</td>
<td>CPY ROWCRS</td>
</tr>
<tr>
<td>5133</td>
<td>F890</td>
<td>80</td>
<td>EC</td>
<td>80</td>
<td>EC</td>
<td>BCS EXEN1</td>
</tr>
<tr>
<td>5134</td>
<td>F892</td>
<td>A5</td>
<td>54</td>
<td>A5</td>
<td>54</td>
<td>EXEN3: LDA ROWCRS</td>
</tr>
<tr>
<td>5135</td>
<td>F894</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5136</td>
<td>F895</td>
<td>69</td>
<td>78</td>
<td>69</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>5137</td>
<td>F897</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5138</td>
<td>F89B</td>
<td>4C</td>
<td>04</td>
<td>4C</td>
<td>04</td>
<td>JSR BITPUT : STORE NEW LINE'S BIT AND RETURN</td>
</tr>
<tr>
<td>5139</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5140</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5141</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5142</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>; CLRLIN: CLEAR LINE CURSOR IS ON</td>
</tr>
<tr>
<td>5143</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5144</td>
<td>F89B</td>
<td>A5</td>
<td>52</td>
<td>A5</td>
<td>52</td>
<td>CLRLIN: LDA LMAGRN</td>
</tr>
<tr>
<td>5145</td>
<td>F89D</td>
<td>85</td>
<td>55</td>
<td>85</td>
<td>55</td>
<td>STA COLCRS</td>
</tr>
<tr>
<td>5146</td>
<td>F89F</td>
<td>20</td>
<td>47</td>
<td>20</td>
<td>47</td>
<td>JSR CONVRT</td>
</tr>
<tr>
<td>5147</td>
<td>F8A2</td>
<td>A0</td>
<td>27</td>
<td>A0</td>
<td>27</td>
<td>LDY #39</td>
</tr>
<tr>
<td>5148</td>
<td>F8A4</td>
<td>A9</td>
<td>00</td>
<td>A9</td>
<td>00</td>
<td>LDA #0</td>
</tr>
<tr>
<td>5149</td>
<td>F8A6</td>
<td>91</td>
<td>64</td>
<td>91</td>
<td>64</td>
<td>CLRLI1: STA (ADRESS),Y</td>
</tr>
<tr>
<td>5150</td>
<td>F8AB</td>
<td>98</td>
<td></td>
<td></td>
<td></td>
<td>DEA</td>
</tr>
<tr>
<td>5151</td>
<td>F8A9</td>
<td>10</td>
<td>FB</td>
<td>10</td>
<td>FB</td>
<td>BPL CLRLI1</td>
</tr>
<tr>
<td>5152</td>
<td>F8AB</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>RTS</td>
</tr>
<tr>
<td>5153</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5154</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5155</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5156</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>; SCROLL: SCROLL SCREEN</td>
</tr>
<tr>
<td>5157</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>5158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>5159</td>
<td>F8AC</td>
<td>20</td>
<td>FA</td>
<td>20</td>
<td>FA</td>
<td>SCROLL: JSR BITROL ; ROLL IN CARRY</td>
</tr>
<tr>
<td>5160</td>
<td>F8AF</td>
<td>A5</td>
<td>58</td>
<td>A5</td>
<td>58</td>
<td>LDA SAVMSC</td>
</tr>
<tr>
<td>5161</td>
<td>F8B1</td>
<td>85</td>
<td>64</td>
<td>85</td>
<td>64</td>
<td>STA ADDRESS</td>
</tr>
<tr>
<td>5162</td>
<td>F8B3</td>
<td>A5</td>
<td>59</td>
<td>A5</td>
<td>59</td>
<td>LDA SAVMSC+1</td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4   DISPLAY HANDLER -- 10-30-78 -- DISPLC   PAGE 115

5163 FBB5 85 65   STA ADDRESS+1
5164 FBB7 A0 28   SCROLL1: LDY #40, LOOP
5165 FBB9 B1 64   LDA (ADDRESS), Y
5166 FBBB A6 6A   LDX ADDRESS, RAMTOP  ; TEST FOR LAST LINE
5167 FBBE CA 6E   DEX
5168 FBBE E4 65   CPX ADDRESS+1
5169 FBC0 D0 08   DISPLC
5170 FBC2 A2 D7   LDX #D7
5171 FBC4 E4 64   CPX ADDRESS
5172 FBC6 B0 02   BCS SCROLL2
5173 FBC8 A9 00   LDA #0  ; YES SO STORE ZERO DATA FOR THIS ENTIRE LINE
5174 FBCA A0 00   SCROLL2: LDY #0
5175 FBCC 91 64   STA (ADDRESS), Y
5176 FBCF E6 64   INC ADDRESS
5177 FBDD D0 E5   BNE SCROLL1
5178 FBDF E6 65   INC ADDRESS+1
5179 FBD4 A5 65   LDA ADDRESS+1
5180 FBD6 C5 6A   CMP RAMTOP
5181 FBD8 D0 DD   BNE SCROLL1
5182 FBD9 4C DD FB   JMP DOLCOL  ; AND RETURN
5183
5184
5185  ; DOLCOL: DO LOGICAL COLUMN FROM BITMAP AND COLCRS
5186
5187 FBD9 A9 00   DOLCOL: LDA #0  ; START WITH ZERO
5188 FBDF 85 63   STA LOGCOL
5189 FBE1 A5 54   LDA ROWCRS
5190 FBE3 B5 51   STA H0L01
5191 FBE5 A5 51   DOLCO1: LDA H0L01  ; ADD IN ROW COMPONENT
5192 FBE7 20 22 FB   JSR LO1SET
5193 FBEA B0 0C   BCS DOLCO2  ; FOUND BEGINNING OF LINE
5194 FBEF A5 63   LDA LOGCOL  ; ADD 40 AND LOOK BACK ONE
5195 FBEE 1B   CLC
5196 FBF0 69 28   ADC #40
5197 FBF1 B5 63   STA LOGCOL
5198 FBF3 C6 51   DEC H0L01  ; UP ONE LINE
5199 FBF5 4C E5 FB   JMP DOLCO1
5200 FBFB 18   DOLCO2: CLC  ; ADD IN COLCRS
5201 FBF9 A5 63   LDA LOGCOL
5202 FBFB 65 55   ADC COLCRS
5203 FBFD B5 63   STA LOGCOL
5204 FBFF 60   RTS
5205
5206
5207
5208  ; DOBUF: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
5209  ; BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
5210
5211 FC00 20 9D FC   DOBUF: JSR PHACRS
5212 FC03 A5 63   LDA LOGCOL
5213 FC05 4B   PHA
5214 FC06 A5 6C   LDA BUFSTR  ; START
5215 FC08 B5 54   STA ROWCRS
5216 FC0A A5 6D   LDA BUFSTR+1
ERR LINE ADDR B1 B2 B3 B4
5217 FCOC B5 55 STA COLCRS
5218 FC0E A9 01 LDA #1
5219 FC10 B5 68 STA BUFCNT
5220 FC12 A2 17 DOBUF1: LDX #23 NORMAL
5221 FC14 A5 7B LDA SWPFLO IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREE
5222 FC16 10 02 BPL DOB1
5223 FC18 A2 03 LDX #3
5224 FC1A E4 54 DOB1: CPX ROWCRS TEST IF CRSR IS AT LAST SCREEN POSITION
5225 FC1C D0 0B BNE DOBU1A
5226 FC1E A5 55 LDA COLCRS
5227 FC20 C5 53 CMP RMARGN
5228 FC22 D0 05 BNE DOBU1A
5229 FC24 E6 6B INC BUFCNT YES, SO FAKE INCRSR TO AVOID SCROLLING
5230 FC26 4C 39 FC JMP DOBUF2
5231 FC29 20 D4 F9 DOBU1A: JSR INCRSB
5232 FC2C E6 6B INC BUFCNT
5233 FC2E A5 63 LDA LOGCOL
5234 FC30 C5 52 CMP LMARGN
5235 FC32 D0 DE BNE DOBUF1 NOT YET EOL
5236 FC34 C6 54 DEC ROWCRS BACK UP ONE INCRsr
5237 FC36 20 99 F7 JSR CRSLF
5238 FC39 20 A2 F5 DOBUF2: JSR GETPLT TEST CURRENT COLUMN FOR NON-ZERO DATA
5239 FC3C D0 17 BNE DOBUF4 QUIT IF NON-ZERO
5240 FC3E C6 68 DEC BUFcnt DECREMENT COUNTER
5241 FC40 A5 63 LDA LOGCOL BEGINNING OF LOGICAL LINE YET?
5242 FC42 C5 52 CMP LMARGN
5243 FC44 F0 0F BEQ DOBUF4 YES, SO QUIT
5244 FC46 20 99 F7 JSR CRSLF BACK UP CURSOR
5245 FC49 A5 55 LDA COLCRS IF LOGCOL=RMARGN, GO UP 1 ROW
5246 FC4B C5 53 CMP RMARGN
5247 FC4D D0 02 BNE DOBUF3
5248 FC4F C6 54 DEC ROWCRS
5249 FC51 A5 68 DOBUF3: LDA BUFcnt LOOP UNLESS BUFcnt JUST WENT TO ZERO
5250 FC53 D0 E4 BNE DOBUF2
5251 FC55 68 DOBUF4: PLA
5252 FC56 B5 63 STA LOGCOL
5253 FC58 20 A8 FC JSR PLACRS
5254 FC5B 40 RTS
5255 ;
5256 ;
5257 ;
5258 ;
5259 ;
5260 ;
5261 FC5C 20 DD FB STRBEG: MOVE BUFSTR TO BEGINNING OF LOGICAL LINE.
5262 FC5F A5 51 JSR DOLCOL USE DOLCOL TO POINT HOLD1 AT BOL
5263 FC61 B5 6C LDA HOLD1
5264 FC63 A5 52 STA BUFSTR
5265 FC65 B5 6D STA BUFSTR+1
5266 FC67 60 RTS
5267 ;
5268 ;
5269 ;
5270 ;
ERR LINE ADDR B1 B2 B3 B4 | DISPLAY HANDLER -- 10-30-78 -- DISPLC
---|---
5271 | ; DELTIM: TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
5272 | ; DELTIA: LDA LOGCOL ; IF LOGCOL<>LMARGIN
5273 | CMP LMARGIN ; THEN DONT MOVE UP ONE
5274 FC68 A5 63 | DEC ROWCRS
5275 FC6A C5 52 | BNE DELTIB ; LINE BEFORE TESTING DELTIM
5276 FC6C D0 02 | DEC LMARGN
5277 FC6E C6 54 | BPL DELTIB ; DELETE A LINE AND RETURN
5278 FC70 20 DD FB | JSR DOLCOL
5279 FC73 A5 63 | JSR CONVRT
5280 FC75 C5 52 | JSR CONVRT
5281 FC77 F0 13 | JSR CONVRT
5282 FC79 40 47 F9 | JSR CONVRT
5283 FC7C A5 53 | JSR CONVERT
5284 FC7E 38 | JSR CONVERT
5285 FC7F E5 52 | JSR CONVERT
5286 FC81 A8 | JSR CONVERT
5287 FC82 B1 64 | JSR CONVERT
5288 FC84 D0 06 | JSR CONVERT
5289 FC86 B8 | JSR CONVERT
5290 FC87 10 F9 | JSR CONVERT
5291 FC89 4C DB FB | JSR CONVERT
5292 FC8C 60 | JSR CONVERT
5293 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5294 | FB DELTIB: RTS
5295 | FB DELTIB: RTS
5296 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5297 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5298 FC8D A2 2D | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5299 FC8F BD C6 FE | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5300 FC92 CD FB 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5301 FC95 F0 05 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5302 FC97 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5303 FC98 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5304 FC99 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5305 FC9A 10 F3 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5306 FC9C 60 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5307 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5308 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5309 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5310 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5311 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5312 FC9D A2 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5313 FC9F B5 54 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5314 FCA1 9D B8 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5315 FCA4 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5316 FCA5 10 F8 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5317 FCA7 60 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5318 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5319 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5320 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5321 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5322 FCA8 A2 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5323 FCA9 BD B8 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5324 FCAD 95 54 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN

ERR LINE ADDR B1 B2 B3 B4 | DISPLAY HANDLER -- 10-30-78 -- DISPLC
---|---
5271 | ; DELTIM: TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
5272 | ; DELTIM: IF LOGCOL<>LMARGIN
5273 | CMP LMARGIN ; THEN DONT MOVE UP ONE
5274 FC68 A5 63 | DEC ROWCRS
5275 FC6A C5 52 | BNE DELTIB ; LINE BEFORE TESTING DELTIM
5276 FC6C D0 02 | DEC LMARGN
5277 FC6E C6 54 | BPL DELTIB ; DELETE A LINE AND RETURN
5278 FC70 20 DD FB | JSR DOLCOL
5279 FC73 A5 63 | JSR CONVRT
5280 FC75 C5 52 | JSR CONVRT
5281 FC77 F0 13 | JSR CONVRT
5282 FC79 40 47 F9 | JSR CONVRT
5283 FC7C A5 53 | JSR CONVERT
5284 FC7E 38 | JSR CONVERT
5285 FC7F E5 52 | JSR CONVERT
5286 FC81 A8 | JSR CONVERT
5287 FC82 B1 64 | JSR CONVERT
5288 FC84 D0 06 | JSR CONVERT
5289 FC86 B8 | JSR CONVERT
5290 FC87 10 F9 | JSR CONVERT
5291 FC89 4C DB FB | JSR CONVERT
5292 FC8C 60 | JSR CONVERT
5293 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5294 | FB DELTIB: RTS
5295 | FB DELTIB: RTS
5296 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5297 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5298 FC8D A2 2D | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5299 FC8F BD C6 FE | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5300 FC92 CD FB 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5301 FC95 F0 05 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5302 FC97 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5303 FC98 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5304 FC99 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5305 FC9A 10 F3 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5306 FC9C 60 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5307 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5308 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5309 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5310 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5311 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5312 FC9D A2 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5313 FC9F B5 54 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5314 FCA1 9D B8 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5315 FCA4 CA | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5316 FCA5 10 F8 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5317 FCA7 60 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5318 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5319 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5320 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5321 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5322 FCA8 A2 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5323 FCA9 BD B8 02 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
5324 FCAD 95 54 | DELTII: JMP DELLIB ; DELETE A LINE AND RETURN
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5325</td>
<td>FCAF</td>
<td>CA</td>
<td>DEX</td>
</tr>
<tr>
<td>5326</td>
<td>FCB0</td>
<td>10 F8</td>
<td>BPL</td>
</tr>
<tr>
<td>5327</td>
<td>FCB2</td>
<td>60</td>
<td>RTS</td>
</tr>
<tr>
<td>5328</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5329</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5330</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5331</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5332</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5333</td>
<td>FCB3</td>
<td>20 B9 FC</td>
<td>SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS</td>
</tr>
<tr>
<td>5334</td>
<td>FCB6</td>
<td>4C 34 F6</td>
<td>SWAP: JSR SWAP1, THIS ENTRY POINT DOES RETURN</td>
</tr>
<tr>
<td>5335</td>
<td>FCB9</td>
<td>AD BF 02</td>
<td>SWAPA: LDA BOTSCR</td>
</tr>
<tr>
<td>5336</td>
<td>FCBC</td>
<td>CA 18</td>
<td>CMP #24</td>
</tr>
<tr>
<td>5337</td>
<td>FCBE</td>
<td>F0 00</td>
<td>BPL SWAP3</td>
</tr>
<tr>
<td>5338</td>
<td>FCO0</td>
<td>A2 00</td>
<td>LDX #11</td>
</tr>
<tr>
<td>5339</td>
<td>FCO2</td>
<td>B5 54</td>
<td>SWAP1: LDA ROWCRS,X</td>
</tr>
<tr>
<td>5340</td>
<td>FCO4</td>
<td>4B</td>
<td>PHA</td>
</tr>
<tr>
<td>5341</td>
<td>FCO5</td>
<td>BD 00 02</td>
<td>LDA TXTROW.X</td>
</tr>
<tr>
<td>5342</td>
<td>FCOB</td>
<td>95 54</td>
<td>STA ROWCRS,X</td>
</tr>
<tr>
<td>5343</td>
<td>FCOA</td>
<td>9B #8</td>
<td>PLA</td>
</tr>
<tr>
<td>5344</td>
<td>FCOB</td>
<td>9D 00 02</td>
<td>STA TXTROW,X</td>
</tr>
<tr>
<td>5345</td>
<td>FCCE</td>
<td>CA</td>
<td>DEX</td>
</tr>
<tr>
<td>5346</td>
<td>FCF0</td>
<td>10 F1</td>
<td>BPL SWAP1</td>
</tr>
<tr>
<td>5347</td>
<td>FCD1</td>
<td>A5 7B</td>
<td>LDA SWPFLG</td>
</tr>
<tr>
<td>5348</td>
<td>FCD3</td>
<td>49 FF</td>
<td>EOR #FF</td>
</tr>
<tr>
<td>5349</td>
<td>FCD5</td>
<td>B5 7B</td>
<td>STA SWPFLG</td>
</tr>
<tr>
<td>5350</td>
<td>FCD7</td>
<td>60</td>
<td>SWAP3: RTS</td>
</tr>
<tr>
<td>5351</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5352</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5353</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5354</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5355</td>
<td>FCD8</td>
<td>A2 7F</td>
<td>CLICK: LDX #7F</td>
</tr>
<tr>
<td>5356</td>
<td>FCDA</td>
<td>AE IF 0D</td>
<td>CLICK1: STX CONSOl</td>
</tr>
<tr>
<td>5357</td>
<td>FCDD</td>
<td>AE 0A D4</td>
<td>STX WSYNC</td>
</tr>
<tr>
<td>5358</td>
<td>FCE0</td>
<td>CA</td>
<td>DEX</td>
</tr>
<tr>
<td>5359</td>
<td>FCE1</td>
<td>10 F7</td>
<td>BPL CLICK1</td>
</tr>
<tr>
<td>5360</td>
<td>FCE3</td>
<td>60</td>
<td>RTS</td>
</tr>
<tr>
<td>5361</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5362</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5363</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5364</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5365</td>
<td>FCE4</td>
<td>A9 00</td>
<td>COlCR: LDA #0</td>
</tr>
<tr>
<td>5366</td>
<td>FCE6</td>
<td>A6 7B</td>
<td>LDX SWPFLG</td>
</tr>
<tr>
<td>5367</td>
<td>FCEB</td>
<td>D0 04</td>
<td>BNE COlCR1</td>
</tr>
<tr>
<td>5368</td>
<td>FCEA</td>
<td>A6 57</td>
<td>LDX DINDEX</td>
</tr>
<tr>
<td>5369</td>
<td>FCEC</td>
<td>D0 02</td>
<td>BNE COlCR2</td>
</tr>
<tr>
<td>5370</td>
<td>FCEE</td>
<td>A5 52</td>
<td>COlCR1: LDA LMARGN</td>
</tr>
<tr>
<td>5371</td>
<td>FCF0</td>
<td>85 55</td>
<td>COlCR2: STA COlCR5</td>
</tr>
<tr>
<td>5372</td>
<td>FCF2</td>
<td>60</td>
<td>RTS</td>
</tr>
<tr>
<td>5373</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5374</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5375</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5376</td>
<td>;</td>
<td>;</td>
<td>;</td>
</tr>
<tr>
<td>5377</td>
<td>FCF3</td>
<td>A5 5A</td>
<td>PUTMSC: PUT SAVMSC INTO ADDRESS</td>
</tr>
<tr>
<td>5378</td>
<td>FCF5</td>
<td>85 64</td>
<td>LDA SAVMSC</td>
</tr>
</tbody>
</table>

CLICK: MAKE CLICK THROUGH KEYBOARD SPEAKER

COlCR: PUTS EITHER 0 OR LMARGN INTO COlCRS BASED ON MODE AND SWPFLG

COlCR: LDA #0
LDX SWPFLG
BNE COlCR1
LDX DINDEX
BNE COlCR2
LDA LMARGN
STA COlCR5
RTS

PUTMSC: PUT SAVMSC INTO ADDRESS

LDA SAVMSC
STA ADDRESS
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5379</td>
<td>FCF7</td>
<td>A5</td>
<td>59</td>
<td></td>
<td></td>
<td>LDA SAVMSC+1</td>
</tr>
<tr>
<td>5380</td>
<td>FCF9</td>
<td>85</td>
<td>65</td>
<td></td>
<td></td>
<td>STA ADDRESS+1</td>
</tr>
<tr>
<td>5381</td>
<td>FCFB</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td>RTS</td>
</tr>
<tr>
<td>5382</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DRAW -- DRAW A LINE FROM OLDROW, OLDCOL TO NEWROW, NEWCOL

T (THE AL MILLER METHOD FROM BASKETBALL)

5383 FCF  A2  00      DRAW:  LDX #0
5389 FCFC A5  22     LDA  ICOMMZ  ;TEST COMMAND: $11=DRAW $12=FILL
5390 FDO0 C9  11     CMP  #$11
5391 FDO1 F0  08     BEQ  DRAW
5392 FDO4 C9  12     CMP  #$12  ;TEST FILL
5393 FDO5 F0  03     BEQ  DRAWB  ;YES
5394 FDO8 A0  B4     LDY  #IVALID  ;NO, SO RETURN INVALID COMMAND
5395 FDOA 60      RTS
5396 FDOB E8     DRAWB:  INX
5397 FD0C BE  B7  02   DRAWA:  STX  FILFLG
5398 FD0D A5  54     LDA  ROWCRS  ;PUT CURSOR INTO NEWROW, NEWCOL
5402 FD17 A5  56     LDA  COLCRS+1
5403 FD19 B5  62     STA  NEWCOL+1
5404 FD1B 38     SEC
5408 FD22 A5  60     LDA  NEWCOL  ;DETERMINE DELTA ROW
5411 FD2B B0  0D     BCS  DRAW1  ;DO DIRECTION AND ABSOLUTE VALUE
5412 FD2C 89  FF     LDA  #$FF  ;BORROW WAS ATTEMPTED
5413 FD2E 89  FF     STA  ROWINC  ;SET DIRECTION=DOWN
5414 FD30 49  FF     EOR  #$FF  ;DELTAR = DELTAR!
5416 FD35 1B     CLC
5417 FD3D 69  01     ADC  #1
541B FD3F B5  76     STA  DELTAR
5420 FD37 38     SEC
5421 FD3A E5  5B     SBC  OLDROW
5422 FD3C B5  77     STA  DELTAR
5423 FD3E A5  62     LDA  NEWCOL+1  ;TWO-BYTE QUANTITY
5424 FD40 E5  5C     SBC  OLDCOL+1
5425 FD42 B5  78     STA  DELTAC+1
5426 FD44 B0  14     BCS  DRAW2  ;DIRECTION AND ABSOLUTE VALUE
5427 FD46 A9  FF     LDA  #$FF  ;BORROW WAS ATTEMPTED
5428 FD47 B5  7A     STA  COLINC  ;SET DIRECTION = LEFT
5429 FD49 A5  77     LDA  DELTAC
5430 FD4C 49  FF     EOR  #$FF  ;DELTAC = DELTAC:
5431 FD4E B5  77     STA  DELTAC
5432 FD50 A5  78     LDA  DELTAC+1
5433 FD52 49  FF     EOR  #$FF
5434 FD54 B5  78     STA  DELTAC+1
5435 FD56 E6  77     INC  DELTAC  ;ADD ONE FOR TWO'S COMPLEMENT
5436 FD58 D0  02     BNE  DRAW2
ERR LINE ADDR B1 B2 B3 B4 DISPLAY HANDLER -- 10-30-78 -- DISPLC

5437 FD5A E6 7B INC DELTAC+1
5438 FD5C A2 02 DRAW2: LDX #2 ; ZERO RAM FOR DRAW LOOP
5439 FD5E A0 00 LDY #0
5440 FD60 B4 73 STY COLAC+1
5441 FD62 9B DRAW3A: TYA
5442 FD63 95 70 STA ROWAC,X
5443 FD65 B5 5A LDA OLDROW,X
5444 FD67 95 54 STA ROWCRS,X
5445 FD69 CA DEX
5446 FD6A 10 F6 BPL DRAW3A
5447 FD6C A5 77 LDA DELTAC ; FIND LARGER ONE (ROW OR COL)
5448 FD6E E8 INX ; PREPARE COUNTR AND ENDPT
5449 FD70 A5 7B STA COUNTR ; MAKE X 0
5450 FD72 85 7F LDA DELTAC+1
5451 FD74 85 75 STA ENDPT+1
5452 FD76 D0 08 BNE DRAW3 ; AUTOMATICALLY LARGER IF MSD>0
5453 FD78 A5 77 LDA DELTAC
5454 FD7A C5 76 CMP DELTAR i LOW COL > LOW ROW? i YES
5455 FD7C B0 05 BCS DRAW3
5456 FD7E A5 76 LDA DELTAR
5457 FD80 A2 02 LDX #2
5458 FD82 A8 DRAW3: TYA ; PUT IN INITIAL CONDITIONS
5459 FD84 90 02 BNE DRAW4A ; IF COUNTER IS ZERO, LEAVE DRAW
5460 FD86 90 03 BNE DRAW5A ; IF COUNTER IS ZERO, LEAVE DRAW
5461 FD88 4A PHA ; SAVE AC
5462 FD8A 02 00 STA ROWAC,X ; PUT LSB OF HIGH BYTE
5463 FD8C 48 PLA ; INTO CARRY
5464 FD8E 65 76 PLA ; RESTORE AC
5465 FD90 95 70 ROR A ; ROR THE 9 BIT ACUMULATOR
5466 FD92 A5 7E DRAW4A: LDA COUNTR ; TEST ZERO
5467 FD94 05 7F ORA COUNTR+1
5468 FD96 D0 03 LSR A ; IF COUNTER IS ZERO, LEAVE DRAW
5469 FD98 6A BNE DRAW11 ; IF COUNTER IS ZERO, LEAVE DRAW
5470 FD9A 9C 42 FE ADC ROWINC ; ADD ROW TO ROWAC (PLOT LOOP)
5471 FD9C 65 76 ADC DELTAR
5472 FD9E 85 70 STA ROWAC
5473 FDA0 90 02 BCC DRAW5
5474 FDA2 E6 71 INC ROWAC+1
5475 FDA4 A5 71 DRAW5: LDA ROWAC+1 ; COMPARE ROW TO ENDPOINT
5476 FDA6 C5 75 CMP ENDPT+1 ; IF HIGH BYTE OF ROW IS > .LT. HIGH
5477 FDAA D0 06 BNE DRAW5A ; BYTE OF ENDPT, BLT TO COLUMN
5478 FDAC A5 70 LDA ROWAC
5479 FDAE C5 74 CMP ENDPT ; LOW BYTE
5480 FDB0 90 0C BCC DRAW5 ; ALSO BLT
5481 FDB2 18 DRAW5A: CLC ; MOVE POINT
5482 FDB4 A5 54 LDA ROWCRS
5483 FDB6 65 79 ADC ROWINC
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5491</td>
<td>FDB7</td>
<td>85 54</td>
<td>STA ROWCRS</td>
</tr>
<tr>
<td>5492</td>
<td>FDB9</td>
<td>A2 00</td>
<td>LDX #0</td>
</tr>
<tr>
<td>5493</td>
<td>FDBB</td>
<td>20 7A FA</td>
<td>JSR SUBEND</td>
</tr>
<tr>
<td>5494</td>
<td>FDBE</td>
<td>18</td>
<td>DRAW6: CLC</td>
</tr>
<tr>
<td>5495</td>
<td>FDBF</td>
<td>A5 72</td>
<td>LDA COLAC</td>
</tr>
<tr>
<td>5496</td>
<td>FDC1</td>
<td>65 77</td>
<td>ADC DELTAC</td>
</tr>
<tr>
<td>5497</td>
<td>FDC3</td>
<td>85 72</td>
<td>STA COLAC</td>
</tr>
<tr>
<td>5498</td>
<td>FDC5</td>
<td>A5 73</td>
<td>LDA COLAC+1</td>
</tr>
<tr>
<td>5499</td>
<td>FDC7</td>
<td>65 78</td>
<td>ADC DELTAC+1</td>
</tr>
<tr>
<td>5500</td>
<td>FDC9</td>
<td>B5 73</td>
<td>STA COLAC+1</td>
</tr>
<tr>
<td>5501</td>
<td>FDCB</td>
<td>C5 75</td>
<td>CMP ENDPT+1</td>
</tr>
<tr>
<td>5502</td>
<td>FDCD</td>
<td>90 27</td>
<td>BCC DRAW8</td>
</tr>
<tr>
<td>5503</td>
<td>FDCF</td>
<td>D0 06</td>
<td>BNE DRAW6A</td>
</tr>
<tr>
<td>5504</td>
<td>FDD1</td>
<td>A5 72</td>
<td>LDA COLAC</td>
</tr>
<tr>
<td>5505</td>
<td>FDD3</td>
<td>C3 74</td>
<td>CMP ENDPT</td>
</tr>
<tr>
<td>5506</td>
<td>FDD5</td>
<td>90 1F</td>
<td>BCC DRAW8</td>
</tr>
<tr>
<td>5507</td>
<td>FDD7</td>
<td>24 7A</td>
<td>DRAW6A: BIT COLINC</td>
</tr>
<tr>
<td>5508</td>
<td>FDD9</td>
<td>10 10</td>
<td>BPL DRAW6B</td>
</tr>
<tr>
<td>5509</td>
<td>FDBB</td>
<td>C6 55</td>
<td>DEC COLCRS</td>
</tr>
<tr>
<td>5510</td>
<td>FDDD</td>
<td>A5 55</td>
<td>LDA COLCRS</td>
</tr>
<tr>
<td>5511</td>
<td>FDDF</td>
<td>C9 FF</td>
<td>CMP #$FF</td>
</tr>
<tr>
<td>5512</td>
<td>FDE1</td>
<td>D0 0E</td>
<td>BNE DRAW7</td>
</tr>
<tr>
<td>5513</td>
<td>FDE3</td>
<td>A5 56</td>
<td>LDA COLCRS+1</td>
</tr>
<tr>
<td>5514</td>
<td>FDE5</td>
<td>F0 0A</td>
<td>BEG DRAW7</td>
</tr>
<tr>
<td>5515</td>
<td>FDE7</td>
<td>C6 56</td>
<td>DEC COLCRS+1</td>
</tr>
<tr>
<td>5516</td>
<td>FDE9</td>
<td>10 06</td>
<td>BPL DRAW7</td>
</tr>
<tr>
<td>5517</td>
<td>FDEB</td>
<td>E6 55</td>
<td>DRAW6B: INC COLCRS</td>
</tr>
<tr>
<td>5518</td>
<td>FDED</td>
<td>D0 02</td>
<td>BNE DRAW7</td>
</tr>
<tr>
<td>5519</td>
<td>FDEF</td>
<td>E6 56</td>
<td>INC COLCRS+1</td>
</tr>
<tr>
<td>5520</td>
<td>FDF1</td>
<td>A2 02</td>
<td>DRAW7: LDX #2</td>
</tr>
<tr>
<td>5521</td>
<td>FDF3</td>
<td>20 7A FA</td>
<td>JSR SUBEND</td>
</tr>
<tr>
<td>5522</td>
<td>FDF6</td>
<td>20 96 FA</td>
<td>JSR RANGE</td>
</tr>
<tr>
<td>5523</td>
<td>FDF9</td>
<td>20 E0 F5</td>
<td>JSR OUTPLT</td>
</tr>
<tr>
<td>5524</td>
<td>FDFC</td>
<td>AD B7 02</td>
<td>LDA FILFLG</td>
</tr>
<tr>
<td>5525</td>
<td>FDFF</td>
<td>F0 2F</td>
<td>BEG DRAW9</td>
</tr>
<tr>
<td>5526</td>
<td>FE01</td>
<td>20 9D FC</td>
<td>JSR PHACSRS</td>
</tr>
<tr>
<td>5527</td>
<td>FE04</td>
<td>AD FB 02</td>
<td>LDA ATACHR</td>
</tr>
<tr>
<td>5528</td>
<td>FE07</td>
<td>BD BC 02</td>
<td>STA HOLD4</td>
</tr>
<tr>
<td>5529</td>
<td>FE0A</td>
<td>A5 54</td>
<td>DRAW8A: LDA ROWCRS</td>
</tr>
<tr>
<td>5530</td>
<td>FE0C</td>
<td>4B</td>
<td>PHA</td>
</tr>
<tr>
<td>5531</td>
<td>FE0D</td>
<td>20 DC F9</td>
<td>JSR INCRTA</td>
</tr>
<tr>
<td>5532</td>
<td>FE10</td>
<td>68</td>
<td>PLA</td>
</tr>
<tr>
<td>5533</td>
<td>FE11</td>
<td>B5 54</td>
<td>STA ROWCRS</td>
</tr>
<tr>
<td>5534</td>
<td>FE13</td>
<td>20 96 FA</td>
<td>DRAW8C: JSR RANGE</td>
</tr>
<tr>
<td>5535</td>
<td>FE16</td>
<td>20 A2 F5</td>
<td>JSR GETPLT</td>
</tr>
<tr>
<td>5536</td>
<td>FE19</td>
<td>D0 0C</td>
<td>BNE DRAW8B</td>
</tr>
<tr>
<td>5537</td>
<td>FE1B</td>
<td>AD FD 02</td>
<td>LDA FILDAT</td>
</tr>
<tr>
<td>5538</td>
<td>FE1E</td>
<td>BD FB 02</td>
<td>STA ATACHR</td>
</tr>
<tr>
<td>5539</td>
<td>FE21</td>
<td>20 E0 F5</td>
<td>JSR OUTPLT</td>
</tr>
<tr>
<td>5540</td>
<td>FE24</td>
<td>4C 0A FE</td>
<td>JMP DRAW8A</td>
</tr>
<tr>
<td>5541</td>
<td>FE27</td>
<td>AD BC 02</td>
<td>DRAW8B: LDA HOLD4</td>
</tr>
<tr>
<td>5542</td>
<td>FE2A</td>
<td>BD FB 02</td>
<td>STA ATACHR</td>
</tr>
<tr>
<td>5543</td>
<td>FE2D</td>
<td>20 AB FC</td>
<td>JSR PLACRS</td>
</tr>
<tr>
<td>5544</td>
<td>FE30</td>
<td>38</td>
<td>DRAW9: SEC</td>
</tr>
</tbody>
</table>

<p>| 5545     | FDDC | D0 06      | BNE DRAW6A                          |
| 5546     | FDF9 | 20 E0 F5   | JSR OUTPLT                          |
| 5547     | FDFC | AD B7 02   | LDA FILFLG                          |
| 5548     | FDFF | F0 2F      | BEG DRAW9                           |
| 5549     | FE01 | 20 9D FC   | JSR PHACSRS                         |
| 5550     | FE04 | AD FB 02   | LDA ATACHR                         |
| 5551     | FE07 | BD BC 02   | STA HOLD4                          |
| 5552     | FE0A | A5 54      | DRAW8A: LDA ROWCRS                  |
| 5553     | FE0C | 4B         | PHA                                  |
| 5554     | FE0D | 20 DC F9   | JSR INCRTA                          |
| 5555     | FE10 | 68         | PLA                                  |
| 5556     | FE11 | B5 54      | STA ROWCRS                          |
| 5557     | FE13 | 20 96 FA   | DRAW8C: JSR RANGE                   |
| 5558     | FE16 | 20 A2 F5   | JSR GETPLT                          |
| 5559     | FE19 | D0 0C      | BNE DRAW8B                          |
| 5560     | FE1B | AD FD 02   | LDA FILDAT                          |
| 5561     | FE1E | BD FB 02   | STA ATACHR                          |
| 5562     | FE21 | 20 E0 F5   | JSR OUTPLT                          |
| 5563     | FE24 | 4C 0A FE  | JMP DRAW8A                         |
| 5564     | FE27 | AD BC 02   | DRAW8B: LDA HOLD4                   |
| 5565     | FE2A | BD FB 02   | STA ATACHR                          |
| 5566     | FE2D | 20 AB FC   | JSR PLACRS                          |
| 5567     | FE30 | 38         | DRAW9: SEC                          |</p>
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5545</td>
<td>FE31</td>
<td>A5</td>
<td>7E</td>
<td></td>
<td></td>
<td>LDA COUNTR</td>
</tr>
<tr>
<td>5546</td>
<td>FE33</td>
<td>E9</td>
<td>01</td>
<td></td>
<td></td>
<td>SBC #1</td>
</tr>
<tr>
<td>5547</td>
<td>FE35</td>
<td>85</td>
<td>7E</td>
<td></td>
<td></td>
<td>STA COUNTR</td>
</tr>
<tr>
<td>5548</td>
<td>FE37</td>
<td>A5</td>
<td>7F</td>
<td></td>
<td></td>
<td>LDA COUNTR+1</td>
</tr>
<tr>
<td>5549</td>
<td>FE39</td>
<td>E9</td>
<td>00</td>
<td></td>
<td></td>
<td>SBC #0</td>
</tr>
<tr>
<td>5550</td>
<td>FE3B</td>
<td>85</td>
<td>7F</td>
<td></td>
<td></td>
<td>STA COUNTR+1</td>
</tr>
<tr>
<td>5551</td>
<td>FE3D</td>
<td>30</td>
<td>03</td>
<td></td>
<td></td>
<td>BMI DRAW10</td>
</tr>
<tr>
<td>5552</td>
<td>FE3F</td>
<td>4C</td>
<td>90</td>
<td>FD</td>
<td></td>
<td>JMP DRAW4A</td>
</tr>
<tr>
<td>5553</td>
<td>FE42</td>
<td>4C</td>
<td>34</td>
<td>F6</td>
<td></td>
<td>DRAW10: JMP RETUR1</td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4

5554
5555
5556
5557
5558
5559
5560
5561
5562 FE45 18 10 0A 0A
5563 FE49 10 1C 34 64
5564 FE4D C4 C4 C4 C4
5565
5566
5567
5568
5569 FE51 17 17 0B 17
5570 FE55 2F 2F 5F 5F
5571 FE59 61 61 61 61
5572 FE5D 13 13 09 13
5573 FE61 27 27 4F 4F
5574 FE65 41 41 41 41
5575
5576
5577
5578
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593 FE69 02 06 07 08
5594 FE6D 09 0A 0B 0D
5595 FE71 0F 0F 0F 0F
5596
5597
5598
5599
5600
5601 FE75 00 00 00 00
5602 FE79 00 00 00 01
5603 FE7D 01 01 01 01
5604
5605
5606
5607

DISPLAY HANDLER -- 10-30-78 -- DISPLC PAGE 124

\section{PAGE}

\subsection{TABLES}

\section{MEMORY ALLOCATION}

\section{NUMBER OF DISPLAY LIST ENTRIES}

\section{ANTIC CODE FROM INTERNAL MODE CONVERSION TABLE}

\section{INTERNAL ANTAGN CODE DESCRIPTION}

\begin{itemize}
\item 0 2 \text{40X2X8 CHARACTERS}
\item 1 6 \text{20X5X8 ''}
\item 2 7 \text{20X5X16 ''}
\item 3 8 \text{40X4X8 GRAPHICS}
\item 4 9 \text{80X2X4 ''}
\item 5 \text{80X4X4 ''}
\item 6 \text{160X2X2 ''}
\item 7 D \text{160X4X2 ''}
\item 8 F \text{320X2X1 ''}
\item 9 \text{SAME AS 8 BUT GTIA 'LUM' MODE}
\item 10 \text{SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE}
\item 11 \text{SAME AS 8 BUT GTIA 'COLOR' MODE}
\end{itemize}

\section{ANCONV}

\section{PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF CROSSING A 256 BYTE PAGE BOUNDARY}

\section{PAGETB}

\section{THIS IS THE NUMBER OF LEFT SHIFTS NEEDED TO MULTIPLY COLCRS BY 10,20, OR 40. (ROWCRS*10)//(2**DHLINE)
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>DISPLAY HANDLER -- 10-30-7B -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5608</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5609</td>
<td>FE81</td>
<td>02</td>
<td>01</td>
<td>01</td>
<td>00</td>
<td>DHLINE: .BYTE 2,1,1,0,0,1,1,2,2,2,2</td>
</tr>
<tr>
<td>5610</td>
<td>FE85</td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>5611</td>
<td>FE89</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>5612</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5613</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5614</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5615</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5616</td>
<td>FE8D</td>
<td>28</td>
<td>14</td>
<td>14</td>
<td>28</td>
<td>COLUMN: .BYTE 40,20,20,40,80,80,160,160,64,80,80,80 ; MODE B IS SPECIAL</td>
</tr>
<tr>
<td>5617</td>
<td>FE91</td>
<td>50</td>
<td>50</td>
<td>A0</td>
<td>A0</td>
<td></td>
</tr>
<tr>
<td>5618</td>
<td>FE95</td>
<td>40</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>5619</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5620</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5621</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5622</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5623</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5624</td>
<td>FE99</td>
<td>18</td>
<td>18</td>
<td>OC</td>
<td>18</td>
<td>NOROWS: .BYTE 24,24,12,24,48,48,96,96,192,192,192,192</td>
</tr>
<tr>
<td>5625</td>
<td>FE9D</td>
<td>30</td>
<td>30</td>
<td>60</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>5626</td>
<td>FEA1</td>
<td>C0</td>
<td>C0</td>
<td>C0</td>
<td>C0</td>
<td></td>
</tr>
<tr>
<td>5627</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5628</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5629</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5630</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5631</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5632</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5633</td>
<td>FEA5</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>02</td>
<td>DIV2TB: .BYTE 0,0,0,2,3,2,3,2,3,1,1,1</td>
</tr>
<tr>
<td>5634</td>
<td>FEA9</td>
<td>03</td>
<td>02</td>
<td>03</td>
<td>02</td>
<td></td>
</tr>
<tr>
<td>5635</td>
<td>FEAD</td>
<td>03</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>5636</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5637</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5638</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5639</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5640</td>
<td>FEB1</td>
<td>00</td>
<td>FF</td>
<td>FO</td>
<td>0F</td>
<td>DMASKT: .BYTE $00,$FF,$FO,$OF</td>
</tr>
<tr>
<td>5641</td>
<td>FEB5</td>
<td>C0</td>
<td>30</td>
<td>OC</td>
<td>03</td>
<td>.BYTE $00,$30,$0C,$03</td>
</tr>
<tr>
<td>5642</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5643</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5644</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5645</td>
<td>FEB9</td>
<td>80</td>
<td>40</td>
<td>20</td>
<td>10</td>
<td>MASKTB: .BYTE $80,$40,$20,$10,$08,$04,$02,$01</td>
</tr>
<tr>
<td>5646</td>
<td>FE8D</td>
<td>08</td>
<td>04</td>
<td>02</td>
<td>01</td>
<td></td>
</tr>
<tr>
<td>5647</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5648</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5649</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5650</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5651</td>
<td>FEC1</td>
<td>28</td>
<td>CA</td>
<td>94</td>
<td>46</td>
<td>COLRTB: .BYTE $28,$CA,$94,$46,$00</td>
</tr>
<tr>
<td>5652</td>
<td>FEC5</td>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5653</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5654</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5655</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5656</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5657</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5658</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5659</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5660</td>
<td>FEC6</td>
<td>1B</td>
<td></td>
<td></td>
<td></td>
<td>CNTRLS: .BYTE $1B</td>
</tr>
<tr>
<td>5661</td>
<td>FEC7</td>
<td>79</td>
<td>F7</td>
<td></td>
<td></td>
<td>.WORD ESCAPE</td>
</tr>
<tr>
<td>ERR LINE</td>
<td>ADDR</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>5662</td>
<td>FEC9</td>
<td>1C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5663</td>
<td>FECA</td>
<td>7F</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5664</td>
<td>FECC</td>
<td>1D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5665</td>
<td>FECD</td>
<td>8C</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5666</td>
<td>FECF</td>
<td>1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5667</td>
<td>FED0</td>
<td>99</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5668</td>
<td>FED2</td>
<td>1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5669</td>
<td>FED3</td>
<td>AA</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5670</td>
<td>FED5</td>
<td>7D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5671</td>
<td>FED6</td>
<td>89</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5672</td>
<td>FED8</td>
<td>7E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5673</td>
<td>FED9</td>
<td>E6</td>
<td>F7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5674</td>
<td>FEDB</td>
<td>7F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5675</td>
<td>FEDC</td>
<td>10</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5676</td>
<td>FEDE</td>
<td>9B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5677</td>
<td>FEDF</td>
<td>30</td>
<td>FA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5678</td>
<td>FEE1</td>
<td>9C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5679</td>
<td>FEE2</td>
<td>D4</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5680</td>
<td>FEE4</td>
<td>9D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5681</td>
<td>FEE5</td>
<td>A4</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5682</td>
<td>FEE7</td>
<td>9E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5683</td>
<td>FEE8</td>
<td>32</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5684</td>
<td>FEEA</td>
<td>9F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5685</td>
<td>FEEB</td>
<td>2D</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5686</td>
<td>FEED</td>
<td>FD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5687</td>
<td>FEEF</td>
<td>0A</td>
<td>F9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5688</td>
<td>FEFO</td>
<td>FE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5689</td>
<td>FEF1</td>
<td>AD</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5690</td>
<td>FEF3</td>
<td>FF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5691</td>
<td>FEF4</td>
<td>37</td>
<td>F8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DISPLAY HANDLER -- 10-30-78 -- DISPLC**

**PAGE 126**

- **BYTE** $1C
- **WORD** CRSRUP
- **BYTE** $1D
- **WORD** CRSRDN
- **BYTE** $1E
- **WORD** CRSRLF
- **BYTE** $1F
- **WORD** CRSRRT
- **BYTE** $7D
- **WORD** CLRSR
- **BYTE** $7E
- **WORD** BS
- **BYTE** $7F
- **WORD** TAB
- **BYTE** $98
- **WORD** DOCRWS
- **BYTE** $9C
- **WORD** DELLIN
- **BYTE** $9D
- **WORD** INSLIN
- **BYTE** $9E
- **WORD** CLRtab
- **BYTE** $9F
- **WORD** SETTAB
- **BYTE** $FD
- **WORD** BELL
- **BYTE** $FE
- **WORD** DELCHR
- **BYTE** $FF
- **WORD** INSCHR

- **BYTE** $40, $00, $20, $60
- **BYTE** $40, $00, $60
- **BYTE** $20, $40, $00, $60
- **BYTE** $6C, $6A, $3B, $80, $80, $68, $2B, $2A
- **BYTE** $6C, $6A, $3B, $80, $5B, $69, $2D, $3D
- **BYTE** $76, $80, $63, $80, $80, $62, $78, $7A
<table>
<thead>
<tr>
<th>ERR LINE</th>
<th>ADDR</th>
<th>B1 B2 B3 B4</th>
<th>DISPLAY HANDLER -- 10-30-78 -- DISPLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>5716</td>
<td>FF16</td>
<td>34 80 33 36</td>
<td>.BYTE $34, $80, $33, $36, $1B, $35, $32, $31</td>
</tr>
<tr>
<td>5717</td>
<td>FF1A</td>
<td>1B 35 32 31</td>
<td></td>
</tr>
<tr>
<td>5718</td>
<td>FF1E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5719</td>
<td>FF1E</td>
<td>2C 20 2E 6E</td>
<td>.BYTE $2C, $20, $2E, $6E, $80, $6D, $2F, $81</td>
</tr>
<tr>
<td>5720</td>
<td>FF22</td>
<td>80 6D 2F 81</td>
<td></td>
</tr>
<tr>
<td>5721</td>
<td>FF26</td>
<td>72 80 65 79</td>
<td>.BYTE $72, $80, $65, $79, $7F, $74, $77, $71</td>
</tr>
<tr>
<td>5722</td>
<td>FF2A</td>
<td>7F 74 77 71</td>
<td></td>
</tr>
<tr>
<td>5723</td>
<td>FF2E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5724</td>
<td>FF2E</td>
<td>39 80 30 37</td>
<td>.BYTE $39, $80, $30, $37, $7E, $3B, $3C, $3E</td>
</tr>
<tr>
<td>5725</td>
<td>FF32</td>
<td>7E 38 3C 3E</td>
<td></td>
</tr>
<tr>
<td>5726</td>
<td>FF36</td>
<td>66 68 64 80</td>
<td>.BYTE $66, $68, $64, $80, $82, $67, $73, $61</td>
</tr>
<tr>
<td>5727</td>
<td>FF3A</td>
<td>82 67 73 61</td>
<td></td>
</tr>
<tr>
<td>5728</td>
<td>FF3E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5729</td>
<td>FF3E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5730</td>
<td>FF3E</td>
<td>4C 4A 3A 80</td>
<td>.BYTE $4C, $4A, $3A, $80, $80, $4B, $5C, $5E : UPPER CASE</td>
</tr>
<tr>
<td>5731</td>
<td>FF42</td>
<td>80 4B 5C 5E</td>
<td></td>
</tr>
<tr>
<td>5732</td>
<td>FF46</td>
<td>4F 80 50 55</td>
<td>.BYTE $4F, $80, $50, $55, $9B, $49, $5F, $7C</td>
</tr>
<tr>
<td>5733</td>
<td>FF4A</td>
<td>9B 49 5F 7C</td>
<td></td>
</tr>
<tr>
<td>5734</td>
<td>FF4E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5735</td>
<td>FF4E</td>
<td>56 80 43 80</td>
<td>.BYTE $56, $80, $43, $80, $80, $42, $58, $5A</td>
</tr>
<tr>
<td>5736</td>
<td>FF52</td>
<td>80 42 5B 5A</td>
<td></td>
</tr>
<tr>
<td>5737</td>
<td>FF56</td>
<td>24 80 23 26</td>
<td>.BYTE $24, $80, $23, $26, $1B, $25, $22, $21</td>
</tr>
<tr>
<td>5738</td>
<td>FF5A</td>
<td>1B 25 22 21</td>
<td></td>
</tr>
<tr>
<td>5739</td>
<td>FF5E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5740</td>
<td>FF5E</td>
<td>5B 20 5D 4E</td>
<td>.BYTE $5B, $20, $5D, $4E, $80, $4D, $3F, $81</td>
</tr>
<tr>
<td>5741</td>
<td>FF62</td>
<td>80 4D 3F 81</td>
<td></td>
</tr>
<tr>
<td>5742</td>
<td>FF66</td>
<td>92 80 45 59</td>
<td>.BYTE $52, $80, $45, $59, $9F, $54, $57, $51</td>
</tr>
<tr>
<td>5743</td>
<td>FF6A</td>
<td>9F 54 57 91</td>
<td></td>
</tr>
<tr>
<td>5744</td>
<td>FF6E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5745</td>
<td>FF6E</td>
<td>2B 80 29 27</td>
<td>.BYTE $2B, $80, $29, $27, $9C, $40, $7D, $9D</td>
</tr>
<tr>
<td>5746</td>
<td>FF72</td>
<td>9C 40 7D 9D</td>
<td></td>
</tr>
<tr>
<td>5747</td>
<td>FF76</td>
<td>46 48 44 80</td>
<td>.BYTE $46, $48, $44, $80, $83, $47, $53, $41</td>
</tr>
<tr>
<td>5748</td>
<td>FF7A</td>
<td>83 47 53 41</td>
<td></td>
</tr>
<tr>
<td>5749</td>
<td>FF7E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5750</td>
<td>FF7E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5751</td>
<td>FF7E</td>
<td>0C 0A 7B 80</td>
<td>.BYTE $0C, $0A, $7B, $80, $80, $0B, $02, $1E, $1F : CONTROL</td>
</tr>
<tr>
<td>5752</td>
<td>FF82</td>
<td>80 08 1E 1F</td>
<td></td>
</tr>
<tr>
<td>5753</td>
<td>FF86</td>
<td>0F 80 10 15</td>
<td>.BYTE $0F, $80, $10, $15, $98, $09, $1C, $1D</td>
</tr>
<tr>
<td>5754</td>
<td>FF8A</td>
<td>9B 09 1C 1D</td>
<td></td>
</tr>
<tr>
<td>5755</td>
<td>FF8E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5756</td>
<td>FF8E</td>
<td>16 80 03 80</td>
<td>.BYTE $16, $80, $03, $80, $80, $02, $1B, $1A</td>
</tr>
<tr>
<td>5757</td>
<td>FF92</td>
<td>80 02 1B 1A</td>
<td></td>
</tr>
<tr>
<td>5758</td>
<td>FF96</td>
<td>80 80 85 80</td>
<td>.BYTE $80, $80, $85, $80, $1B, $80, $FD, $80</td>
</tr>
<tr>
<td>5759</td>
<td>FF9A</td>
<td>1B 80 FD 80</td>
<td></td>
</tr>
<tr>
<td>5760</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5761</td>
<td>FF9E</td>
<td>00 20 60 0E</td>
<td>.BYTE $00, $20, $60, $0E, $80, $0D, $80, $81</td>
</tr>
<tr>
<td>5762</td>
<td>FF9E</td>
<td>80 0D 80 81</td>
<td></td>
</tr>
<tr>
<td>5763</td>
<td>FF9E</td>
<td>12 80 05 19</td>
<td>.BYTE $12, $80, $05, $19, $9E, $14, $17, $11</td>
</tr>
<tr>
<td>5764</td>
<td>FF9E</td>
<td>9E 14 17 11</td>
<td></td>
</tr>
<tr>
<td>5765</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5766</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5767</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5768</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5769</td>
<td>FF9E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ERR LINE ADDR B1 B2 B3 B4  DISPLAY HANDLER -- 10-30-78 --  DISPLC  PAGE 128

5770
5771
5772
5773
5774
5775 FFBE AD 09 D2  PIRG5: LDA KBCODE
5776 FFC1 CD F2 02  CMP CH1 ; TEST AGAINST LAST KEY PRESSED
5777 FFC4 D0 05  BNE PIRG3 ; IF NOT, GO PROCESS KEY
5778 FFC6 AD F1 02  LDA KEYDEL ; IF KEY DELAY BYTE > 0
5779 FFC9 D0 20  BNE PIRG4 ; IGNORE KEY AS BOUNCE
5780 FFCB AD 09 D2  PIRG3: LDA KBCODE ; RESTORE AC
5781 FFCE C9 9F  CMP #CNTL1 ; TEST CONTROL 1 (SSFLAG)
5782 FFD0 D0 0A  BNE PIRG1
5783 FFD2 AD FF 02  LDA SSFLAG0
5784 FFD5 49 FF  EOR #$FF
5785 FFD7 8D FF 02  STA SSFLAG0
5786 FFDA BD 0F  BCS PIRG4 ; (UNCONDITIONAL) MAKE ^1 INVISIBLE
5787 FFDC BD FC 02  PIRG1: STA CH
5788 FFD8 BD F2 02  STA CH1
5789 FFE2 A9 03  LDA #$3
5790 FFE4 BD F1 02  STA KEYDEL ; INITIALIZE KEY DELAY FOR DEBOUNCE
5791 FFE7 A9 00  LDA #$0 ; CLEAR COLOR SHIFT BYTE
5792 FFE9 B5 4D  STA ATRACT
5793 FFEB A9 30  PIRG4: LDA #$30
5794 FFED 8D 28 02  STA SRTIMR
5795 FFF0 6B  PIRG2: PLA
5796 FFF1 40  RTI
5797
5798
5799 FFF2 FF FF FFFF  .BYTE $FF,$FF,$FF,$FF
5800 FFF6 FF FF
5801
5802 FFF8 CRNTPC ==
5803 **==14
5804 0014 00  KBDSPR: .BYTE $FF8-CRNTPC ; ^DISPLC IS TOO LONG
5805 0015 .END

ASSEMBLY ERRORS = 0
### CROSS REFERENCE

<table>
<thead>
<tr>
<th>LABEL</th>
<th>VALUE</th>
<th>REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>0041</td>
<td>-1632 1902</td>
</tr>
<tr>
<td>ACKREC</td>
<td>E9C6</td>
<td>1785 -1794</td>
</tr>
<tr>
<td>ADDC0R</td>
<td>030E</td>
<td>-498 2515 2519 2523</td>
</tr>
<tr>
<td>ADJ1</td>
<td>ED0C</td>
<td>2533 -2537</td>
</tr>
<tr>
<td>ADJUST</td>
<td>ED04</td>
<td>2470 2473 -2532</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>0064</td>
<td>-269 4120 4145 4171 4173 4182 4183 4184</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4186 4252 4254 4258 4260 4263 4265 4296</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4335 4357 4426 4443 4449 4570 4573 4574</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4653 4655 4681 4686 4748 4755 4758 4760</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4762 4766 4834 4835 4838 4839 5092 5094</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5101 5104 5106 5149 5161 5163 5165 5168</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5171 5175 5176 5178 5179 5287 5378 5380</td>
</tr>
<tr>
<td>ADRTAB</td>
<td>E6FE</td>
<td>-1292 1327</td>
</tr>
<tr>
<td>AFP</td>
<td>D800</td>
<td>-574</td>
</tr>
<tr>
<td>ALLPOT</td>
<td>D20B</td>
<td>-659</td>
</tr>
<tr>
<td>ALLSEC</td>
<td>F30E</td>
<td>3886 -3891</td>
</tr>
<tr>
<td>ALOCAT</td>
<td>FE45</td>
<td>4146 -5562</td>
</tr>
<tr>
<td>ANCONV</td>
<td>FE69</td>
<td>4139 -5593</td>
</tr>
<tr>
<td>ANTIC</td>
<td>D400</td>
<td>-735 736 737 738 739 740 741 742</td>
</tr>
<tr>
<td></td>
<td></td>
<td>743 744 745 746 747 748 749 750</td>
</tr>
<tr>
<td>APPEND</td>
<td>0001</td>
<td>-111</td>
</tr>
<tr>
<td>APPMHI</td>
<td>000E</td>
<td>-179 4761 4765</td>
</tr>
<tr>
<td>ASCCC01</td>
<td>F705</td>
<td>4478 -4480</td>
</tr>
<tr>
<td>ASCZER</td>
<td>0030</td>
<td>-764 1193</td>
</tr>
<tr>
<td>ATACHR</td>
<td>02FB</td>
<td>-469 4308 4311 4316 4333 4372 4394 4417</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4421 4428 4481 4515 4526 4528 4606 5082</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5300 5527 5538 5542</td>
</tr>
<tr>
<td>ATINT</td>
<td>FEF6</td>
<td>4343 -5699</td>
</tr>
<tr>
<td>ATAN</td>
<td>BE43</td>
<td>-602</td>
</tr>
<tr>
<td>ATASCI</td>
<td>FEFE</td>
<td>4480 -5709</td>
</tr>
<tr>
<td>ATE0F</td>
<td>F00B</td>
<td>3250 -3256</td>
</tr>
<tr>
<td>ATRACT</td>
<td>004D</td>
<td>-248 1340 1395 1401 1403 5792</td>
</tr>
<tr>
<td>AUDC01</td>
<td>D201</td>
<td>-667 2343 2353 2378</td>
</tr>
<tr>
<td>AUDC02</td>
<td>D203</td>
<td>-669 2354</td>
</tr>
<tr>
<td>AUDC03</td>
<td>D205</td>
<td>-671 2349</td>
</tr>
<tr>
<td>AUDC04</td>
<td>D207</td>
<td>-673</td>
</tr>
<tr>
<td>AUDC05</td>
<td>D208</td>
<td>-674 2336</td>
</tr>
<tr>
<td>AUDC06</td>
<td>D209</td>
<td>-666 2297</td>
</tr>
<tr>
<td>AUDC07</td>
<td>D210</td>
<td>-668 2295</td>
</tr>
<tr>
<td>AUDC08</td>
<td>D211</td>
<td>-670 1749 2203 2607 3200</td>
</tr>
<tr>
<td>AUDC09</td>
<td>D212</td>
<td>-672 1751 2205 2609 3202</td>
</tr>
<tr>
<td>B1920HI</td>
<td>0000</td>
<td>-1641 1750</td>
</tr>
<tr>
<td>B1920LO</td>
<td>002B</td>
<td>-1640 1748</td>
</tr>
<tr>
<td>B600HI</td>
<td>0005</td>
<td>-1643 2204</td>
</tr>
<tr>
<td>B600LO</td>
<td>00CC</td>
<td>-1642 2202</td>
</tr>
<tr>
<td>BAD</td>
<td>EA63</td>
<td>1919 -1925</td>
</tr>
<tr>
<td>BADCOM</td>
<td>E9BF</td>
<td>1782 -1788 1809</td>
</tr>
<tr>
<td>BADDSK</td>
<td>F306</td>
<td>-3887 3923 3929</td>
</tr>
<tr>
<td>BADLOC</td>
<td>00B6</td>
<td>-142 826</td>
</tr>
</tbody>
</table>
BADMOD 0091 -154 4141
BADST EE9E 2914 2917 2963 -2965
BEEP F058 3164 3197 -3314
BEEP1 F05A -3315 3349
BEGIN ED10 2255 -2558 2570
BELL F90A 4402 -4726 5687
BELL1 F90C -4727 4729
BFENHI 0035 -224 1774 1886 1985 2147 2184
BFENLO 0034 -223 1771 1883 1983 2145 2179
BITCLR FB12 4624 5038 -5046
BITCON FAEB -5019 5040 5046 5056
BITGE1 FB31 5060 -5062
BITGET FB25 4618 4890 -5056
BITPUT FB04 4711 -5038 5129 5138
BITROL FAFA -5032 5159
BITSET FB06 4622 -5040
BLACK F22A -3740 3741
BLFILL EEC1 2992 -2995 3015
BLIM 02BA -398 3183 3235 3254
BLKB2 F230 3740 -3742
BLKBDV E471 -77 3483 3601 3603
BLOAD F36C 3928 -3933
BOOT F36F 3710 -3862
BOOTAD 0242 -360 3896 3898 3934 3937
BOOT? 0009 -176 3864 3931 3991 4007
BOTSCR 02BA -429 4193 4201 4542 4548 4664 4668 4953 5335
BPR 003D -234 3182 3225 3234 3238 3246 3263 3265
BRKABT 0080 -136 2637 3188 4466 4983
BRKKEY 0011 -184 1337 1957 2064 2558 2590 2643 3189
BRKKEY2 0236 -345 1292 3833 3835
BRKKEYY E754 -1336 3832 3834
BROKE EDA0 1959 2087 2666 2560 2581 -2632
BS F7E6 -4591 5673
B5 F80D 4593 -4608
BS2 FB05 4601 4603 -4605
BS3 F7F5 4596 -4598
BSA F7EC -4594
BUFADR 0015 -187 2829 2838 2842 2863 2865 3249
BUFCNT 0068 -274 4386 4310 4412 5219 5229 5232 5240
BUFFH 0004 -3421 3422 3883
BUFFL 0000 -3422 3881
BUFFUL EECB 2989 -3000
BUFFRL EE6B 227 2054 2111 2160
BUFFRH 0033 -222 1773 1885 1980 1984 2142 2146 2182
BUFFRLD 0032 -221 1769 1881 1951 1978 1982 2009 2133
BUFFSTR 006C -275 4389 4391 4406 4408 4584 4588 4927
5214 5216 5263 5265
<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKTIM</td>
<td>EAF9</td>
<td>2136</td>
<td>2138</td>
<td>2619</td>
<td></td>
</tr>
<tr>
<td>CHROG</td>
<td>E000</td>
<td>-22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CICL02</td>
<td>E53F</td>
<td>901</td>
<td>-903</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CICL03</td>
<td>E533</td>
<td>853</td>
<td>-898</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIERR1</td>
<td>E4D1</td>
<td>821</td>
<td>-826</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIERR2</td>
<td>E6B0</td>
<td>1173</td>
<td>-1183</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIERR3</td>
<td>E50F</td>
<td>-869</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIERR4</td>
<td>E511</td>
<td>842</td>
<td>-870</td>
<td>874</td>
<td>880</td>
</tr>
<tr>
<td>CIJUMP</td>
<td>E693</td>
<td>1152</td>
<td>-1158</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CINI</td>
<td>F3E1</td>
<td>3994</td>
<td>-4014</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CID</td>
<td>E4C4</td>
<td>771</td>
<td>-815</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIOCHR</td>
<td>002F</td>
<td>-217</td>
<td>815</td>
<td>951</td>
<td>956</td>
</tr>
<tr>
<td>CIO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIO1</td>
<td>E4A6</td>
<td>-787</td>
<td>798</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIOINT</td>
<td>E4A6</td>
<td>775</td>
<td>-796</td>
<td>1212</td>
<td></td>
</tr>
<tr>
<td>CIOINV</td>
<td>003E</td>
<td>-1215</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIOPEN</td>
<td>E509</td>
<td>851</td>
<td>-864</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIOPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIREAD</td>
<td>E569</td>
<td>857</td>
<td>-934</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI1</td>
<td>E62B</td>
<td>-1075</td>
<td>1080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI2</td>
<td>E61B</td>
<td>827</td>
<td>870</td>
<td>940</td>
<td>1009</td>
</tr>
<tr>
<td>CI3</td>
<td>E61D</td>
<td>893</td>
<td>909</td>
<td>930</td>
<td>952</td>
</tr>
<tr>
<td>CI4</td>
<td>00F2</td>
<td>-616</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI5</td>
<td>00F2</td>
<td>-244</td>
<td>3856</td>
<td>3997</td>
<td>4006</td>
</tr>
<tr>
<td>CI6</td>
<td>EE11</td>
<td>2808</td>
<td>-2810</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI7</td>
<td>000C</td>
<td>-93</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI8</td>
<td>F02B</td>
<td>3119</td>
<td>-3386</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI9</td>
<td>F27A</td>
<td>-3820</td>
<td>3825</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI10</td>
<td>007D</td>
<td>-4030</td>
<td>4312</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI11</td>
<td>FBA6</td>
<td>-5149</td>
<td>5151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI12</td>
<td>F99B</td>
<td>4697</td>
<td>-5144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI13</td>
<td>F140</td>
<td>-3591</td>
<td>3594</td>
<td>3598</td>
<td></td>
</tr>
<tr>
<td>CI14</td>
<td>F78F</td>
<td>-4570</td>
<td>4572</td>
<td>4576</td>
<td></td>
</tr>
<tr>
<td>CI15</td>
<td>F7C8</td>
<td>-4578</td>
<td>4581</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI16</td>
<td>F7B9</td>
<td>4280</td>
<td>4314</td>
<td>-4567</td>
<td>5671</td>
</tr>
<tr>
<td>CI17</td>
<td>F832</td>
<td>-4623</td>
<td>5683</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI18</td>
<td>F430</td>
<td>-4125</td>
<td>4127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI19</td>
<td>007D</td>
<td>-3416</td>
<td>3327</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI20</td>
<td>F03B</td>
<td>3287</td>
<td>-3293</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI21</td>
<td>03E1</td>
<td>-3071</td>
<td>3087</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI22</td>
<td>E56F</td>
<td>-1282</td>
<td>1313</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI23</td>
<td>009F</td>
<td>-4031</td>
<td>5761</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CI24</td>
<td>FEC6</td>
<td>4442</td>
<td>4444</td>
<td>5299</td>
<td>-5660</td>
</tr>
<tr>
<td>CI25</td>
<td>0072</td>
<td>-279</td>
<td>2440</td>
<td>5495</td>
<td>5497</td>
</tr>
<tr>
<td>CI26</td>
<td>D01A</td>
<td>-708</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLCR</td>
<td>FCE4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLCR1</td>
<td>FC60</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLCR2</td>
<td>FCFO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLCRS</td>
<td>0055</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLDST</td>
<td>0244</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLDSV</td>
<td>E477</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLINC</td>
<td>007A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLON</td>
<td>003A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLOR0</td>
<td>02C4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLOR1</td>
<td>02C5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLOR2</td>
<td>02C6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLOR3</td>
<td>02C7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLOR4</td>
<td>02C8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPFO</td>
<td>D016</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPF1</td>
<td>D017</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPF2</td>
<td>D018</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPF3</td>
<td>D019</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPM0</td>
<td>D012</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPM1</td>
<td>D013</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPM2</td>
<td>D014</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLPM3</td>
<td>D015</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLRSH</td>
<td>004F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLRTB</td>
<td>FE71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN</td>
<td>FEBD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COM1</td>
<td>E647</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COM2</td>
<td>E662</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMENT</td>
<td>E63D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMFRM</td>
<td>E978</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMND</td>
<td>E974</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPLT</td>
<td>0043</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPUT</td>
<td>ECA3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMRET</td>
<td>F7A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMRE1</td>
<td>F7A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMTA1</td>
<td>F7B9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMTA2</td>
<td>F7B9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONSOL</td>
<td>D01F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONTIN</td>
<td>EC31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVR1</td>
<td>F97E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVR2</td>
<td>F98B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVR3</td>
<td>F98F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVR4</td>
<td>F99C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVR5</td>
<td>F9A6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVR6</td>
<td>F9A7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVRT</td>
<td>F947</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>BD73</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT</td>
<td>ED3B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNTR</td>
<td>007E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR</td>
<td>009B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>CRETRI</td>
<td>000D</td>
<td>4510</td>
<td>-1672</td>
<td>1743</td>
<td>1802</td>
</tr>
<tr>
<td>CRETRN</td>
<td>EDBF</td>
<td>2227</td>
<td>-2259</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRETRY</td>
<td>0036</td>
<td>-225</td>
<td>1744</td>
<td>1788</td>
<td>1803</td>
</tr>
<tr>
<td>CRITIC</td>
<td>0042</td>
<td>-239</td>
<td>1412</td>
<td>1729</td>
<td>1863</td>
</tr>
<tr>
<td>CRL000</td>
<td>F5E7</td>
<td>-4329</td>
<td>4332</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP1</td>
<td>E6D5</td>
<td>-1213</td>
<td>1215</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP2</td>
<td>E90B</td>
<td>-1582</td>
<td>1584</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP3</td>
<td>EDEB</td>
<td>-2729</td>
<td>2727</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP4</td>
<td>EE7B</td>
<td>-2872</td>
<td>2875</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP5</td>
<td>EF41</td>
<td>-3093</td>
<td>3097</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP6</td>
<td>F0E3</td>
<td>-3396</td>
<td>3398</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTP7</td>
<td>F3E4</td>
<td>-4020</td>
<td>4023</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRNTPC</td>
<td>FFFB</td>
<td>-5802</td>
<td>5804</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRSINH</td>
<td>02F0</td>
<td>-459</td>
<td>1339</td>
<td>4122</td>
<td>4365</td>
</tr>
<tr>
<td>CRSPDN</td>
<td>F7BC</td>
<td>-4546</td>
<td>5665</td>
<td>4554</td>
<td>-4557</td>
</tr>
<tr>
<td>CRSRLF</td>
<td>F799</td>
<td>-4552</td>
<td>4598</td>
<td>5237</td>
<td>5244</td>
</tr>
<tr>
<td>CRSQOR</td>
<td>00BD</td>
<td>-149</td>
<td>4989</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRSRTT</td>
<td>F7AA</td>
<td>-4560</td>
<td>4609</td>
<td>5669</td>
<td></td>
</tr>
<tr>
<td>CRSRUP</td>
<td>F77F</td>
<td>-4540</td>
<td>4604</td>
<td>5663</td>
<td></td>
</tr>
<tr>
<td>CSBOOT</td>
<td>F3B2</td>
<td>3694</td>
<td>-3989</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSBDT2</td>
<td>F3C0</td>
<td>3990</td>
<td>-3997</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSIDE</td>
<td>EF2E</td>
<td>3076</td>
<td>-3079</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSIO</td>
<td>F0AC</td>
<td>-3370</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSQPIV</td>
<td>E47D</td>
<td>-81</td>
<td>3132</td>
<td>4002</td>
<td></td>
</tr>
<tr>
<td>CSTAT</td>
<td>02BB</td>
<td>-396</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTIA</td>
<td>D000</td>
<td>-681</td>
<td>682</td>
<td>683</td>
<td>684</td>
</tr>
<tr>
<td>CTIMHI</td>
<td>0000</td>
<td>-1675</td>
<td>2446</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTIMLO</td>
<td>0002</td>
<td>-1674</td>
<td>2445</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTRLC</td>
<td>0092</td>
<td>-3417</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0044</td>
<td>-2901</td>
<td>3075</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAUX1</td>
<td>030A</td>
<td>-494</td>
<td>1762</td>
<td>2960</td>
<td>3084</td>
</tr>
<tr>
<td>DAUX2</td>
<td>030B</td>
<td>-495</td>
<td>1764</td>
<td>2210</td>
<td>2237</td>
</tr>
<tr>
<td>DBDDEC</td>
<td>F913</td>
<td>4188</td>
<td>-4740</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBDEC</td>
<td>F91F</td>
<td>4178</td>
<td>-4751</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBSECT</td>
<td>0241</td>
<td>-359</td>
<td>3916</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDBSUB</td>
<td>F921</td>
<td>4148</td>
<td>4165</td>
<td>4741</td>
<td>-4752</td>
</tr>
<tr>
<td>DDBSUB1</td>
<td>F934</td>
<td>4759</td>
<td>-4761</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDBUFI</td>
<td>0305</td>
<td>-489</td>
<td>2181</td>
<td>2815</td>
<td>2864</td>
</tr>
<tr>
<td>DDBUFL</td>
<td>0304</td>
<td>-488</td>
<td>2176</td>
<td>2813</td>
<td>2862</td>
</tr>
<tr>
<td>DDBUFZ</td>
<td>0014</td>
<td>-2894</td>
<td>3077</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDBYTHI</td>
<td>0309</td>
<td>-493</td>
<td>2183</td>
<td>2820</td>
<td>2850</td>
</tr>
<tr>
<td>DDBYTL</td>
<td>0308</td>
<td>-492</td>
<td>2178</td>
<td>2818</td>
<td>2848</td>
</tr>
<tr>
<td>DDB</td>
<td>0300</td>
<td>-483</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCOMND</td>
<td>0302</td>
<td>-486</td>
<td>1759</td>
<td>2799</td>
<td>2806</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3083</td>
<td>3361</td>
<td>3376</td>
<td>3872</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCTIM1</td>
<td>E8DD 1536 -1540</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCTIMR</td>
<td>E8DD 1409 1445 1454 -1535</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCTXF</td>
<td>EBEA 1538 1541 1543 -1546</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDEVIC</td>
<td>0300 1754 2288 2350 2797 3037 3204 3371</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECBF1</td>
<td>E66D 1124 -1126</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECBFL</td>
<td>E66D 969 973 1040 1044 -1121</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECON</td>
<td>0006 -624</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELAYO</td>
<td>EC8C -2437 2441</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELAY1</td>
<td>EC8E -2438 2439</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELCH1</td>
<td>FB70 -4652 4669</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELCH2</td>
<td>FB76 4662 4665 -4670</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELCHR</td>
<td>FB6D -4651 5689</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>0021 -101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELL1</td>
<td>FBDD -4703 4714</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELL2</td>
<td>FBFB -4718</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELLIA</td>
<td>FB77 -4700</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELLIB</td>
<td>FB78 -4702 4724 5291</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELLIN</td>
<td>FBD4 -4699 5679</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTAC</td>
<td>0077 -282 5422 5429 5431 5432 5434 5435 5437 5447 5452 5456 5496 5499</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTAR</td>
<td>0076 -281 5410 5414 5418 5457 5459 5477</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELT11</td>
<td>FC82 -5287 5290</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELT12</td>
<td>FC89 -5291</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELT13</td>
<td>FC8C 5281 5288 -5292</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTIA</td>
<td>FC6B 4673 -5274</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTIB</td>
<td>FC70 5276 -5278</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELTIM</td>
<td>FC73 4597 -5279</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERR</td>
<td>E9F6 1821 -1842</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERR1</td>
<td>EA06 1791 1830 -1853</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERR5</td>
<td>F10D -3539 3543 3544</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERRH</td>
<td>00F1 -3543 3544 3949</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERRL</td>
<td>000D -3544 3948</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DERROR</td>
<td>0090 -153 1910</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV1</td>
<td>E6A6 -1175 1180</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV2</td>
<td>E6B5 1176 -1188</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV3</td>
<td>E6C5 1195 -1197</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEV4</td>
<td>E6CB 1195 -1201</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEVRC</td>
<td>E69E 873 919 -1171</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFLAGS</td>
<td>0240 -398 3893</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLINE</td>
<td>FEB1 4804 -5609</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIGRT</td>
<td>00F1 -615</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DINDEX</td>
<td>0057 -260 4106 4138 4194 4225 4363 4803 4864 4877 4905 4916 4963 5070 5368</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DINI</td>
<td>F37E 3867 3930 -3941</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DINIT</td>
<td>EDEA 2759 -2788</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIRECT</td>
<td>0002 -112</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISK</td>
<td>0044 -126</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISKID</td>
<td>0031 -2744 2796</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISKIV</td>
<td>E450 -66</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISKM</td>
<td>F3A4 3976 -3978</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPLA</td>
<td>E410 -4096</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPLY 0053</td>
<td>-122</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIVGTB FEAS</td>
<td>4810 5633</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLISHT D403</td>
<td>-739 1424</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLISTL D402</td>
<td>-738 1426</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMACTL D400</td>
<td>-736 1428</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMASK G2A0</td>
<td>-416 4297 4351 4353 4842</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMASKT FEB1</td>
<td>4841 -5640</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DNACK 006B</td>
<td>-147 1914</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOB1 FC1A</td>
<td>5222 -5224</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBDDT F2ED</td>
<td>3874 -3877 3889</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBU1A FC29</td>
<td>5225 5228 -5231</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBUF1 FC12</td>
<td>-5220 5235</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBUF2 FC39</td>
<td>5230 -5238 5250</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBUF3 FC51</td>
<td>5247 -5249</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBUF4 FC55</td>
<td>5239 5243 -5251</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOBUFC FC00</td>
<td>4405 -5211</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR FA34</td>
<td>4613 4878 4883 4886 -4901</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR1 FA00</td>
<td>4874 -4877</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR1A FA29</td>
<td>4891 4893 -4896</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR1B FA14</td>
<td>4881 -4886</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR2 FA3D</td>
<td>-4905</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR2A FA4A</td>
<td>4908 -4912</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR2B FA4D</td>
<td>4911 -4913</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR4B FA61</td>
<td>4923 -4925 4931</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOCR5S FA30</td>
<td>4319 4419 4884 -4899 5677</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DINTP ECE5</td>
<td>-2509 2512</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOLCO1 FBE5</td>
<td>-5191 5199</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOLCO2 FBF8</td>
<td>5193 -5200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOLCOL FBDD</td>
<td>4599 4608 4620 4649 4675 4699 4699 4725</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN F3F6</td>
<td>4056 -4099</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN1 F460</td>
<td>-4147 4150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN2 F4D5</td>
<td>4203 4206</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN3 F4FA</td>
<td>-4221 4224</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN4 F51C</td>
<td>-4237 4240</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN5 F524</td>
<td>4227 -4241</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN7 F589</td>
<td>4279 -4284</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPEN9 F577</td>
<td>4271 -4277</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOPENA F457</td>
<td>4140 -4143</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSDINI 000C</td>
<td>-178 3901 3903 3941 4008 4010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSSS F6AD</td>
<td>4397 -4431</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSVSEC 000A</td>
<td>-177 3602 3604 3730</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOUBLE 0044</td>
<td>-1626</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW FCFC</td>
<td>4061 -5388</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW1 FD37</td>
<td>5411 -5419</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW10 FE42</td>
<td>5474 5551 -5553</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW11 FD99</td>
<td>5473 -5475</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW2 FDB5</td>
<td>5426 5436 -5439</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW3 FDB3</td>
<td>5455 5459 -5462</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW3A FD62</td>
<td>-5441 5446</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW4A FD90</td>
<td>-5471 5552</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAW5 FDA4</td>
<td>5479 -5481</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSPE2</td>
<td>F257</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENSPEC</td>
<td>F254</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENTVEC</td>
<td>002C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOF</td>
<td>00BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOFERR</td>
<td>00BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOL</td>
<td>009B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOPEN</td>
<td>F3FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EOT</td>
<td>00FE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUTC5</td>
<td>F6BE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUTC6</td>
<td>F6B5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDUTC8</td>
<td>F6A5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERANGE</td>
<td>F9BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERETNC</td>
<td>F6BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRFLG</td>
<td>023F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>0045</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRRTN</td>
<td>E4C0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRTNH</td>
<td>00EA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRTNH</td>
<td>00EA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRTNL</td>
<td>00CC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESCAPE</td>
<td>F779</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESCFLG</td>
<td>02A2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESCN</td>
<td>00EF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESTSMM</td>
<td>F173</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP</td>
<td>D2CC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXP10</td>
<td>D2CC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEN1</td>
<td>F87E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEN2</td>
<td>F8BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEN3</td>
<td>F8BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEN4</td>
<td>F892</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTEND</td>
<td>F87B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FADD</td>
<td>DA66</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FASC</td>
<td>DB66</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCAX</td>
<td>F032</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FCHRFL</td>
<td>00F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FDIV</td>
<td>D82B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEOS</td>
<td>003F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILDAT</td>
<td>02FD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIFLGL</td>
<td>02C7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILLBF</td>
<td>EE93</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FILLIN</td>
<td>0012</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINDX</td>
<td>ECD6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOP</td>
<td>DDB9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLDOR</td>
<td>DDB9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLD1P</td>
<td>D97C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLD1R</td>
<td>D97B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOPPY</td>
<td>0030</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLPRTR</td>
<td>00FC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMOVE</td>
<td>DDB6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMSIZE</td>
<td>0043</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMTD</td>
<td>EE4A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FMUL</td>
<td>DADB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNCN1T</td>
<td>0092</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOMAT</td>
<td>0021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOPDEY</td>
<td>EADE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td>0022</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>HOLD2</td>
<td>029F</td>
<td>-415</td>
<td>4792</td>
<td>4801</td>
<td></td>
</tr>
<tr>
<td>HOLD3</td>
<td>029D</td>
<td>-413</td>
<td>4915</td>
<td>4928</td>
<td>4932</td>
</tr>
<tr>
<td>HOLD4</td>
<td>028C</td>
<td>-426</td>
<td>5528</td>
<td>5541</td>
<td></td>
</tr>
<tr>
<td>HOLD5</td>
<td>028D</td>
<td>-427</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOLDCH</td>
<td>007C</td>
<td>-286</td>
<td>4472</td>
<td>4512</td>
<td>4522</td>
</tr>
<tr>
<td>HOME</td>
<td>F7D6</td>
<td>-4582</td>
<td>4988</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOWMCH</td>
<td>F25F</td>
<td>-3798</td>
<td>3809</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM0</td>
<td>D004</td>
<td>-686</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM1</td>
<td>D005</td>
<td>-687</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM2</td>
<td>D006</td>
<td>-688</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM3</td>
<td>D007</td>
<td>-689</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM0</td>
<td>D000</td>
<td>-682</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM1</td>
<td>D001</td>
<td>-683</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM2</td>
<td>D002</td>
<td>-684</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPDSM3</td>
<td>D003</td>
<td>-688</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HSCROL</td>
<td>D404</td>
<td>-740</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICAX1</td>
<td>034A</td>
<td>-528</td>
<td>3682</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICAX1Z</td>
<td>002A</td>
<td>-213</td>
<td>935</td>
<td>1004</td>
<td>3153</td>
</tr>
<tr>
<td>ICAX2</td>
<td>034B</td>
<td>-529</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICAX2Z</td>
<td>002B</td>
<td>-214</td>
<td>3070</td>
<td>3151</td>
<td>4099</td>
</tr>
<tr>
<td>ICBH</td>
<td>0345</td>
<td>-523</td>
<td>1072</td>
<td>3680</td>
<td>3962</td>
</tr>
<tr>
<td>ICBH</td>
<td>0025</td>
<td>-208</td>
<td>1073</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICBL</td>
<td>0344</td>
<td>-522</td>
<td>1070</td>
<td>3678</td>
<td>3960</td>
</tr>
<tr>
<td>ICBL</td>
<td>0024</td>
<td>-207</td>
<td>959</td>
<td>1025</td>
<td>1071</td>
</tr>
<tr>
<td>ICBH</td>
<td>0349</td>
<td>-527</td>
<td>1143</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICBH</td>
<td>0029</td>
<td>-212</td>
<td>1145</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICBLL</td>
<td>0348</td>
<td>-526</td>
<td>1140</td>
<td>3966</td>
<td></td>
</tr>
<tr>
<td>ICBLL</td>
<td>0028</td>
<td>-211</td>
<td>947</td>
<td>948</td>
<td>1016</td>
</tr>
<tr>
<td>ICOM</td>
<td>0342</td>
<td>-520</td>
<td>3676</td>
<td>3964</td>
<td></td>
</tr>
<tr>
<td>ICOM</td>
<td>0017</td>
<td>-189</td>
<td>849</td>
<td>887</td>
<td>1106</td>
</tr>
<tr>
<td>ICOM</td>
<td>0022</td>
<td>-205</td>
<td>840</td>
<td>934</td>
<td>961</td>
</tr>
<tr>
<td>ICN</td>
<td>0341</td>
<td>-519</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICN</td>
<td>0021</td>
<td>-204</td>
<td>1197</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICHD</td>
<td>0340</td>
<td>-518</td>
<td>788</td>
<td>928</td>
<td></td>
</tr>
<tr>
<td>ICHD</td>
<td>0020</td>
<td>-203</td>
<td>864</td>
<td>904</td>
<td>914</td>
</tr>
<tr>
<td>ICIN</td>
<td>002E</td>
<td>-216</td>
<td>816</td>
<td>927</td>
<td>1069</td>
</tr>
<tr>
<td>ICPTH</td>
<td>0347</td>
<td>-525</td>
<td>792</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICPTH</td>
<td>0027</td>
<td>-210</td>
<td>892</td>
<td>906</td>
<td></td>
</tr>
<tr>
<td>ICPTL</td>
<td>0346</td>
<td>-524</td>
<td>790</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICPTL</td>
<td>0026</td>
<td>-209</td>
<td>890</td>
<td>908</td>
<td></td>
</tr>
<tr>
<td>ICSPR</td>
<td>034C</td>
<td>-530</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICSPR</td>
<td>002C</td>
<td>-215</td>
<td>216</td>
<td>217</td>
<td>810</td>
</tr>
<tr>
<td>ICSTA</td>
<td>0343</td>
<td>-521</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICSTA</td>
<td>0023</td>
<td>-206</td>
<td>899</td>
<td>995</td>
<td>1065</td>
</tr>
<tr>
<td>IDENT</td>
<td>F0F2</td>
<td>-3527</td>
<td>3535</td>
<td>3536</td>
<td>3538</td>
</tr>
<tr>
<td>IDENT</td>
<td>00F0</td>
<td>-3535</td>
<td>3536</td>
<td>3734</td>
<td></td>
</tr>
<tr>
<td>IDENT</td>
<td>00F2</td>
<td>-3536</td>
<td>3733</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFP</td>
<td>D9AA</td>
<td>-577</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IHINIT</td>
<td>E6D5</td>
<td>1230</td>
<td>-1269</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IMASK 028B -399
INATA1 FB4A 5072 -5082
INATAR CFB32 4292 -5069
INBUFF 00F3 -617
INCa A F9FB 4866 4871 -4873
INCFB1 E76 1132 -1134
INCFP E70 960 1029 -1131
INCRS1 FA77 4883 4914 4917 4919 -4934
INCRS2 F9E4 4861 -4863
INCRS3 F9F7 4868 4870 -4872 4876
INCRSA F9DC 4633 -4859 5531
INCRSB F9D4 4293 4659 -4855 5231
INCRSC F9DA 4856 -4858
INCRSR F9DB 4322 -4857
INIMLH 0007 -3410 3843
INIMLL 0000 -3409 3841
INIT EF41 3122 -3141
INSC1 F852 -4637
INSC3 F85E -4643
INSC4 F844 -4630 4642
INSC5 F86A 4645 -4648
INSC6 F861 -4644 4647
INSCR F837 -4625 5691
INSDAT 0020 -117
INSLI1 F8C6 -4693 4696
INSLI2 F8CE 4692 -4697
INSLIA F8A5 -4677 4895
INSLIN F8A4 -4676 5681
INTABS 0200 -310 1232 1292 1299 1293 1294 1295 1296 1297 1298 1299 1329 1331 1694 3631 4081 4918 5081 5704
INATA FEFA
INTEMP 022D -335 1557 1563
INTINV E46B -75 1229 3852
INTORG E6D5 -26 1215 1268
INTSPR 0014 -1584
INTTBL EC84 -2419
INTZBS 0010 -182 3616
INVFLG 02B2 -421 4486 4488 4527
IOC1 E4D6 823 -830
IOC1A E4DB -831 836
IOC2 E4F3 847 -849
IOC6 E514 866 -873
IOC7 E519 -879
IOC8 0340 -517 831 1076
IOCBS 0020 -202 832 1075
IOCBS2 0010 -200 201 795
IOCREF 00FF -107 787 865 903 915
IRGEN D20E -679 1306 1308 1324 1326 2000 2034 2332 2374 4117
IRG0 T D20E -664 1302 1318
ISLO D 00D 3233 -3257
<table>
<thead>
<tr>
<th>Variable</th>
<th>Address 1</th>
<th>Address 2</th>
<th>Address 3</th>
<th>Address 4</th>
<th>Address 5</th>
<th>Address 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDEVN</td>
<td>0040</td>
<td>-2896</td>
<td>3036</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PENH</td>
<td>D40C</td>
<td>-746</td>
<td>1421</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PENV</td>
<td>D40D</td>
<td>-747</td>
<td>1419</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHACR1</td>
<td>FC9F</td>
<td>-5313</td>
<td>5316</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHACRS</td>
<td>FC9D</td>
<td>4625</td>
<td>4651</td>
<td>5211</td>
<td>-5312</td>
<td>5526</td>
</tr>
<tr>
<td>PHCHLO</td>
<td>EE7F</td>
<td>-2940</td>
<td>3002</td>
<td>3003</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHCLOS</td>
<td>EE6C</td>
<td>2913</td>
<td>-3013</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHINIT</td>
<td>EE7B</td>
<td>2918</td>
<td>-2932</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHOPEN</td>
<td>EE9F</td>
<td>2912</td>
<td>-2972</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHPUT</td>
<td>EF14</td>
<td>2964</td>
<td>-3059</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHSTAT</td>
<td>EE81</td>
<td>2916</td>
<td>-2954</td>
<td>2972</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHSTLD</td>
<td>EE7D</td>
<td>-2939</td>
<td>2956</td>
<td>2957</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHWIT</td>
<td>EE7E</td>
<td>2915</td>
<td>-2982</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIA</td>
<td>D300</td>
<td>-751</td>
<td>752</td>
<td>753</td>
<td>754</td>
<td>755</td>
</tr>
<tr>
<td>PIRO</td>
<td>E6F3</td>
<td>-1281</td>
<td>1577</td>
<td>1578</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIRO1</td>
<td>FFDC</td>
<td>5782</td>
<td>-5795</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIRO2</td>
<td>FFF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIRO3</td>
<td>FFC8</td>
<td>5777</td>
<td>-5780</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIRO4</td>
<td>FFE8</td>
<td>5779</td>
<td>5786</td>
<td>-5793</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIRO5</td>
<td>FFBE</td>
<td>4082</td>
<td>-5775</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIROH</td>
<td>O0E6</td>
<td>1252</td>
<td>-1577</td>
<td>1578</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIROL</td>
<td>O0F3</td>
<td>1284</td>
<td>-1579</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLACR1</td>
<td>FCAA</td>
<td>-5323</td>
<td>5326</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLACRS</td>
<td>FCA8</td>
<td>4643</td>
<td>4674</td>
<td>5235</td>
<td>-5322</td>
<td>5543</td>
</tr>
<tr>
<td>PLOT</td>
<td>0050</td>
<td>-1627</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLUS</td>
<td>ECF8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLYARG</td>
<td>05E0</td>
<td>-633</td>
<td>634</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLYEVL</td>
<td>D840</td>
<td>-591</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMBASE</td>
<td>D407</td>
<td>-742</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNML</td>
<td>E791</td>
<td>-1372</td>
<td>1579</td>
<td>1580</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNML1</td>
<td>E799</td>
<td>1373</td>
<td>-1375</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNMLH</td>
<td>00E7</td>
<td>1256</td>
<td>-1579</td>
<td>1580</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PNMLIL</td>
<td>0091</td>
<td>1258</td>
<td>1580</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POINT</td>
<td>0025</td>
<td>-105</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POKEY</td>
<td>D200</td>
<td>-650</td>
<td>651</td>
<td>652</td>
<td>653</td>
<td>654</td>
</tr>
<tr>
<td>POTGO</td>
<td>0010</td>
<td>-183</td>
<td>1307</td>
<td>1316</td>
<td>1325</td>
<td>1997</td>
</tr>
<tr>
<td>POKTAB</td>
<td>EDD0</td>
<td>2521</td>
<td>2522</td>
<td>-2699</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTA</td>
<td>D300</td>
<td>-752</td>
<td>1275</td>
<td>1347</td>
<td>1482</td>
<td>1489</td>
</tr>
<tr>
<td>PORTB</td>
<td>D301</td>
<td>-753</td>
<td>1276</td>
<td>1351</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POT</td>
<td>D200</td>
<td>-651</td>
<td>1499</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POT6</td>
<td>D206</td>
<td>-657</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POT8</td>
<td>D208</td>
<td>-662</td>
<td>1503</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCI2</td>
<td>E5AC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCI3</td>
<td>E5B7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCI4</td>
<td>E5C3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCI6</td>
<td>E5B2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDBAD</td>
<td>EE51</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDBYTE</td>
<td>F310</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDONLY</td>
<td>00B7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>0052</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECEIV</td>
<td>EAE0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECVDN</td>
<td>0039</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECVDS</td>
<td>EC60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECEVEN</td>
<td>EC1B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REDOE</td>
<td>0027</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RELONE</td>
<td>EAB1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RENAME</td>
<td>0020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>F11B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETUR1</td>
<td>F634</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETUR2</td>
<td>F621</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETUR3</td>
<td>FAE1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETURN</td>
<td>EAE0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIRGHI</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RIRGLO</td>
<td>007B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMARGN</td>
<td>0053</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNGER1</td>
<td>FAD8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNGER2</td>
<td>FAD6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNGERR</td>
<td>FAD1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNOGOK</td>
<td>FAC4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROWAC</td>
<td>0070</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROWCRS</td>
<td>0054</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROWINC</td>
<td>0079</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RRETRN</td>
<td>EBOE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSIRG</td>
<td>000A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTCLOK</td>
<td>0012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>0053</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAVADR</td>
<td>0068</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAVIO</td>
<td>0316</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAVMBC</td>
<td>0058</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBUFSZ</td>
<td>001D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCOLLP</td>
<td>EBOE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCREDT</td>
<td>0045</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCRENV</td>
<td>E410</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCRFLG</td>
<td>0288</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCRMEM</td>
<td>0093</td>
<td>-156</td>
<td>4768</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCRNOK</td>
<td>01DB</td>
<td>3685</td>
<td>-3687</td>
<td>3690</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCROLL1</td>
<td>0BB7</td>
<td>4721</td>
<td>-5164</td>
<td>5177</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCROLL2</td>
<td>0BCA</td>
<td>5169</td>
<td>5172</td>
<td>-5174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCROLL</td>
<td>0BAC</td>
<td>4925</td>
<td>-5159</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDLSTH</td>
<td>0231</td>
<td>-339</td>
<td>1423</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDLSTL</td>
<td>0230</td>
<td>-338</td>
<td>1425</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDMCTL</td>
<td>022F</td>
<td>-337</td>
<td>1427</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECTI</td>
<td>F301</td>
<td>-3885</td>
<td>4003</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECTX</td>
<td>F34C</td>
<td>-3919</td>
<td>3924</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td>E6A8</td>
<td>-1940</td>
<td>2225</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENDDS</td>
<td>E5CF</td>
<td>1861</td>
<td>1964</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENDEN</td>
<td>EBF2</td>
<td>1692</td>
<td>1943</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENDEV</td>
<td>E468</td>
<td>-74</td>
<td>1691</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SENDOR</td>
<td>EC8A</td>
<td>1779</td>
<td>1807</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SERIN</td>
<td>D20D</td>
<td>-663</td>
<td>2114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEROUT</td>
<td>D20D</td>
<td>-677</td>
<td>1952</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETBSZ</td>
<td>E343</td>
<td>3074</td>
<td>3078</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETDCB</td>
<td>EEE6</td>
<td>2961</td>
<td>3004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETL0P</td>
<td>EBF7</td>
<td>-1561</td>
<td>1562</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETTAB</td>
<td>FB2D</td>
<td>-4621</td>
<td>5685</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETVBL</td>
<td>EBE6</td>
<td>1224</td>
<td>1226</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETVXX</td>
<td>E45C</td>
<td>-70</td>
<td>1228</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETVBS</td>
<td>EDB6</td>
<td>2215</td>
<td>2242</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEX</td>
<td>0000</td>
<td>-3415</td>
<td>3675</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SFH</td>
<td>EF64</td>
<td>-3163</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHFAMT</td>
<td>006F</td>
<td>-277</td>
<td>4298</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHFLOK</td>
<td>02BE</td>
<td>-428</td>
<td>4092</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFT1</td>
<td>F811</td>
<td>4299</td>
<td>-4302</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFT2</td>
<td>F610</td>
<td>4348</td>
<td>-4351</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFT3</td>
<td>F5AA</td>
<td>-4298</td>
<td>4301</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHIFTU</td>
<td>F608</td>
<td>-4347</td>
<td>4350</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIDWAY</td>
<td>0053</td>
<td>-1624</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIGNON</td>
<td>F223</td>
<td>3485</td>
<td>-3733</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIN</td>
<td>0B81</td>
<td>-600</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIO</td>
<td>E959</td>
<td>1686</td>
<td>-1726</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI0INT</td>
<td>E944</td>
<td>1689</td>
<td>-1706</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI0INV</td>
<td>E465</td>
<td>-73</td>
<td>1688</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI0ORG</td>
<td>E944</td>
<td>-27</td>
<td>1384</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI0SB</td>
<td>F095</td>
<td>3242</td>
<td>-3361</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI0SPR</td>
<td>0014</td>
<td>-2727</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIOV</td>
<td>EAS9</td>
<td>-69</td>
<td>1685</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIRHI</td>
<td>00EB</td>
<td>-2423</td>
<td>2424</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRLO</td>
<td>000F</td>
<td>-2424</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIZEM</td>
<td>00C</td>
<td>-694</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIZEP0</td>
<td>00B</td>
<td>-690</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIZEP1</td>
<td>009</td>
<td>-691</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIZEP2</td>
<td>00A</td>
<td>-692</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIZEP3</td>
<td>00B</td>
<td>-693</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKCCL</td>
<td>D20F</td>
<td>-679</td>
<td>1716</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKRES</td>
<td>D20A</td>
<td>-676</td>
<td>2097</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKSTAT</td>
<td>D20F</td>
<td>-665</td>
<td>1459</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOUND0</td>
<td>0041</td>
<td>-238</td>
<td>1715</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td>Value 4</td>
<td>Value 5</td>
<td>Value 6</td>
</tr>
<tr>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>SPACE</td>
<td>0020</td>
<td>-2899</td>
<td>2995</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPECIA</td>
<td>EF4B</td>
<td>3119</td>
<td>-3145</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPECIL</td>
<td>000E</td>
<td>-95</td>
<td>846</td>
<td>848</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPECLE</td>
<td>F23F</td>
<td>3582</td>
<td>-3781</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQR</td>
<td>BE1</td>
<td>-603</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRETNR</td>
<td>EB34</td>
<td>2116</td>
<td>-2121</td>
<td>2156</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRTSTA</td>
<td>0040</td>
<td>-3103</td>
<td>3377</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRTIM2</td>
<td>0006</td>
<td>-1219</td>
<td>1475</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRTIMR</td>
<td>022B</td>
<td>-333</td>
<td>1467</td>
<td>1472</td>
<td>1476</td>
<td>1526</td>
</tr>
<tr>
<td>SRTIRO</td>
<td>EB9B</td>
<td>2211</td>
<td>-2214</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRTIR1</td>
<td>EBC1</td>
<td>2238</td>
<td>-2241</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRTIR2</td>
<td>EB9</td>
<td>2260</td>
<td>-2265</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSFLAG</td>
<td>02FF</td>
<td>-473</td>
<td>1338</td>
<td>4326</td>
<td>5783</td>
<td>5785</td>
</tr>
<tr>
<td>SSSKCTL</td>
<td>0232</td>
<td>-340</td>
<td>1714</td>
<td>2285</td>
<td>2299</td>
<td>2321</td>
</tr>
<tr>
<td>STACK</td>
<td>S</td>
<td>0000</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STACKP</td>
<td>031B</td>
<td>-507</td>
<td>1727</td>
<td>2640</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATC</td>
<td>0053</td>
<td>-2748</td>
<td>2810</td>
<td>2825</td>
<td>2958</td>
<td>3042</td>
</tr>
<tr>
<td>STATIS</td>
<td>000D</td>
<td>-94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATU</td>
<td>F02B</td>
<td>3119</td>
<td>-3280</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>0030</td>
<td>-219</td>
<td>1846</td>
<td>1849</td>
<td>1864</td>
<td>1894</td>
</tr>
<tr>
<td>STATVH</td>
<td>0002</td>
<td>-2736</td>
<td>2737</td>
<td>2814</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATVL</td>
<td>00E8</td>
<td>-2737</td>
<td>2812</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STICK0</td>
<td>027B</td>
<td>-379</td>
<td>1487</td>
<td>1491</td>
<td>1509</td>
<td></td>
</tr>
<tr>
<td>STICK1</td>
<td>0279</td>
<td>-380</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STICK2</td>
<td>027A</td>
<td>-381</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STICK3</td>
<td>027B</td>
<td>-382</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STIMER</td>
<td>D209</td>
<td>-675</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STLOOP</td>
<td>EB77</td>
<td>-1482</td>
<td>1494</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORE</td>
<td>F917</td>
<td>4190</td>
<td>4204</td>
<td>4210</td>
<td>4214</td>
<td>4222</td>
</tr>
<tr>
<td>STORE1</td>
<td>F91D</td>
<td>-4748</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRBEG</td>
<td>FC5C</td>
<td>4545</td>
<td>-5261</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRERR</td>
<td>F942</td>
<td>4764</td>
<td>-4768</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIG0</td>
<td>02B8</td>
<td>-391</td>
<td>1498</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIG1</td>
<td>02B5</td>
<td>-392</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIG2</td>
<td>02B6</td>
<td>-393</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIG3</td>
<td>02B7</td>
<td>-394</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRL</td>
<td>EB90</td>
<td>-1497</td>
<td>1504</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRK</td>
<td>F946</td>
<td>4746</td>
<td>4754</td>
<td>4763</td>
<td>4767</td>
<td>-4770</td>
</tr>
<tr>
<td>STTMOT</td>
<td>EC75</td>
<td>1815</td>
<td>2252</td>
<td>-2396</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBBFL</td>
<td>E677</td>
<td>998</td>
<td>1059</td>
<td>-1138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBEND</td>
<td>FA7A</td>
<td>-4939</td>
<td>5493</td>
<td>5521</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBTMP</td>
<td>029E</td>
<td>-414</td>
<td>4752</td>
<td>4757</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUCCES</td>
<td>0001</td>
<td>-134</td>
<td>898</td>
<td>1850</td>
<td>1895</td>
<td>1940</td>
</tr>
<tr>
<td>SUSUAL</td>
<td>EB3B</td>
<td>-2124</td>
<td>2148</td>
<td>2162</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SV7H</td>
<td>00E8</td>
<td>-1523</td>
<td>1524</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SV7L</td>
<td>0073</td>
<td>-1524</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWAP1</td>
<td>FCB3</td>
<td>4384</td>
<td>4398</td>
<td>4418</td>
<td>4424</td>
<td>4429</td>
</tr>
<tr>
<td>SWAP2</td>
<td>FCC2</td>
<td>-5339</td>
<td>5346</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWAP3</td>
<td>FCD7</td>
<td>5337</td>
<td>-5350</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWAPA</td>
<td>FCB9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMPFLG</td>
<td>007B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SWSTA</td>
<td>0080</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYIRIG</td>
<td>E706</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYIRIG2</td>
<td>E71B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYIRIG8</td>
<td>E762</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYIRIG9</td>
<td>E76F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYIRIGA</td>
<td>E77A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYIRIGB</td>
<td>E78F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYRTI</td>
<td>E790</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYRTI2</td>
<td>E78B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB1</td>
<td>E7BA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB2</td>
<td>E7D6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB3</td>
<td>E7E5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB4</td>
<td>E832</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB5</td>
<td>E8C3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB6</td>
<td>E873</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB7</td>
<td>E844</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB8</td>
<td>E834</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSVB9</td>
<td>E857</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAB</td>
<td>FB10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAB1</td>
<td>FB23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAB2</td>
<td>FB2A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABMAP</td>
<td>02A3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLNENT</td>
<td>F0E3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLLEN</td>
<td>000E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDH1</td>
<td>00EA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDL0</td>
<td>00CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMP</td>
<td>023E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMP1</td>
<td>0312</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMP2</td>
<td>0314</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMP3</td>
<td>0315</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMPHI</td>
<td>0002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMPLD</td>
<td>003E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMER1</td>
<td>030C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMER2</td>
<td>0310</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMFLG</td>
<td>0317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMIT</td>
<td>EBA5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMIT1</td>
<td>EBCB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIMOUT</td>
<td>008A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TINDEX</td>
<td>0293</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMPCHR</td>
<td>0050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMPCOL</td>
<td>02B9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMPX1</td>
<td>029C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOADR</td>
<td>0066</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TONE1</td>
<td>0002</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TONE2</td>
<td>0001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOUT</td>
<td>E80A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOUT1</td>
<td>ED44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The values are in hexadecimal format.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Offset</th>
<th>Size</th>
<th>Offset</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCI5</td>
<td>E615</td>
<td>1028</td>
<td>1041</td>
<td>1052</td>
<td>1059</td>
</tr>
<tr>
<td>WDLR</td>
<td>EFC6</td>
<td>-3220</td>
<td>3223</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WFAK</td>
<td>F087</td>
<td>3337</td>
<td>-3350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WFAK1</td>
<td>F08C</td>
<td>3350</td>
<td>-3353</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WFL</td>
<td>F060</td>
<td>-3324</td>
<td>3333</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIRGHI</td>
<td>0000</td>
<td>-1659</td>
<td>2214</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WIRGLO</td>
<td>00B4</td>
<td>-1654</td>
<td>2213</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMODE</td>
<td>0289</td>
<td>-397</td>
<td>3161</td>
<td>3191</td>
<td>3195</td>
</tr>
<tr>
<td>WOK</td>
<td>EA3D</td>
<td>-1901</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td>0057</td>
<td>-1617</td>
<td>2807</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRITEC</td>
<td>0057</td>
<td>-2898</td>
<td>3069</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WRONLY</td>
<td>0083</td>
<td>-139</td>
<td>939</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSYNC</td>
<td>F087</td>
<td>3272</td>
<td>3297</td>
<td>3305</td>
<td>3389</td>
</tr>
<tr>
<td>WSIRG</td>
<td>00F</td>
<td>-1656</td>
<td>2209</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XBOOT</td>
<td>F046</td>
<td>3294</td>
<td>-3299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XITVBL</td>
<td>E905</td>
<td>3926</td>
<td>-3928</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XITVBL</td>
<td>E905</td>
<td>1228</td>
<td>1248</td>
<td>1418</td>
<td>-1571</td>
</tr>
<tr>
<td>XITVBL</td>
<td>E462</td>
<td>-72</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMTDON</td>
<td>003A</td>
<td>-229</td>
<td>1948</td>
<td>1961</td>
<td>2029</td>
</tr>
<tr>
<td>XXIT</td>
<td>E7E2</td>
<td>1413</td>
<td>-1418</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZERIT</td>
<td>EC6D</td>
<td>-2378</td>
<td>2381</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZERORM</td>
<td>F13B</td>
<td>-3587</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIOCB</td>
<td>0020</td>
<td>-199</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZOSRAM</td>
<td>F160</td>
<td>3585</td>
<td>-3610</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZOSRAM2</td>
<td>F163</td>
<td>-3612</td>
<td>3615</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZOSRM3</td>
<td>F16E</td>
<td>-3617</td>
<td>3619</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZTBUF</td>
<td>F04A</td>
<td>-3301</td>
<td>3303</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZTEMP1</td>
<td>00F5</td>
<td>-618</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZTEMP3</td>
<td>00F9</td>
<td>-620</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZTEMP4</td>
<td>00F7</td>
<td>-619</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Every effort has been made to ensure that this manual accurately documents this product of the ATARI Home Computer Division. However, due to the ongoing improvement and update of the computer software and hardware, ATARI, INC. cannot guarantee the accuracy of printed material after the date of publication and disclaims liability for changes, errors, or omissions.
TABLE OF CONTENTS

I. INTRODUCTION ................................................... I.1

II. DESCRIPTION OF HARDWARE ..................................... II.1
   A. ANTIC and CTIA ............................................ II.1
   B. POKEY .................................................. II.23
   C. SERIAL PORT ........................................... II.25
   D. INTERRUPT SYSTEM ..................................... II.28
   E. CONTROLLERS .......................................... II.30

III. HARDWARE REGISTERS ........................................ III.1
    A. PAL .................................................. III.1
    B. INTERRUPT CONTROL ................................... III.1
    C. TV LINE CONTROL ..................................... III.3
    D. GRAPHICS CONTROL ................................... III.4
    E. PLAYERS AND MISSILES ................................. III.9
    F. AUDIO ................................................ III.12
    G. KEYBOARD and SPEAKER ............................... III.15
    H. SERIAL PORT .......................................... III.17
    I. CONTROLLER PORTS ..................................... III.19

IV. SAMPLE DISPLAY PROGRAM ...................................... IV.1

V. HARDWARE REGISTER LISTS .................................... V.1
   A. ADDRESS ORDER ......................................... V.1
   B. ALPHABETICAL ORDER ................................... V.5

VI. FIGURES ......................................................... VI.1
    A. MEMORY MAP ........................................... VI.1
    B. NTSC and PAL DISPLAY ............................... VI.2
    C. SCHEMATICS ............................................ VI.3

APPENDIX A: USE OF PLAYER/MISSILE GRAPHICS WITH BASIC

APPENDIX B: MIXING GRAPHICS MODES

APPENDIX C: PINOUTS
I. INTRODUCTION

The ATARI (R) 800™ and ATARI 400™ Personal Computer Systems contain a 6502 microprocessor, 4 I/O chips, operating system ROM, expandable RAM, and several MSI chips for address decoding and data bus buffering. This manual is primarily intended to describe the 4 I/O chips in sufficient detail to allow experienced programmers to create assembly language programs, such as video games. All four Input/Output chips are controlled by the microprocessor by writing directly into their registers which are decoded to exist in microprocessor memory space just as RAM does. These I/O chips can also be interrogated by the microprocessor by reading similar registers.

Many registers are write only and cannot be read after they are written. In some cases, reading from the same address gives the value contained in a separate read only register. Some write only registers are strobes. No data bits are needed in this case since the presence of the address on the bus is what triggers the requested action. The usual convention is to use the STA (Store Accumulator) instruction for such registers. For example, STA WSYNC performs the wait for Sync function. STX (Store X) or STY (Store Y) would work just as well. In BASIC, a POKE could be used (the data could be anything). Reading a register is accomplished by using any of the load instructions (LDA, LDX etc.). In BASIC a PEEK would be used. When the hardware register names are defined in an equate list, the programmer can refer to the registers by name rather than using the addresses directly.

It is really not necessary for the programmer to know which I/O functions are performed by which of the 4 chips, however it does help in learning these functions.

This manual should be used in conjunction with the Operating System (OS) Manual, a 6502 programming manual, and the ATARI 400/800 Basic Reference Manual.

<table>
<thead>
<tr>
<th>CHIP NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANTIC</td>
<td>DMA(Direct Memory Access) control</td>
</tr>
<tr>
<td></td>
<td>NMI(Non-Maskable Interrupt) control</td>
</tr>
<tr>
<td></td>
<td>Vertical and Horizontal fine scrolling</td>
</tr>
<tr>
<td></td>
<td>Light pen position registers</td>
</tr>
<tr>
<td></td>
<td>Vertical line counter</td>
</tr>
<tr>
<td></td>
<td>WSYNC(wait for horizontal sync)</td>
</tr>
<tr>
<td>CTIA</td>
<td>Priority control (display of overlapping objects)</td>
</tr>
<tr>
<td></td>
<td>Color-Luminance control (colors and brightness assigned to all objects including DMA objects from ANTIC)</td>
</tr>
<tr>
<td></td>
<td>PLAYER-MISSILE objects (4 players and 4 missiles)</td>
</tr>
<tr>
<td></td>
<td>Graphics registers</td>
</tr>
<tr>
<td></td>
<td>Size control</td>
</tr>
<tr>
<td></td>
<td>Horizontal position control</td>
</tr>
<tr>
<td></td>
<td>Collision detection between all objects</td>
</tr>
<tr>
<td></td>
<td>Switches and triggers (miscellaneous I/O functions)</td>
</tr>
</tbody>
</table>
CHIP NAME | FUNCTION
---|---
POKEY | Keyboard scan and control
| Serial communications port (bidirectional)
| Pot scan (digitizes position of 8 independent pots)
| Audio generation (4 channels)
| Timers
| IRQ (maskable interrupt) control from peripherals
| Random number generator

PIA | Controller (Joystick) jacks read or write
| Peripheral control and interrupt lines
| IRQ (maskable) interrupt control from peripherals

Section II describes the hardware in some detail, including the various graphics modes. Section III lists the hardware registers one at a time, describing what each bit is used for. It is organized by functional groups (interrupts, graphics, audio, etc.). Section IV contains a sample display program. Section V contains various figures and block diagrams of the system. Sections VI and VII list the hardware registers in address order and alphabetical order. Section VII includes hex and decimal addresses, the OS shadow registers and the page numbers where more information can be found.
II. DESCRIPTION OF HARDWARE

A. ANTIC AND CTIA

**TV Display:** The ANTIC and CTIA chips generate the television display at the rate of 60 frames per second on the NTSC (US) system. The PAL (European) system is different and is described in the section on NTSC vs PAL. Each frame consists of 262 horizontal TV lines and each line is made up of 228 color clocks, as shown in figure VI-3. The 6502 microprocessor runs at 1.79 MHz. This rate was chosen so that one machine cycle is equivalent in length to two color clocks. One clock is approximately equal in width to two TV lines.

In any graphics mode, the display is divided up into small squares or rectangles called pixels (picture elements). The highest resolution graphics mode has a pixel size of 1/2 color clock by 1 TV line. A sample display list is given in section IV.

The current TV line may be determined by reading the vertical counter (VCOUNT). This register gives the line count divided by 2. There are 262 lines per frame so VCOUNT runs from 0 to 130 (0 to 155 on the PAL system). The 0 point occurs near the end of vertical blank (see figure VI.5). Vertical blank (VBLANK) is the time during which the electron beam returns back to the top of the screen in preparation for the next frame. The Atari 800 does not do interlacing, so each frame is identical unless the program which is being executed changes the display. Vertical sync (VSYNC) occurs during the fourth through sixth lines of vertical blank (VCOUNT = hex 7D through 7F). This tells the TV set where each frame starts. After VSYNC, there are 16 more lines of VBLANK for a total of 22 lines of VBLANK. The display list jump and wait instruction (to be described later) causes the display list graphics to start at the end of VBLANK.

**Operating System (OS):** The ATARI 400/800 comes with a 10K Operating System (OS) in ROM. The OS affects some of the hardware registers, so it will be mentioned from time to time in this manual. Refer to the OS manual for more details. The OS descriptions in this manual apply to the version that was being distributed when this manual was written.

The OS supports most of the hardware graphics modes (BASICS, GRAPHICS, PLOT, and DRAWTO commands). The OS always displays 24 background lines after the end of vertical blank. This convention is used at Atari to compensate for television sets which overscan. Most TV's are designed so that the edges of the picture are cut off. This is fine for ordinary broadcasts, but with a computer it is essential for all important information to be displayed on the screen. It is fairly common for four to eight color clocks at the right or left edge of the picture to overscan. A TV set that has excessive overscan may have to be readjusted to obtain a satisfactory display.
The OS uses 192 TV lines for its display and devotes the remaining 24 lines to overscan. It uses the standard display width of 160 color clocks. The hardware will allow displays of any length, but it is recommended that the standards be followed. The exception might be a border or other information which is merely decorative and not essential to use of the program.

**OS Shadowing:** Since many of the hardware registers are write-only and cannot be read the OS has a number of "shadow registers" in RAM. Every TV frame during vertical blank the OS takes the values in some of its shadow registers, and writes them out to the corresponding hardware register. The OS does attract color shifting on all of the color registers if ATRACT (on OS register) is negative. This is to prevent damage to the TV screen phosphors which can occur if the brightness is turned up too high and the same high-luminance display is left on for a long time. The OS also reads the joysticks and other controllers during vertical blank and stores the results in shadow registers, so that user programs do not have to include code to unpack the data. There are a few interrupt-related registers which the OS changes or reads during interrupt processing. Programs usually access the OS shadow registers instead of accessing the hardware directly. However, the OS shadowing can be disabled by changing the vertical blank and interrupt vectors (see OS manual).

**WSYNC:** In addition to a Vertical Blank Interrupt, which allows the Microprocessor to synchronize to the vertical TV display, this system also provides a Wait for Horizontal Sync (WSYNC) command that allows the microprocessor to synchronize itself to the TV horizontal line rate. This sync takes effect when the processor writes to an I/O location called WSYNC, whenever it desires horizontal synchronization. Writing to this address sets a latch which pulls to zero a pin on the microprocessor called READY. When READY goes to zero the microprocessor stops and waits. The latch is automatically reset (returning READY true) at the beginning of the next horizontal blank interval, releasing the microprocessor to resume program execution.

**Object DMA (Direct Memory Access):** The primary function of the Antic chip is to fetch data from memory (independent of the microprocessor) for display on the TV screen. It does this with a technique called "Direct Memory Access" or DMA. It requests the use of the memory address and data bus by sending a signal called HALT to the microprocessor, causing the processor to become "TRI-STATE" (open circuit) all during the next computer cycle. The ANTIC chip then takes over the address bus and reads any data it wishes from memory. Another name for this type of DMA is "cycle stealing". Once initiated, this DMA is completely and automatically controlled by the Antic chip without need for further microprocessor intervention.

There are two types of DMA: Playfield and Player-Missile (see Figure 11.2). The playfield DMA control circuit on the Antic chip resembles a small dumb microprocessor. By halting the main microprocessor it can fetch its own instructions from memory (the display list) addressed by its program counter(display list pointer). Each instruction defines the type (alpha character or memory map), and the resolution (size of bits on the screen), and the location of the data in memory which is to be displayed on the next group of lines.
In order to begin this DMA the main microprocessor must store a display list of instructions in memory, store data to be displayed in memory, tell the ANTIC where the display list is (initialize the display list pointer) and enable the DMA control flags on the ANTIC (DMACTL register).

In addition to the playfield DMA described above, the ANTIC chip simultaneously controls another DMA channel. This type of DMA addresses PLAYER-MISSILE graphics data stored in memory and passes the graphics data on to the CTIA chip graphics registers. This type of DMA (if enabled) occurs automatically, interspersed with the playfield DMA described previously. This PLAYER-MISSILE DMA has no display list or instructions, and is therefore much simpler than the PLAYFIELD DMA.

In addition to the two types of display DMA, the ANTIC chip also generates DMA addresses for the refresh of the dynamic memory RAM used in this system. This is also completely automatic and need be considered by the programmer only if he is concerned with real-time programming where an exact count of the computer cycles is important.

**Color-luminance:** A color-luminance register is used on the CTIA chip for each Player-Missile and Playfield type. Each color-luminance register is loaded by the microprocessor with a code representing the desired color and luminance of its corresponding Player-Missile or Playfield type. As the serial data passes through the CTIA chip it is "impressed" with the color and luminance values contained in these registers, before being sent to the TV display. In areas of the screen where there are no objects the background color (COLBK) is displayed. The CTIA also does collision detection (to be described later).

**Priority:** When moving objects, such as players and missiles, overlap on the TV screen (with each other or with Playfield) a decision must be made as to which object shows in front of the other. Objects which appear to pass in front of others are said to have Priority over them. Priority is assigned to all objects by the CTIA chip before the serial data from each object is combined with the other objects and sent to the TV screen.

The priority of objects can be controlled by the microprocessor by writing into the control register PRIOR. The functions of the bits in this register are given in the table in the PRIOR register description in section III.

**Players and Missiles:** The players and missiles are small objects which can be moved quickly in the horizontal direction by changing their position registers. They are called players and missiles because they were originally designed to be used in games for objects such as airplanes and bullets. However, there are many other possible applications for them. The four player-missile color registers, in conjunction with the four playfield color registers and the background color register, make it possible to display 9 different colors at the same time.
Figure II.2  OBJECT DISPLAY SOURCES
There are a total of four players and four missiles. The four missiles may be grouped together and used as a 5th player. These objects are positioned horizontally by 8 horizontal position registers (HPOS (X)). These registers may be reloaded at any time by the processor, allowing an object to be replicated many times across a horizontal TV line.

The shape of a player-missile is determined by the data in its graphics register (GRAF (X)). Players have independent 8 bit graphics registers. The four missiles have 2 bit registers (located within one address). These registers may also be reloaded at any time by the processor, although they are usually changed during horizontal blank time. The data in each graphics register is placed on the display whenever the horizontal sync counter equals the corresponding horizontal position register. The same data will be displayed every line unless the graphic registers are reloaded with new data.

The player-missile graphic registers may be reloaded by the microprocessor (GRAF (X)), or automatically from memory with direct memory access (DMA) (see figure 11.3). The programmer must place the object graphics in memory, write the player-missile base address (PMBASE), and enable player-missile DMA (DMACTL, GRACTL). The transfer of object graphics from memory to display is then fully automatic.

PMBASE specifies the most significant byte (MSB) of the address of the player-missile graphics. The location of the graphics for each object is determined by adding an offset to PMBASE *256 (decimal). The bytes between the base address and the missile data are not used by Antic, so they are available to the programmer.

Only the five most significant bits of PMBASE are used with single-line resolution and the six most significant bits are used with two-line resolution. This means that the location of the graphics in memory is restricted to certain page boundaries. Two-line resolution means that each byte of data is repeated for two lines. (see DMACTL, bit 4). 640 (decimal) bytes (5X128) are required for two-line resolution and 1280 bytes (5x256) for one-line resolution.

Each byte in the player graphics area represents eight pixels which are to be displayed on the corresponding line(s) of the TV screen. A 1 indicates that the player's color-lum is to be displayed in that pixel. The graphics may be anything, not just rectangles like the ones in figure II.3. The player graphics may fill the entire height of the screen or they may be only a couple of lines high if the rest of the display data is all 0's. Each byte in the missile display also represents eight pixels, two pixels for each missile. Each pixel may be 1, 2, or 4 color clocks, and is determined by the SIZE registers.

Playfield: Playfield is always generated by DMA. There are four playfields, each identified by its own color-lum register and collision detection. Playfield is generated by two different DMA techniques: memory map and character. Both methods provide lists of instructions in memory, independent of the player-missile generation.
Player-Missile Base Address (PMBASE) = MSB of address. Resolution is controlled by bit 4 of DMACTL.

<table>
<thead>
<tr>
<th>ADDRESS (hex)</th>
<th>OFFSET (hex)</th>
<th>PMBASE*100(hex)</th>
<th>MISSILE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>+180</td>
<td>+300</td>
<td>3 2 1 0</td>
<td>M</td>
</tr>
<tr>
<td>+200</td>
<td>+400</td>
<td>P0</td>
<td></td>
</tr>
<tr>
<td>+280</td>
<td>+400</td>
<td>P1</td>
<td></td>
</tr>
<tr>
<td>+300</td>
<td>+600</td>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>+380</td>
<td>+700</td>
<td>P3</td>
<td></td>
</tr>
<tr>
<td>+400</td>
<td>+800</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Player-Missile Vertical screen map in memory

Missile TV SCREEN

Horizontal position for each object is set independently by 8 horizontal position registers.

Each section of memory maps directly onto total height of TV screen. Object vertical position is determined only by its location in its section of memory. One byte of memory equals 1 or 2 television lines vertically.

Figure II.2 PLAYER - MISSILE DMA
Unlike players and missiles, there are no horizontal position registers for playfield. Each player can only have one byte of display per line. Playfield, on the other hand, may require up to 48 bytes per line because it can fill the entire width of the screen.

There are three different playfield widths: narrow (128 color clocks), standard (160 color clocks), and wide (192 color clocks). The width is selected by storing into DMACTL. The advantage of a narrower width is that less RAM is required and fewer machine cycles are stolen for DMA. The OS graphics modes use the standard screen width.

**Display List:** The display list is a sequence of display instructions stored in memory. These instructions are either one (1) byte or three (3) bytes long. The display list can be considered a display program, and the Display List Counter that fetches these instructions can be thought of as a display program counter. (10 bit counter plus 6 bit base register.)

The display list counter can be initialized by writing to DLISTH and DLISTL. (or OS shadow registers SDLSTH and SDLSTL). Once initialized this counter value is used to address the display list, fetch the instruction, display one (1) to sixteen (16) lines of data on the TV screen, increment the Display List Counter, fetch the next display instruction, and so on automatically without microprocessor control (see DLISTL and DLISTH). DLISTL and DLISTH should be altered only during vertical blank or when DMA is disabled (see DMACTL).

Each instruction defines the type (alpha character or memory map) and the resolution (size of bits on screen) and the location of data in memory to be displayed for a group (1 to 16) of lines. Each group of lines is called a display block.

THE DISPLAY LIST CANNOT CROSS A 1K BYTE MEMORY BOUNDARY UNLESS A JUMP INSTRUCTION IS USED.
Display Instruction Format: Each instruction consists of either an opcode only, or of an opcode followed by two (2) bytes of operand.

- **Opcode** -------Single Byte Display Instruction
- **Operand** -------Triple Byte Display Instruction

The opcode is always fetched first and placed in the Instruction Register. This opcode defines the type of instruction (1 or 3 bytes) and will cause two more bytes to be fetched if needed. If fetched, these next two (2) bytes will be placed in the Memory Scan Counter, or in the Display List Counter (if the instruction is a Jump).

Display Instruction Register (IR): This register is loaded with the opcode of the current display list instruction. It cannot be accessed directly by the programmer. There are three basic types of display list instructions: blank, jump, and display.

### Blank (1-byte)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
</table>

This instruction is used to create 1 to 8 blank lines on the display (blackground color).

- D7 = display list instruction interrupt
- D6 - D4 = 0-7 = 1-8 blank lines
- D3 - D0 = 0 = blank

### Jump (3-bytes)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>X</th>
<th>X</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

This instruction is used to reload the Display List Counter. The next two bytes specify the address to be loaded (LSB first).

- D7 = display list instruction interrupt
- D6 = 0 = jump (creates one blank line on display)
- 1 = jump and wait until end of next vertical blank
- D5-D4 = X = don’t care
- D3-D0 = 1 = jump

### Display (1 or 3 bytes)

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

This instruction specifies the type of display for the next display block.

- D7 = display list instruction interrupt
- D6 = 0 = 1 byte instruction
- 1 = 3 byte instruction (reload Memory Scan Counter using address in next two bytes, LSB first)
- D5 = vertical scroll enable
- D4 = horizontal scroll enable
- D3-D0 = 2-F = display mode (memory or character map - see following pages).

II.8
## Horizontal Scrolling

<table>
<thead>
<tr>
<th>HSCROL</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
</tr>
</thead>
</table>

## Vertical Scrolling

<table>
<thead>
<tr>
<th>VSCROL</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
<th>XX</th>
</tr>
</thead>
</table>

## Load Memory Scan

| LD MEM SCAN | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |

## Display Instruction Interrupt

| INST INTERRUPT | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |

### Character Mode Instructions

- **BLK 1**: Blank 1 line
- **" 2**: Blank 2 lines
- **" 3-7**: Blank 3 thru 7 lines
- **" 8**: Blank 8 lines

- **JMP**: Jump (3 byte instruction)

- **JVB**: Jump & wait for Vert. Blank (also 3 byte)

### Memory Map Mode Instructions

- **CHR (40,2,8)**
  - 02 12 22 32 42 52 62 72 82 92 A2 B2 C2 D2 E2 F2
- **CHR (40,4,8)**
  - 04 14 24 34 44 54 64 74 84 94 A4 B4 C4 D4 E4 F4
- **CHR (40,4,16)**
  - 05 15 25 35 45 55 65 75 85 95 A5 B5 C5 D5 E5 F5

- **MAP (40,4,8)**
  - 08 18 28 38 48 58 68 78 88 98 A8 B8 C8 D8 E8 F8

- **CHR (80,2,4)**
  - 09 19 29 39 49 59 69 79 89 99 A9 B9 C9 D9 E9 F9

- **CHR (80,4,8)**
  - 0A 1A 2A 3A 4A 5A 6A 7A 8A 9A AA BA CA DA EA FA

- **CHR (160,2,2)**
  - 0B 1B 2B 3B 4B 5B 6B 7B 8B 9B AB BB CB DB EB FB

- **CHR (160,2,1)**
  - 0C 1C 2C 3C 4C 5C 6C 7C 8C 9C AC BC CC DC EC FC

- **CHR (160,4,2)**
  - 0D 1D 2D 3D 4D 5D 6D 7D 8D 9D AD BD CD DD ED FD

- **CHR (160,4,1)**
  - 0E 1E 2E 3E 4E 5E 6E 7E 8E 9E AE BE CE DE EE FE

- **CHR (320,2,1)**
  - 0F 1F 2F 3F 4F 5F 6F 7F 8F 9F AF BF CF DF EF FF

### Figure II.3

**DISPLAY INSTRUCTION OPCODES**

II.9
Bit 7 of a display list instruction can be set to create a display list interrupt if bit 7 of NMIEN is set. The display list interrupt code can change the colors or graphics during the middle of the TV display. The type of interrupt is determined by checking NMIST. NMIRES clears NMIST. The current OS will vector through VDSLST (Hex 200 and 201) to the user's display list interrupt routine. See the OS manual for programming details.

Bits 5 and 4 of a display type of display list instructions are used to enable vertical and horizontal scrolling. The amount of scrolling depends on the values in the VSCROL and HSCROL registers (to be described later).

**Memory Scan Counter:** This counter is not directly accessible by the programmer. It is loaded with the value in the last 2 bytes of a 3 byte (non-Jump) instruction.

This counter points to the location (address) in memory of data to be directly displayed (memory map display) or to the location of character name strings to be indirectly displayed (character display).

A single byte instruction does not reload this counter. This implies a continuation in memory of data to be displayed from that displayed by the previous instruction. Since this counter really consists of 4 bits of register and 12 of actual counter, a continuous memory block cannot cross 4K byte memory boundaries, unless the counter is repositioned with a 3 byte Load Memory Scan Counter instruction.

Memory Map Display Instructions: Data in memory (addressed by the Memory Scan Counter) is displayed directly when executing a memory (bit) map display instruction. As data is being displayed it is also stored in a shift register so that it can be redisplayed for as many TV lines as required by the instruction.
In Instruction Register (IR) display modes 8 through F, one or two bits of memory are used to specify what is to be displayed on each pixel of the screen. Pixel sizes range from 1/2 clock by 1 TV line to 4 clocks by 8 TV lines. The OS and BASIC support most of these graphics modes (BASIC GRAPHICS command). Two modes, C and E, are not supported by the OS. These modes have rectangular pixels, which are approximately twice as wide as they are high.

In IR mode F, only one color (COLPF2) can be displayed. Two different luminances are available. If a bit is a zero, then the luminance of the corresponding pixel comes from COLPF2. If the bit is a one, then the luminance is determined by the contents of COLPF1 (abbreviated to PF1).

In IR modes 9, B, and C, two different colors can be displayed. A zero indicates background color and a one indicates PFO color. The difference between the various modes is in the size of the pixels.

In IR modes 8, A, D, and E, two bits are used to specify the color of each pixel. This allows four different colors to be displayed. However, only four pixels can be packed into each byte, instead of eight as in the previous modes. The bit assignments are shown below.

<table>
<thead>
<tr>
<th>SHIFT REGISTER</th>
<th>7 6 5 4 3 2 1 0</th>
<th>7 6 5 4 3 2 1 0</th>
</tr>
</thead>
</table>

2 bits form
one pixel
Memory Map Display Modes

<table>
<thead>
<tr>
<th>OS and Inst.</th>
<th>Colors per</th>
<th>Pixels per</th>
<th>Bytes per</th>
<th>Scan Line</th>
<th>Line Pixel</th>
<th>Pixel Bit</th>
<th>Color Values</th>
<th>Color Select</th>
<th>Mode</th>
<th>Reg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Reg.</td>
<td>Line</td>
<td>Line</td>
<td>Line</td>
<td>Pixels</td>
<td>Bits</td>
<td>Clocks</td>
<td>Reg.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modes HEX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>4</td>
<td>40</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>00</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>2</td>
<td>80</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>A</td>
<td>4</td>
<td>80</td>
<td>20</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>00</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td>2</td>
<td>160</td>
<td>20</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>C</td>
<td>2</td>
<td>160</td>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>D</td>
<td>4</td>
<td>160</td>
<td>40</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>00</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF2</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>E</td>
<td>4</td>
<td>160</td>
<td>40</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>00</td>
<td>BAK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>1½</td>
<td>320</td>
<td>40</td>
<td>1</td>
<td>½</td>
<td>1</td>
<td>0</td>
<td>PF2</td>
<td>PF1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>PF1</td>
<td>(LUM)</td>
</tr>
</tbody>
</table>
Character Display Instructions: The first step in using the character map mode is to create a character set in memory (or the built-in OS character set at hex E000 may be used). The character set contains eight bytes of data for the graphics for each character. The meaning of the data depends on the mode. The character set can contain 64 or 128 characters, also depending on the mode. The MSB (Most Significant Byte) of the address of the character set is stored in CHBASE (or the OS Shadow CHBAS). Only the most significant six or seven bits of CHBAS are used (see CHBASE description in section III). The other one or two bits and the LSB of the address are assumed to be zero, so the character set must start at an acceptable page boundary.

The next step is to set up the display list for the desired mode. Then the actual display is set up. This consists of a string of character names or codes. Each name takes one byte. The last 6 or 7 bits of the name selects a character. For a 64 character set, the name would range from 0 through 63 (decimal). For a 128 character set, the range would be 0 through 127 (decimal). The upper one or two bits of the name byte are used to specify the color or other special information, depending on the mode.

Character names (codes) are fetched by the memory scan counter, and are placed in a shift register. On any given line of display the shift register rotates, changing only the name portion of the character address, as shown below.

After a full line of character data has been displayed the line counter will increment. The next line again addresses all characters by name for that line number.

In 20 character per line modes the seven most significant bits of CHBASE are used. This requires that the character set to start upon a 512 byte memory boundary. The set must contain 64 characters, 8 bytes each, giving a total of 512 bytes for the set.

In 40 character per line modes the six most significant bits of CHBASE, forcing the character set to start on a 1K byte memory boundary. The set must have 128 characters of 8 bytes each. This gives a total of 1024 bytes for the set.

<table>
<thead>
<tr>
<th>Hex Code</th>
<th>Graphics Mode</th>
<th>Chars. Per Line</th>
<th>Number of Colors</th>
<th>Bytes per Char.</th>
<th>Number of Char. in set</th>
<th>Bytes in Char Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>40</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>1024</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>40</td>
<td>2</td>
<td>8</td>
<td>128</td>
<td>1024</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>40</td>
<td>4</td>
<td>8</td>
<td>128</td>
<td>1024</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>40</td>
<td>4</td>
<td>8</td>
<td>128</td>
<td>1024</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>20</td>
<td>5</td>
<td>8</td>
<td>64</td>
<td>512</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>20</td>
<td>5</td>
<td>8</td>
<td>64</td>
<td>512</td>
</tr>
</tbody>
</table>
Character Display

(20 Character per line mode example)

Codes (names)
Stored in
Shift Register

Internal codes for characters in memory

F
X
Z

Shift Register

Color Register Select

Address portion of Character name

CHBASE

not used

Line Counter

Character Data Address

Character Set in Memory

W

Addresses data in character set
and displays on the TV

Y

Color assigned by color register selected

0
TV
1
Scan
2
Lines
3

4

5

6

7
There are six character map modes, IR modes 2 through 7. Modes 2, 6 and 7 are supported by the OS and BASIC (GRAPHICS 0, 1 and 2).

In IR modes 6 and 7, the upper two bits of each character name select one of four playfield colors. For each data bit that contains a one, the selected playfield color is displayed. For each zero data bit, the background color is displayed. The four character colors plus the background color gives a total of five different colors. The mode 6 characters are eight lines high and the mode 7 characters are sixteen lines high (each data byte is displayed for two lines).

In IR modes 4 and 5, each character is only four pixels wide instead of eight (as in the other modes). Two bits per pixel of data are used to select one of three playfield colors, or background. Seven name bits are used to select the character. If the most significant name bit is a zero then data of 10 (binary) selects PF1. If the name bit 7 is one, then data bits of 10 select PF2. This makes it possible to display two characters with different colors, using the same data but different name bytes.

In IR modes 2 and 3, each pixel is half of a color clock in width. This makes it possible to have forty-eight-pixel-wide characters in a standard width line. These modes are similar to memory mode F in that two luminances can be displayed, but only one color is available at a time. In IR mode 3, each character is 10 lines high. This makes it possible to define lower case characters with descenders. The last fourth of the character set (name bits 5 and 6 equal to one) is lowered. The hardware takes the first two data bytes and moves them to the bottom of the character, displaying two blank lines at the top of the character (see next page).

In IR modes 2 and 3, bit 7 of the character name is used for inverse video or blanking. This is controlled by CHACTL (Character Control). If bit 2 of CHACTL is a one then all of the characters will be displayed upside down, regardless of mode. If CHACTL bit 1 is set, then each character which has bit 7 of its name set will be displayed in inverse video (the luminances will be reversed). If CHACTL bit 0 is set, then each character which has bit 7 set will be blanked (only background will be displayed). Characters can be blinked on and off by setting name bit 7 to 1 and toggling CHACTL bit 0. Inverse video and blank apply only to IR modes 2 and 3. If both inverse video and blank are set then the character will appear as an inverse video blank character (solid square).

**Hardware Collision Detection:** 60 bits of collision register are provided to detect and store overlap (hits) between players, missiles and playfield. These collisions can be read by the microprocessor from addresses D000 through D00F. There are no bits for missile to missile collisions.

- 16 bits for Missile to Playfield
- 16 bits for Player to Playfield
- 16 bits for Missile to Player
- 12 bits for Player to Player (PO to PO always reads as zero, etc.)

The 1/2 clock memory map mode (IR code 1111) and the 1/2 clock Character mode (IR codes 0011 and 0010) are both playfield type 2 collisions and will be stored in bit 2 of the playfield collision registers.
IR Mode 3—Upper and Lower Case

Upper Case

Data

G

Actual Display

G

Lower Case

g
## Character Map Display Modes

<table>
<thead>
<tr>
<th>OS and Inst.</th>
<th>Colors per Inst.</th>
<th>Chars. per Line</th>
<th>Scan per Char.</th>
<th>Color</th>
<th>Data</th>
<th>Color</th>
<th>Bits</th>
<th>Select</th>
<th>Values</th>
<th>Color Bits</th>
<th>Color</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC Reg.</td>
<td>HEX Mode</td>
<td>Line Pixel</td>
<td>Pixel Name</td>
<td>Data</td>
<td>Select</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1.5</td>
<td>40</td>
<td>8</td>
<td>0.5</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>PF2</td>
<td>1</td>
<td>PF1 (LUM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-3</td>
<td>1.5</td>
<td>40</td>
<td>10</td>
<td>0.5</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>PF2</td>
<td>1</td>
<td>PF1 (LUM)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-4</td>
<td>5</td>
<td>40</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>Bit 7</td>
<td>00</td>
<td>BAK</td>
<td>0</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
<td>01</td>
<td>PF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-5</td>
<td>5</td>
<td>40</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>Bit 7</td>
<td>00</td>
<td>BAK</td>
<td>0</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 0</td>
<td>01</td>
<td>PF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-6</td>
<td>5</td>
<td>20</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>BAK</td>
<td>0</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>PF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-7</td>
<td>5</td>
<td>20</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>BAK</td>
<td>0</td>
<td>PF0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>PF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>PF2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>PF3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Vertical and Horizontal Fine Scrolling:** Playfield objects are difficult to move smoothly. Memory map playfield can be moved by rewriting sections of memory. However, this is extremely time-consuming if large sections of the screen must be moved smoothly. Character playfield objects can be moved easily in a jerky fashion by changing the memory scan counter. However, this results in a large position jump from one character position to another, not a smooth motion. For this reason hardware registers (VSCROL and HSCROL) and counters are provided to allow smooth horizontal or vertical motion, up to one character width horizontally and up to one character height vertically. After this much smooth motion has been done by increasing the value in these registers, memory is rewritten or the memory scan counter is modified and smooth motion is resumed for another character distance.

**Vertical Scrolling:** A zone of playfield on the screen can be scrolled upward by using VSCROL and bit 5 of the display list instruction. The display blocks at the upper and lower boundaries of the zone must have a variable vertical size. In particular, the first display block within that zone must be shortened from the top, and the last display block must be shortened from the bottom (i.e. not all of the top and bottom blocks will be displayed).

The vertical dimension of each display block is controlled by a 4 bit counter within the ANTIC, called the ‘Delta Counter’ (DCTR). Without vertical scrolling, it starts at 0 on the first line, and counts up to a standard value, determined by the current display instruction. (Ex: for upper and lower case text display, the end value is 9. For 5 color character displays, it is 7 or 15.)

If bit 5 of the instruction remains unchanged between consecutive display blocks, then the second block is displayed in the normal fashion. If bit 5 of the instruction goes from 1 to 0 between two consecutive display blocks, the second block will start with Delta = 0, as usual, but will count up until Delta=VSCROL, instead of the standard value. This shortens that display block from the bottom.

To define a vertically scrolled zone, the most direct method is to set bit 5 to 1 in the first display instruction for that zone, and in all consecutive blocks but the last one. If the VSCROL register is not rewritten on the fly, this results in a total scrolled zone that has a constant number of lines (provided that the VSCROL value does not exceed the standard individual block size). If N is the standard block size, the top block will be \[N-VSCROL\] lines (\(N > VSCROL\)), and the last block will be \[VSCROL + 1\] lines: \((N-VSCROL) + (VSCROL + 1) = N + 1\). Shown on the following page is an example of a scrolled zone, top block, for 8 VSCROL values for \(N = 8\).

Horizontal scrolling is described under HSCROL in section III.
Simple Display List Example: BASIC starts out in OS graphics mode 0 which displays 40 characters across by 24 rows. This is IR mode 2 with a standard screen width. The OS sets up the display list near the top of RAM with room for the character names at the top of RAM. On a 32 K-byte machine, the display list would start at hex 7C20. The next three bytes are hex 70's to create 24 blank lines. The next byte is a hex 42. The 4 tells the hardware to reload the memory scan counter with the following address (7C40). This is the address of the data to be displayed. The 2 tells the hardware to display one line of IR mode 2 characters. The next 23 bytes specify 23 more lines of mode 2 characters. Hex 41 is the code for jumping and waiting until the end of the next vertical blank. The address to jump to is 7C20, the start of the display list. The next 960 bytes are the list of characters to be displayed, 40 bytes per line. The OS must set up the display list pointer (DLISTH and DLISTL) to the starting address of the display list (7C20). It also sets CHBASE to the MSB of the address of the character set (E0).

This is a simple example because only one mode is used and the memory scan counter is only loaded at one point. It is possible to have different modes on different lines, change character sets and colors, etc., as shown in the example in Section IV.
**OS Mode 0 Display List** (40 chars x 24 lines)

<table>
<thead>
<tr>
<th>Address (hex)</th>
<th>Data (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7C20</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>7C</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>.</td>
</tr>
<tr>
<td></td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>more</td>
</tr>
<tr>
<td></td>
<td>IR mode 2</td>
</tr>
<tr>
<td></td>
<td>instructions</td>
</tr>
<tr>
<td></td>
<td>24 blank</td>
</tr>
<tr>
<td></td>
<td>lines</td>
</tr>
<tr>
<td></td>
<td>reload</td>
</tr>
<tr>
<td></td>
<td>memory</td>
</tr>
<tr>
<td></td>
<td>scan</td>
</tr>
<tr>
<td></td>
<td>counter</td>
</tr>
<tr>
<td></td>
<td>with 7C4C</td>
</tr>
<tr>
<td></td>
<td>IR mode 2</td>
</tr>
<tr>
<td></td>
<td>7C20</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Wait for</td>
</tr>
<tr>
<td></td>
<td>end of</td>
</tr>
<tr>
<td></td>
<td>vertical</td>
</tr>
<tr>
<td></td>
<td>blank.</td>
</tr>
<tr>
<td>7C40</td>
<td>960</td>
</tr>
<tr>
<td></td>
<td>bytes of</td>
</tr>
<tr>
<td></td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>data</td>
</tr>
<tr>
<td></td>
<td>(character</td>
</tr>
<tr>
<td></td>
<td>names)</td>
</tr>
</tbody>
</table>

**Cycle Counting:** As explained previously, the ATARI 800 6502 microprocessor runs at a rate of 114 machine cycles per TV line (1.79 MHz). There are 262 lines per TV frame and 60 frames per second on the NTSC (US) system. (The PAL (European) system is different. See the section on NTSC vs. PAL.)

Each machine cycle is equivalent in length to 2 color clocks. There are 228 color clocks on a TV line. The highest resolution graphics modes have a pixel size of 1/2 color clock by 1 TV line. Horizontal blank takes 40 machine cycles. This is when the beam returns to the left edge of the screen in preparation for displaying the next TV line. A wait for Sync (WSYNC) instruction stops the 6502. The processor is restarted exactly 7 machine cycles before the beginning of the next TV line. The program can thus change graphics or colors during horizontal blank in preparation for the next line.

The ANTIC chip steals cycles from the 6502 in order to do memory refresh and fetch graphics data when needed. The general rule to remember is that each byte fetched from memory requires one machine cycle. If a display list memory map instruction extends over several lines then the data is only fetched on the first line. Memory refresh takes 9 cycles out of every line, unless preempted by a high-resolution graphics mode. Memory refresh continues during vertical blank.

Missile DMA takes one cycle per line in the one-line resolution mode and one cycle every other line in the two line resolution mode. Missile DMA can be enabled without doing player DMA. However, if player DMA is enabled then missile DMA will also be done (see DMACTL, GRACTL bits). Player DMA requires 4 cycles every one or two lines, depending on the resolution used.
Each fetch of a display list byte takes one cycle, so three cycles are required for a three byte instruction.

Player/missile and display list instruction fetch DMA take place during horizontal blank, if they are required for the next line.

In memory map modes, the graphics data is fetched as needed throughout the first line of the display list instruction, then saved by ANTIC for use in succeeding lines. In character modes, the character codes are fetched during the first line of each row of characters, along with the graphics data needed for that line. On the next lines, only the graphics data is fetched, since ANTIC remembers the character codes.

In the 40 x 24 character mode, with a standard screen width, most of the cycles during the top line of each row of characters are required to fetch the character codes and data, so there is only time for one memory refresh cycle instead of the usual nine. Less DMA is required with a narrow screen width so two memory refresh cycles would occur in this case.

The memory refresh is done fast enough to make up for the lost cycles in the high resolution modes. Once memory refresh starts on a line, it occurs every four cycles unless pre-empted by DMA.

All interrupts reach the 6502 near the end of horizontal blank. With standard or narrow screen widths, refresh DMA starts after the end of horizontal blank.

The time at which ANTIC does cycle stealing is deterministic, but depends on the graphics mode, screen width and whether or not horizontal scrolling is enabled. Horizontal scrolling requires extra graphics data; see HSCROL.

ANTIC does horizontal scrolling of an even number of color clocks by delaying the time at which it DMA's the data. To do an odd number of color clocks (which involves half of a machine cycle), ANTIC has a one color clock internal delay.

Theoretically, it is possible to write a program which changes graphics or colors "on the fly", i.e. during the middle of a TV line. However, with all the DMA going on, the cycle counting gets to be quite complicated, and is beyond the scope of this manual.

There are a number of delays associated with the display of graphics. These occur in the ANTIC and the CTIA. The ANTIC sends data to the CITA which adds in the color information. Thus the timing for changing colors on the fly is different from that for changing graphics on the fly.
Horizontal Blank DMA Timing

When DMA is enabled, cycles are stolen at the times shown below.

End of
Previous Line  Horizontal Blank

20 machine cycles (40 color clocks)

<table>
<thead>
<tr>
<th>5</th>
<th>4</th>
<th>1</th>
<th>1</th>
<th>4</th>
<th>2</th>
<th>8</th>
</tr>
</thead>
</table>
| WSYNC

1-9 refresh cycles.
char. and graphics
data DMA (depends on
graphic mode)

Interrupt
Address DMA (3-byte display list
instruction)
Player
Display list instruction fetch DMA
Missile DMA

Cycle Counting Example: This example uses the 40 character by 24 line
display list given on page II.24. This display list is 32 bytes long so
display list DMA takes 32 machine cycles. It takes 960 cycles to DMA the
characters and 8*960 to DMA the character data. The refresh DMA takes 9
cycles for each of 262 lines, except for the 24 lines where the characters
are read, where only 1 refresh cycle occurs.

<table>
<thead>
<tr>
<th>DMA description</th>
<th>Machine cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>display list</td>
<td>32</td>
</tr>
<tr>
<td>characters</td>
<td>40x24 = 960</td>
</tr>
<tr>
<td>character data</td>
<td>960x8 = 7680</td>
</tr>
<tr>
<td>refresh</td>
<td>262x9-24x8 = 2166</td>
</tr>
<tr>
<td>total</td>
<td>10838</td>
</tr>
</tbody>
</table>

Thus the total DMA per frame is 10838 machine cycles. One frame
is 262 lines with 114 machine cycles per line for a total of 29868 machine
cycles per frame. Thus 36% of each frame is required for DMA in OS graphics
mode 0.

NTSC vs. PAL Systems: There are two versions of the ATARI 800: the NTSC
(United States T.V. standard) and PAL (one of the European T.V. standards).
The PAL system has been designed so that most programs will run without
being modified. However, some differences may be noticeable. There is a
hardware register (PAL) which a program can read to determine which type of
system it is running on and adjust accordingly.
The PAL T.V. has a slower frame rate (50 Hz. instead of 60 Hz.) so games will be slower unless an adjustment is made. PAL has more T.V. lines per frame (312 instead of 262). The Atari 800 hardware compensates for this by adding extra lines at the beginning of vertical blank. Display lists do not have to be altered. However, their actual vertical height will be shorter. PAL ATARI 800 colors are similar to NTSC because of a hardware modification.

B. POKEY

Audio: There are 4 semi-independent audio channels, each with its own frequency, noise, and volume control. Each has an 8 bit "divide by N" frequency divider, controlled by an 8 bit register (AUDFX). (See audio-serial port block diagram.) Each channel also has an 8 bit control register (AUDCX) which selects the noise (poly counter) content, and the volume.

Frequency Dividers: All 4 frequency dividers can be clocked simultaneously from 64 KHZ or 15 KHZ. (AUDCTL bit 0). Frequency dividers 1 and 3 can alternately be clocked from 1.79 MHZ (AUDCTL bits 6 and 5). Dividers 2 and 4 can alternately be clocked with the output of dividers 1 and 3 (AUDCTL bits 4 and 3). This allows the following options: 4 channels of 8 bits resolution, 2 channels of 16 bit resolution, or 1 channel of 16 bit and 2 channels of 8 bit.

Poly Noise Counters: There are 3 polynomial counters (17 bit, 5 bit and 4 bit) used to generate random noise. The 17 bit poly counter can be reduced to 9 bits (AUDCTL bit 7). These counters are all clocked by 1.79 MHZ. Their outputs, however, can be sampled independently by the four audio channels at a rate determined by each channel's frequency divider. Thus each channel appears to contain separate poly counters (3 types) clocked at its own frequency. This poly counter noise sampling is controlled by bits 5, 6 and 7 of each AUDCX register. Because the poly counters are sampled by the "divide by N" frequency divider, the output obviously cannot change faster than the sampling rate. In these modes (poly noise outputted) the dividers are therefore acting as "low pass" filter clocks, allowing only the low frequency noise to pass.

The output of the noise control circuit described above consists of pure tones (square wave type), or polynomial counter noise at a maximum frequency set by the "divide by N" counter (low pass clock). This output can be routed through a high pass filter if desired (AUDCTL bits 1 and 2).
Audio Noise Filters:

Low pass noise cut off set by Divide by N counter.

Any channel noise output (without high pass filter)

Channel 1 output (with high pass filter)

Channel 2 output (with high pass filter)

Clock
High Pass Filters: The high pass filter consists of a "D" flip flop and an exclusive-OR Gate. The noise control circuit output is sampled by this flip flop at a rate set by the "High Pass" clock. The input and output of the flip flop pass through the exclusive-OR Gate. If the flip flop input is changing much faster than the clock rate, the signal will pass easily through the exclusive-OR Gate. However, if it is lower than the clock rate, the flip flop output will tend to follow the input and the two exclusive-OR Gate inputs will mostly be identical (11 or 00) giving very little output. This gives the effect of a crude high pass filter, passing noise whose minimum frequency is set by the high pass clock rate. Only channels 1 and 2 have such a high pass filter. The high pass clock for channel 1 comes from the channel 3 divider. The high pass clock for channel 2 comes from the channel 4 divider. This filter is included only if bit 1 or 2 of AUDCTL is true.

Volume Control: A volume control circuit is placed at the output of each channel. This is a crude 4 bit digital to analog converter that allows selection of one of 16 possible output current levels for a logic true audio input. A logic zero audio input to this volume circuit always gives an open circuit (zero current) output. The volume selection is controlled by bits 0 thru 3 of AUDCX. "Volume Control only" mode can be invoked by forcing this circuit's audio input true with bit 4 of AUDCX. In this mode the dividers, noise counters, and filter circuits are all disconnected from the channel output. Only the volume control bits (0 to 8 of AUDCX) determine the channel output current.

The audio output of any channel can be completely turned off by writing zero to the volume control bits of AUDCX. All ones gives maximum volume.

C. SERIAL PORT

The serial port consists of a serial data output (transmission) line, a serial data input (receiver) line, a serial output clock line, a bi-directional serial data clock line, and other miscellaneous control lines described in the Operating System Manual. Data is transmitted and received as 8 bits of serial data preceded by a logic zero start bit, and succeeded by a logic true stop bit. Input and output clocks are equal to the baud (bit) rate, not 16 times baud rate. Transmitted data changes when the output clock goes true. Received data is sampled when the input clock goes to zero.

Serial Output: The transmission sequence begins when the processor writes 8 bits of parallel data into the serial output register (SEROUT)(see audio and serial port block diagram). When any previous data byte transmission is finished the hardware will automatically transfer new data from (SEROUT) to the output shift register, interrupt the processor to indicate an empty (SEROUT) register (ready to be reloaded with the next byte of data), and automatically serially transmit the shift register contents with start-stop bits attached. If the processor responds to the interrupt, and reloads SEROUT before the shift register is completely transmitted, the serial transmission will be smooth and continuous.
Output data is normally transmitted as logic levels (+4V=true 0V=False). Data can also be transmitted as two tone information. This mode is selected by bit 3 of SKCTL. In this mode audio channel 1 is transmitted in place of logic true, and audio channel 2 in place of logic zero. Channel 2 must be the lower tone of the tone pair.

The processor can force the data output line to zero (or to audio channel 2, if in two tone mode) by setting bit 7 of SKCTL. This is required to force a break (10 zeros) code transmission.

**Serial Output Clock:** The serial output data always changes when the serial output clock goes true. The clock then returns to zero in the center of the output data bit time.

The baud (bit) rate of the data and clock is determined by audio channel 4 audio channel 2, or by the input clock, depending on the serial mode selected by bits 4, 5, and 6 of SKCTL. (See chart at end of this section.)

**Serial Input:** The receiving sequence begins when the hardware has received a complete 8 bit serial data word plus start and stop bits. This data is automatically transferred to the 8 bit parallel input register (SERIN), and the processor is interrupted to indicate an input data byte ready to read in SERIN. The processor must respond to this interrupt, and read SERIN, before the next input data word reception is complete, otherwise an input data "over-run" will occur. This over-run will be indicated by bit 5 of SKSTAT (if bit 5 of IRQST is not RESET (true) before next input complete), and means input data has been lost. This bit should be tested whenever SERIN is read. Bit 7 of SKSTAT should also be tested to detect frame errors caused by extra (or missing) data bits.

**Direct Serial Input:** The serial data input line can be read directly by the microprocessor if desired, ignoring the shift register, by reading bit 4 of SKSTAT.

**Bi-Directional Clock:** This clock line is used to either receive a clock from an external clock source for clocking transmitted or received data, or is used to supply a clock to external devices indicating the transmit or reception rate. This clock line direction is determined by the serial mode selected by bits 4, 5, and 6 of SKCTL. (See mode chart at the end of this section.) Transmitted data changes on the rising edge of this clock. Received data is sampled on the trailing edge of this clock.

**Asynchronous Serial Input:** Unclocked serial data (at an approximately known (+5%) rate) can be received in the asynchronous modes. The receive (input) shift register is clocked by audio channel 4. Channels 3 and 4 should be used together (AUDCTL bit 3 = 1) for increased resolution. In asynchronous modes, channels 3 and 4 are reset by each start bit at the beginning of each serial data byte. This allows the serial data rate to be slightly different from the rate set by channels 3 and 4.
Serial Mode Control: There are 6 useful modes (of the possible 8) controlled by bits 4, 5, and 6 of SKCTL. These are described on the next page.

Note that two tone output (bit 3 of SKCTL) may be used in any of these modes except for the bottom pair. This is because channel 2 is used to set the output transmit rate and is therefore not available for one of the two tones.

Note that the output clock rate is identical to the output data rate.

Serial Mode Control (see also register description SKCTL):

Force Break

<table>
<thead>
<tr>
<th>D7 D6 D5 D4 D3 D2 D1 D0</th>
<th>SKCTL REGISTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pot scan and keyboard CTRL</td>
<td></td>
</tr>
</tbody>
</table>

Two Tone Control

<table>
<thead>
<tr>
<th>Mode Control Bits</th>
<th>A=asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out</td>
<td>Out</td>
</tr>
<tr>
<td>D6 D5 D4 Rate</td>
<td>Clock Rate</td>
</tr>
<tr>
<td>0 0 0</td>
<td>ext</td>
</tr>
<tr>
<td>0 0 1</td>
<td>chan</td>
</tr>
<tr>
<td>0 1 0</td>
<td>chan</td>
</tr>
<tr>
<td>0 1 1</td>
<td>CH4</td>
</tr>
<tr>
<td>1 0 0</td>
<td>chan</td>
</tr>
<tr>
<td>1 0 1</td>
<td>CH4</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Chan</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Chan</td>
</tr>
</tbody>
</table>

Two tone (bit3) not useable in these modes
D. INTERRUPT SYSTEM

There are two basic types of interrupts defined on the microprocessor: NMI (non maskable interrupt) and IRQ (interrupt request). It is recommended that a thorough understanding of these interrupt types be acquired by reading all chapters concerning interrupts in the 6502 microprocessor programming and hardware manuals.

In this system NMI interrupts are used for video display and reset. IRQ interrupts are used for serial port communication, peripheral devices, timers, and keyboard inputs.

**NMI Interrupts:** Even though NMI interrupts are "unmaskable" on the microprocessor, this system has interrupt enable (mask) bits for NMI function. (Bits 6 and 7 of NMIEN) When these bits are zero NMI interrupts are disabled (masked) and prevented from causing a microprocessor NMI interrupt. (see NMIEN register description) The 3 types of NMI interrupts are:

1. **D7 = Instruction Interrupt** (during display time any display instruction with bit 7=1 will cause this interrupt to occur (if enabled) at the start of the last video line displayed by that instruction.)

2. **D6 = Vertical Blank Interrupt** (interrupt occurs (if enabled) at the beginning of the vertical blank time interval.)

3. **D5 = Reset Button Interrupt** (pushing the SYSTEM RESET button will cause this interrupt to occur.)

Since any of these interrupts will cause the processor to jump to the same NMI address, the system also has NMI status bits which may be examined by the processor to determine which source caused the NMI interrupt. Bits 5, 6, and 7 of NMIST serve this function (see NMIST register description). These status bits are set by the corresponding interrupt function (even if the interrupt is masked from the processor by NMIEN). The status bits may be reset together by writing to the address NMIRES.

Two of the interrupt enable bits (bits 6 and 7 of NMIEN) are cleared automatically during system power turn on and therefore these NMI interrupts are initially disabled (masked), preventing any power turn on service routine from being interrupted before proper initialization of registers and pointers.* They can then be enabled by the processor whenever desired, by writing into bits 6 and 7 of NMIEN. Except for the reset button interrupt, they can also be disabled by the processor by writing a zero into bits 6 or 7 of NMIEN. The reset button cannot be disabled, allowing an unstoppable escape from any possible "hangup" condition.

These NMI interrupt functions are each separated in time (to prevent overlaps) and converted to pulses by the system hardware, in order to supply NMI transitions required by the microprocessor logic.

* - NOTE: Bit 5 is never disabled and therefore the Reset Button should not be pressed during power turn on.
**IRQ Interrupts:** IRQ interrupts are all "maskable" together by one bit of the status register on the microprocessor. This bit is set to the disable condition automatically by power turn on to prevent interrupt of power turn on service routines.** In addition to this processor IRQ mask bit, there are separate system IRQ interrupt enable bits for each IRQ interrupt function (bits 0 thru 7 of IRQEN). These bits are not initialized by power turn on, and must be initialized by the program before enabling the processor IRQ. The 8 types of IRQ interrupts are:

- **D7 = BREAK KEY** (depression of the break key)
- **D6 = OTHER KEY** (depression of any other key)
- **D5 = SERIAL INPUT READY** (Byte of serial data has been received and is ready to be read by the processor in SERIN register).
- **D4 = SERIAL OUTPUT NEEDED** (Byte of serial data is being transmitted and SEROUT is ready to be written to again by the processor).
- **D3 = TRANSMISSION FINISHED** (serial data transmission is finished. Output shift register is empty).
- **D2 = TIMER #4** (audio divider #4 has counted down to zero)
- **D1 = TIMER #2** (audio divider #2 has counted down to zero)
- **D0 = TIMER #1** (audio divider #1 has counted down to zero)

In addition to the above IRQ interrupts (enabled by bits 0 through 7 of IRQEN and identified by status bits 0 thru 7 of IRQST) there are two more system IRQ interrupts which are generated over the serial bus Proceed and Interrupt lines.

- **D7 of PACTL = peripheral "A" interrupt status bit**
- **D0 of PACTL = peripheral "A" interrupt enable bit**
- **D7 of PBCTL = peripheral "B" interrupt status bit**
- **D0 of PBCTL = peripheral "B" interrupt enable bit**

These last two interrupts are automatically disabled by power turn on, and their status bits are reset by reading from port A register and port B register. (See PORTA, PACTL, PORTB, and PBCTL Register descriptions.)

The IRQEN register, like the NMIEN register, enables interrupts when its bits are 1 (logic true). The IRQST however (unlike the NMIST) has interrupt status bits that are normally logic true, and go to zero to indicate an interrupt request. The IRQST status bits are returned to logic true only by writing a zero into the corresponding IRQEN bit. This will disable the interrupt and simultaneously set the interrupt status bit to one. Bit 3 of IRQST is not a latch and does not get reset by interrupt disable. It is zero when the serial out is empty (out finished) and true when it is not.

** - NOTE: An NMI also disables the I bit.
## INTERRUPT SUMMARY

<table>
<thead>
<tr>
<th>NAME</th>
<th>FUNCTIONS</th>
<th>ENABLE</th>
<th>STATUS</th>
<th>STATUS</th>
<th>RESET</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMI</td>
<td>Display</td>
<td>NMIEN</td>
<td>NMIST</td>
<td>Address</td>
<td>NMIRES</td>
</tr>
<tr>
<td>INTERRUPTS</td>
<td>Instruction</td>
<td>(Bits 6 thru 7)</td>
<td>(Bits 5 thru 7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vert. Blank</td>
<td>Normally Zero</td>
<td>Normally Zero</td>
<td>(no interrupt)</td>
<td>(no interrupt)</td>
<td>(no interrupt)</td>
</tr>
<tr>
<td>Reset Button</td>
<td>(Disabled)</td>
<td>(no interrupt)</td>
<td>(no interrupt)</td>
<td>(no interrupt)</td>
<td>(status together)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEYS</td>
<td></td>
<td>IRQEN</td>
<td>IRQST</td>
<td>Reset (to true)</td>
<td>By Zero in</td>
</tr>
<tr>
<td>Serial</td>
<td></td>
<td>(Bits 0 thru 7)</td>
<td>(Bits 0 thru 7)</td>
<td>Corresponding</td>
<td>Bit of IRQEN</td>
</tr>
<tr>
<td>ports Timers</td>
<td>zero is</td>
<td>Normally True</td>
<td>(no interrupt)</td>
<td>(except Bit 3)</td>
<td></td>
</tr>
<tr>
<td>IRQ</td>
<td></td>
<td>(Disabled)*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERRUPTS</td>
<td>Peripheral A</td>
<td>D0 of PACTL</td>
<td>D7 of PACTL</td>
<td>Reset by</td>
<td>Reading PORT A</td>
</tr>
<tr>
<td></td>
<td>Normally Zero</td>
<td>Normally Zero</td>
<td>(no interrupt)</td>
<td>Register</td>
<td></td>
</tr>
<tr>
<td>Peripheral B</td>
<td></td>
<td>D0 of PBCTL</td>
<td>D7 of PBCTL</td>
<td>Reset by</td>
<td>Reading PORT B</td>
</tr>
<tr>
<td></td>
<td>Normally Zero</td>
<td>Normally Zero</td>
<td>(no interrupt)</td>
<td>Register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Disabled)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### E. CONTROLLERS

A variety of controllers can be plugged into the four jacks on the front of the console. This includes joysticks, paddle (pot), twelve-key keyboard, and light pen (when available).

The controller ports are read through the PORTA and PORTB registers and the POT and TRIG registers. The OS reads these registers during vertical blank and stores into its own RAM locations. These are STICK, PADDLO through PADDL7, PTRIG’S and STRIG’S. The OS sets up PORTA AND PORTB for input. This is done by setting PACTL or PORTB (Port Control) bit 2 to a 0 (to select the direction control register), then writing all 0’s to the desired port. PACTL (PBCTL) bit 2 is then changed back to a 1, allowing the program to read from the port. The ports can also be set up for output by writing 1’s instead of 0’s while the direction control mode is selected.

Joysticks: The joysticks have four switches, one each for right (R), left (L), back (B) and forward (F).

These switches are read through PORTA and PORTB. A fifth switch is activated by pressing the red trigger button. The trigger buttons are read from TRIGO through TRIG3. A value of 0 indicates that a button has been pressed and a 1 indicates that it has not been pressed.
The TRIG registers are normally read directly, but they can be used in a latched mode. Writing a zero to bit 2 of GRACtl disables the latches and sets them to 1. Writing a 1 to bit 2 enables the latches. If a joystick trigger button is pushed at any time while bit 2 of GRACtl is 1 the latch value will change to zero and stay that way. A program can use this to determine whether the joystick trigger buttons have ever been pressed during a certain period of time.

Paddles: The paddles come in pairs, so eight paddles can be connected to the four jacks. The paddles are read by storing into POTGO, then reading the POT registers at least 228 lines later. The values range from 0 (with the paddle turned to the right) to 228 (paddle turned counter-clockwise). The value indicates how many TV lines it takes to charge up the capacitor which is the series with the potentiometer. Turning the knob to the right lowers the resistance, so the capacitor charges up quickly. Turning the knob to the left increases the resistance and the charging time. The capacitor dump transistors are used to discharge the capacitors so that a new reading can be made. The POTGO command clears the counters and turns off the dump transistors to allow the capacitors to charge up. The ALLPOT register contains one bit for each paddle. When the capacitor has charged up to the threshold value the ALLPOT bit changes from one to zero and the POT register contains the correct readings. Bit 2 of SKCTL (Serial Port Control) enables fast pot scan. In this mode, it takes only two scan lines to charge up the capacitors to the maximum level instead of 228 lines. Bit 2 is first set to 0 to dump the capacitors. Then Bit 2 is set to 1 to start the pot scan. The fast pot scan is not as accurate as the normal scan mode. Bit 2 of SKCTL must be set to 0 to use normal scan mode. Otherwise, the capacitors will never dump. Note that some paddles have a range smaller than 0 to 228 due to differences in the pots. The left and right paddle triggers for each paddle pair are read from the left and right bits for the corresponding joystick (PORTA or PORTB).

Keyboard Controllers: Each keyboard controller has a twelve-key pad and plugs into a joystick controller port. The first step in using the keyboard is to select a row by setting the port direction to output and writing a 0 to the bit in the PORTA or PORTB register which selects the desired row (see PORTA, SECTION III). The other rows should have 1's written to them. Columns are read through the POT and TRIG registers (see controller PORT PINOUT chart in section III). Appendix H of the BASIC Reference Manual contains a Basic program which reads the controllers. The first and second columns of the keyboard use the same pins as the pots for the paddle controllers, so they are read by reading the POT (or PADDL) registers. When a button is pushed, the pot line is grounded, so the pot capacitors never charge up to the threshold level and the reading is 228 (the maximum). When the button in the selected row and column is not pushed the capacitor is connected to +5V through a relatively small resistor, giving a POT value of about 2 (this may vary). Since the reading is not critical, the fast pot scan mode can be used, so that only a 2 line wait is required between selecting the row and reading the POT register. The convention has been adopted of comparing the POT reading with 10 (decimal). If is it greater than 10 then the button has been pressed. The third column is read through the joystick trigger line, so it works just like a joystick trigger (0=button is pressed, 1=not pressed).
Light Pen: A light pen is a device that can detect the electron beam as it sweeps across the TV screen. It is used to point directly at an image on the TV display. Applications include selecting menu items and drawing lines. The ATARI 400/800 hardware was designed so that a light pen can be plugged into any of the joystick controller ports (see end of section III).

When any one of the joystick trigger lines (pin 6) is pulled low, the ANTIC chip takes the current VCOUNT value and stores it in PENV. The horizontal color clock value (0-227 decimal) is stored in PENH. The least significant bit is inaccurate and should be ignored. Since there are a number of delays involved in displaying the data and changing the light pen register, each system must be calibrated. Software which uses the light pen should contain a user-interactive calibration routine. For example, the user could point the light pen at a crosshair in the center of the screen and the program could compute the required horizontal offset. PENH will wrap around from 227 to 0 near the right hand edge of a standard width display because of the delay. The pen will not work if it is pointed at a black area of the screen, since the electron beam is turned off. It is a good idea to read two (or more) values and average them, since the user will probably not hold the pen perfectly steady.
III. HARDWARE REGISTERS

This section lists the hardware registers and Operating System (OS) shadow registers.

In the following descriptions, true always refers to a bit whose value is 1.

A. PAL (D014)

<table>
<thead>
<tr>
<th></th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>Not</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>D3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NTSC (US TV)

0 0 0 PAL (European TV)

This byte can be read by a program to determine which type of system the program is running on.

B. INTERRUPT CONTROL

NMIEN (Non Maskable Interrupt Enable)(D40E): This address writes data to the NMI interrupt enable bits.

0 = disabled (masked)
1 = enabled

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>Not</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used</td>
</tr>
<tr>
<td>D7</td>
<td>Display List Instruction Interrupt Enable. This bit is cleared by Power Reset, and may be set or cleared by the processor.</td>
<td></td>
</tr>
<tr>
<td>D6</td>
<td>Vertical Blank Interrupt Enable. This bit is cleared by Power Reset, and may be set or cleared by the processor.</td>
<td></td>
</tr>
</tbody>
</table>

SYSTEM RESET Button Interrupt

This interrupt is always enabled. The SYSTEM RESET button should not be pressed during power turn on.

(SET to hex 40 by OS IRQ code.)
NMIST (Non Maskable Interrupt Status)(D40F): This address read the NMI Status Register (Read by OS NMI code).

0 = no interrupt
1 = interrupt

| D7 | D6 | D5 | Not | Used |

D7 This bit identifies an NMI interrupt caused by bit 7 of a Display List Instruction.

D6 This bit identifies an NMI interrupt caused by the beginning of vertical blank.

D5 This bit identifies an NMI interrupt caused by the SYSTEM RESET button.

NMIRES (NMI Status Register Reset)(D40F): This write address resets the Non Maskable Interrupt Status Register (NMIST).

| Not | Used |

(Written by OS NMI code.)

IRQST (IRQ Interrupt Status)(D20E): This address reads the data from the IRQ Interrupt Status Register.

0 = Interrupt
1 = No Interrupt

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

D7 = 0 Break Key Interrupt
D6 = 0 Other Key Interrupt
D5 = 0 Serial Input Data Ready Interrupt
D4 = 0 Serial Output Data Needed Interrupt
D3 = 0 Serial Output (Byte) Transmission Finished Interrupt *
D2 = 0 Timer 4 Interrupt
D1 = 0 Timer 2 Interrupt
D0 = 0 Timer 1 Interrupt

* - NOTE: Used for generation of 2 stop bits. See IRQ description in section II (no direct reset on bit 3).
**IRQEN (IRQ Interrupt Enable) (D20E):** This address writes data to the IRQ Interrupt Enable bits.

- 0 = disable, corresponding IRQST bit is set to 1
- 1 = enable

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>Break Key Interrupt Enable</td>
<td>D6</td>
<td>Other Key Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D5</td>
<td>Serial Input Data Ready Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>Serial Output Data Needed Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>Serial Out Transmission Finished Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>Timer 4 Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1</td>
<td>Timer 2 Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>Timer 1 Interrupt Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OS SHADOW: POKMSK (hex 10)**

Use AND's and OR's to change one bit in POKMSK without affecting the others. Store the desired value in both IRQEN and POKMSK.

**C. TV LINE CONTROL**

**VCOUNT (Vertical Counter) (D40B):** This address reads the Vertical TV Line Counter (8 most significant bits).

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>V8</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>V2</td>
<td>V1</td>
</tr>
</tbody>
</table>

V0 not read.

Two line resolution supplied.

**WSYNC (Wait for Horizontal Blank Synchronism - i.e. wait until start of next TV line.) (D40A):**

not used

This address sets a latch that pulls down on the RDY line to the microprocessor, causing it to wait until this latch is automatically reset by the beginning of horizontal blank. Display list interrupts may be delayed by 1 line if WSYNC is used. (Used by OS keyboard click routine.)
D. GRAPHICS CONTROL

**DMACTL** (Direct Memory Access Control)(D400): This address writes data into the DMA Control Register.

<table>
<thead>
<tr>
<th></th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **D5 = 1** Enable instruction fetch DMA
- **D4 = 1** 1 line P/M resolution
- **D4 = 0** 2 line P/M resolution
- **D3 = 1** Enable Player DMA
- **D2 = 1** Enable Missile DMA
- **D1, D0 = 0 0** No Playfield DMA
  - **= 0 1** Narrow Playfield DMA (128 Color Clocks)
  - **= 1 0** Standard Playfield DMA (160 Color Clocks)
  - **= 1 1** Wide Playfield DMA (192 Color Clocks)

See **GRACTL**. OS Shadow: **SDMCTL** (22F) default value hex 22

**GRACTL** (Graphics Control)(D01D): This address writes data to the Graphic Control Register.

<table>
<thead>
<tr>
<th></th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Used</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **D2 = 1** Enable latches on TRIGO - TRIG3 inputs (latches are cleared and TRIGO - TRIG3 act as normal inputs when this control bit is zero).
- **D1 = 1** Enable Player DMA to Player Graphics Registers.
- **D0 = 1** Enable Missile DMA to Missile Graphics Registers.

DMA is enabled by setting bits in both DMACTL and GRACTL. Setting DMACTL only will result in cycles being stolen but no display will be generated.
CHACTL (Character Control) (D401): This address writes data into the Character Control Register.

<table>
<thead>
<tr>
<th>Not Used</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

D2  Character Vertical Reflect Bit. This bit is sampled at the beginning of each line of characters. If true it causes the line of characters to reflect (invert) vertically (for upside down characters).

D1  Character Video Invert Flag (used for 40 Character Mode only). If bit 7 of character code is true this flag causes that character to be blue on white (if normal colors are white on blue).

D0  Character Blank (Blink) Flag (used for 40 Character Mode only). If bit 7 of character code is true this flag causes that character to blank. Blinking characters are produced by setting bit 7 of the characters to 1, then periodically changing D0 of CHACTL.

OS SHADOW: CHACT (2F3)

DLISTL (Display List Low) (D402): This address writes data into the low byte of the Display List Counter.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Display List Counter Bit Position.

OS SHADOW: SDLSTL (hex 230)

DLISTH (Display List High) (D403): This address writes data into the high byte of the Display List Counter.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
</tbody>
</table>

Display List Counter Bit Position.

OS SHADOW: SDLSTH (HEX 231)

The Display List is a list of display instructions in memory. These instructions are addressed by the Display List Counter. Loading these registers defines the address of the beginning of the Display List. (See sections I and II.)

Note: The top 6 bits are latches only and have no count capability, therefore the display list can not cross a 1K byte memory boundary unless a jump instruction is used.
DLISTL and DLISTH should be changed only during vertical blank or with DMA disabled. Otherwise, the screen may roll. Bit 7 of NMIEN must be set in order to receive display list interrupts.

CHBASE (Character Address Base Register) (D409): This address writes data into the Character Address Base Register. The data specifies the most significant byte (MSB) of the address of the desired character set (see section II). Note that the last 1 or 2 bits are assumed to be 0.

<table>
<thead>
<tr>
<th>Character Modes</th>
<th>CHBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

OS SHADOW: CHBAS (2F4)

PMBASE (Player-Missile Address Base Register) (D407): This address writes data into the Player-Missile Address Base Register. The data specifies the MSB of the address of the player and missile DMA data (see section II).

<table>
<thead>
<tr>
<th>Two Line Resolution</th>
<th>PMBASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td></td>
<td>Select</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
</tr>
</tbody>
</table>

* = Not Used
HSCROL (Horizontal Scroll Register)(D404): This address writes data into the Horizontal Scroll Register. Only playfield is scrolled, not players and missiles.

| not used | D3 | D2 | D1 | D0 |

0 to 15 color
clock right shifts

The display is shifted to the right by the number of color clocks specified by HSCROL for each display list instruction that contains a 1 in its HSCROL Flag bit (bit 4 of instruction byte).

When horizontal scrolling is enabled, more bytes of data are needed. For a narrow playfield (see DMACTL bits 1 and 0) there should be the same number of bytes per line as for standard playfield with no scrolling. Similarly, for standard playfield use the same number of bytes as for the wide playfield. For wide playfield, there is no change in the number of bytes and background color is shifted in.

VSCROL (Vertical Scroll Register)(D405): This address writes data into the Vertical Scroll Register.

| not used | D2 | D1 | D0 |

8 line display modes

| not used | D3 | D2 | D1 | D0 |

16 line display modes

The display is scrolled upward by the number of lines specified in the VSCROL register for each display list instruction that contains a 1 in its VSCROL Flag bit (bit 5 of instruction byte). The scrolled area will terminate with the first instruction having a zero in bit 5. (see section II for more details).

PRIOR (Priority)(D01B): This address writes data into the Priority Control Register.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

D7-D6 = 0 D5

Multiple Color Player Enable.
This bit causes the logical "or" function of the bits of the colors of Player 0 with Player 1, and also of Player 2 with Player 3. This permits overlapping the position of 2 players with a choice of 3 colors in the overlapped region.

III.7
D4  **Fifth Player Enable.**
This bit causes all missiles to assume the color of Playfield Type 3. (COLPF3). This allows missiles to be positioned together with a common color for use as a fifth player.

D3, D2, D1, & D0  **Priority Select (Mutually Exclusive).**
These bits select one of 4 types of priority. Objects with higher priority will appear to move in front of objects with lower priority.

<table>
<thead>
<tr>
<th>D3=1</th>
<th>D2=1</th>
<th>D1=1</th>
<th>D0=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF0</td>
<td>PF0</td>
<td>PC</td>
<td>P0</td>
</tr>
<tr>
<td>PF1</td>
<td>PF1</td>
<td>P1</td>
<td>P1</td>
</tr>
<tr>
<td>P0</td>
<td>PF2</td>
<td>PF0</td>
<td>P2</td>
</tr>
<tr>
<td>P1</td>
<td>PF3 + P5</td>
<td>PF1</td>
<td>P3</td>
</tr>
<tr>
<td>P2</td>
<td>P0</td>
<td>PF2</td>
<td>PF0</td>
</tr>
<tr>
<td>P3</td>
<td>P1</td>
<td>PF3 + 5</td>
<td>P1</td>
</tr>
<tr>
<td>PF2</td>
<td>P2</td>
<td>P2</td>
<td>PF2</td>
</tr>
<tr>
<td>PF3 + P5</td>
<td>P3</td>
<td>P3</td>
<td>PF3 + 5</td>
</tr>
<tr>
<td>BAK</td>
<td>BAK</td>
<td>BAK</td>
<td>BAK</td>
</tr>
</tbody>
</table>

**NOTE:** The use of Priority bits in a "non-exclusive" mode (more than 1 bit true) will result in objects (whose priorities are in conflict) turning BLACK in the overlap region.

**EXAMPLE:** PRIOR code = 1010  This will black P0 or P1 if they are over PF0 or PF1. It will also black P2 or P3 if they are over PF2 or PF3. In the one-color 40 character modes, the luminance of a pixel in a character is determined by COLPF1, regardless of the priority. If a higher priority player or missile overlaps the character then the color is determined by the player's color.

OS SHADOW: GPRIOR (26F)

**COLPF0 - COLPF3 (Playfield Color)(D016, D017, D018, D019):** These addresses write data to the Playfield Color-Lum Registers.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

(see COLBK for bit assignment)

OS SHADOWS: COLOR0 - 3 (2C4-2C7)
COLBK (Background Color) (D01A): This address writes data to the Background Color-Lum Register.

<table>
<thead>
<tr>
<th>Color</th>
<th>Luminance</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OS SHADOW: COLOR4 (2C8)

E. PLAYERS AND MISSILES

DMACTL, GRACLTL, PMBASE and PRIOR also affect players and missiles.

COLPMO - COLPM3 (Player-Missile Color) (D012, D013, D014, D015): These addresses write to the Player-Missile Color-Lum Registers. Missiles have the same color-lum as their player unless missiles are used as a 5th player (see bit 4 of PRIOR). A 5th player missile gets its color from COLPF3.

OS SHADOWS: PCOLRO - 3 (2C0-2C3)

GRAFPO - GRAFP3 (Player Graphics Registers): (P0, D00D, P1, D00E, P2, D00F, P3, D010): These addresses write data directly into the Player Graphics Registers, independent of DMA. If DMA is enabled then the graphics registers will be loaded automatically from the memory area specified by PMBASE (see page II.3).
**GRAFM (Missile Graphics Registers)(D011):** This address writes data directly into the Missile Graphics Register, independent of DMA.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>R</td>
<td>L</td>
<td>R</td>
<td>L</td>
<td>R</td>
<td>L</td>
<td>R</td>
</tr>
<tr>
<td>M3</td>
<td>M2</td>
<td>M1</td>
<td>MO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SIZEP0 - SIZEP3 (Player Size)(P0 D008, P1 D009, P2 D00A, P3 D00B):**
These addresses write data into the Player Size Control Registers.

<table>
<thead>
<tr>
<th>Not Used</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Normal</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Size</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Normal</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Twice</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Normal</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Size</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Normal</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Size</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>Normal</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Size</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

With normal size objects, each bit in the graphics register corresponds to one color clock. For larger objects, each bit is extended over more than one color clock.

**SIZEM (Missile Size)(D00C):** This address writes data into the Missile Size Control Register.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>DO</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td>M2</td>
<td>M1</td>
<td>MO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Horizontal Size Register (Player)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 Normal Size (8 color clocks wide)</td>
</tr>
<tr>
<td>0 1 Twice Normal Size (16 color clocks wide)</td>
</tr>
<tr>
<td>1 0 Normal Size</td>
</tr>
<tr>
<td>1 1 4 Times Normal Size (32 color clocks wide)</td>
</tr>
</tbody>
</table>

**HPOSPO - HPOSPO (Player Horizontal Position)(P0 D000, P1 D001, P2 D002, P3 D003):** These addresses write data into the Player Horizontal Position Register (see display diagram in section IV). The horizontal position value determines the color clock location of the left edge of the object. Hex 30 is the left edge of a standard width screen. Hex D0 is the right edge of a standard screen.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>DO</th>
</tr>
</thead>
</table>

III.10
HPOS0 - HPOS3 (Missile Horizontal Position) (M0 D004, M1 D005, M2 D006, M3 D007): These addresses write data into the Missile Horizontal Position Registers (see HPOS0 description).

\[
\begin{array}{cccccccc}
D7 & D6 & D5 & D4 & D3 & D2 & D1 & D0 \\
\end{array}
\]

VDELAY (Vertical Delay) (D01C): This address writes data into the Vertical Delay Register.

\[
\begin{array}{cccccccc}
D7 & D6 & D5 & D4 & D3 & D2 & D1 & D0 \\
\end{array}
\]

VDELAY is used to give one-line resolution in the vertical positioning of an object when the 2-line resolution display is enabled. Setting a bit in VDELAY to 1 moves the corresponding object down by one TV line.

If player-missile DMA is enabled then changing the vertical location of an object by more than one line is accomplished by moving bits around in the memory map. If DMA is disabled then the vertical location can be set up by assembly language code which stores data into the graphics registers at the desired line.

MOPF, M1PF, M2PF, M3PF (Missile to Playfield Collisions) (D000, D001, D002, D003): These addresses read Missile to Playfield Collisions. A 1 bit means that a collision has been detected since the last HITCLR.

\[
\begin{array}{cccc}
\text{Not Used (zero forced)} & D3 & D2 & D1 & D0 \\
\end{array}
\]

3 2 1 0 Playfield Type

POP, P1PF, P2PF, P3PF (Player to Playfield Collisions) (D004, D005, D006, D007): These addresses read Player to Playfield Collisions.

\[
\begin{array}{cccc}
\text{Not Used (zero forced)} & D3 & D2 & D1 & D0 \\
\end{array}
\]

3 2 1 0 Playfield Type

MOPL, M1PL, M2PL, M3PL (Missile to Player Collision) (D008, D009, D00A, D00B): These addresses read Missile to Player Collisions.

\[
\begin{array}{cccc}
\text{Not Used (zero forced)} & D3 & D2 & D1 & D0 \\
\end{array}
\]

3 2 1 0 Player Number

POPL, P1PL, P2PL, P3PL (Player to Player Collisions) (D00C, D00D, D00E, D00F): These addresses read Player to Player Collisions.

\[
\begin{array}{cccc}
\text{Not Used (zero forced)} & D3 & D2 & D1 & D0 \\
\end{array}
\]

3 2 1 0 Player Number

(Player 0 against Player 0 is always a zero). Etc.
HITCLR (Collision "HIT" Clear)

This write address clears all collision bits described above.

F. AUDIO

AUDCTL (Audio Control) (D208): This address writes data into the Audio Mode Control Register. (Also see SKCTL two-tone bit 3 and notes).

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>
D7 Change 17 bit poly into a 9 bit below poly.
D6 Clock Channel 1 with 1.79 MHz, instead of 64 KHz.
D5 Clock Channel 3 with 1.79 MHz, instead of 64 KHz.
D4 Clock Channel 2 with Channel 1, instead of 64 KHz (16 BIT).
D3 Clock Channel 4 with Channel 3, instead of 64 KHz (16 BIT).
D2 Insert Hi Pass Filter in Channel 1, clocked by Channel 3.
   (See section II.)
D1 Insert Hi Pass Filter in Channel 2, clocked by Channel 4.
D0 Change Normal 64 KHz frequency, into 15 KHz.

Exact Frequencies: The frequencies given above are approximate. The Exact Frequency (fin) that clocks the divide by N counters is given below (NTSC only, PAL different).

<table>
<thead>
<tr>
<th>FIN (Approximate)</th>
<th>FIN (Exact)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.79 MHZ</td>
<td>1.78979 MHZ</td>
</tr>
<tr>
<td>64 KHz</td>
<td>63.9210 KHz</td>
</tr>
<tr>
<td>15 KHz</td>
<td>15.6999 KHz</td>
</tr>
</tbody>
</table>

Use modified formula for fout

The Normal Formula for output frequency is:

\[ Fout = \frac{Fin}{2N} \]

Where N = The binary number in the frequency register (AUDF), plus 1 (N=AUDF+1). The MODIFIED FORMULA should be used when Fin = 1.79 MHZ and a more exact result is desired:

\[ Fout = \frac{Fin}{2(AUDF + M)} \]

Where:  
M = 4 if 8 bit counter (AUDCTL bit 3 or 4 = 0)  
M = 7 if 16 bit counter (AUDCTL bit 3 or 4 = 1)
AUDF1, AUDF2, AUDF3, AUDF4 (Audio Frequency) (D200, D202, D204, D206)
These addresses write data into each of the four Audio Frequency Control Registers. Each register controls a divide by "N" counter.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
<th>&quot;N&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ETC.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>256</td>
</tr>
</tbody>
</table>

Note: "N" is one greater than the binary number in Audio Frequency Register AUDF(X).

AUDC1, AUDC2, AUDC3, AUDC4 (Audio Channel Control) (D201, D203, D205, D207): These addresses write data into each of the four Audio Control Registers. Each Register controls the noise content and volume of the corresponding Audio Channel.

<table>
<thead>
<tr>
<th>Noise Content or Distortion</th>
<th>Volume</th>
<th>Divisor &quot;N&quot; set by audio frequency register.</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEX</td>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

III.13
### Pitch Values for the Musical Notes — AUDCTL = 0, AUDC = hex AX

<table>
<thead>
<tr>
<th>Octave</th>
<th>Notes</th>
<th>Hex</th>
<th>Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>C</td>
<td>1D</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>1F</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>A or Bb</td>
<td>21</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>23</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>G or Ab</td>
<td>25</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>28</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>F or Gb</td>
<td>2A</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>2D</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>2F</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>D or Eb</td>
<td>32</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>35</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>C or Db</td>
<td>39</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>3C</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>40</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>A or Bb</td>
<td>44</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>48</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>G or Ab</td>
<td>4C</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>51</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>F or Gb</td>
<td>55</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>5B</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>60</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>D or Eb</td>
<td>66</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>6C</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>C or Db</td>
<td>72</td>
<td>114</td>
</tr>
<tr>
<td>MIDDLE C</td>
<td>C</td>
<td>79</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>80</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>A or Bb</td>
<td>88</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>90</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>G or Ab</td>
<td>99</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>A2</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>F or Gb</td>
<td>AD</td>
<td>173</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>B6</td>
<td>182</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>C1</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td>D or Eb</td>
<td>CC</td>
<td>204</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>D9</td>
<td>217</td>
</tr>
<tr>
<td>LOW</td>
<td>C or Db</td>
<td>E6</td>
<td>230</td>
</tr>
<tr>
<td>NOTES</td>
<td>C</td>
<td>F3</td>
<td>243</td>
</tr>
</tbody>
</table>

**STIMER (Start Timer)(D209):** This write address resets all audio frequency dividers to their "AUDF" value. These dividers generate timer interrupts when they count down to zero (if enabled by IRQEN). (also see IRQST)  

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RANDOM (Random Number Generator)(D20A):** This address reads the high order 8 bits of a 17 bit polynomial counter (9 bit, if bit 7 of AUDCTL=1).
G. KEYBOARD AND SPEAKER

CONSOL (Console Switch Port)(D01F): This address reads or writes data from the console switches and indicators. (Set to 8 by OS Vertical Blank code.)

<table>
<thead>
<tr>
<th>Not Used</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>(zero forced)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hex 08 should be written to this address before reading the switches.

Ones written will pull down on the switch line.

CONSOL Bit Assignment:

- D0 Game Start
- D1 Game Select
- D2 Option Select
- D3 Loudspeaker

- 0 means switch pressed.
- should be held at 1 except when writing 0 momentarily. OS writes a 1 during vertical blank.

KBCODE (Keyboard Code)(D209): This address reads the Keyboard Code, and is usually read in response to a Keyboard Interrupt (IRQ and bits 6 or 7 of IRQST). See IRQEN for information on enabling keyboard interrupts. See SKCTL bits 1 and 0 for key scan and debounce enable.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

D7 = Control Key
D6 = Shift Key

Read by OS into shadow CH when key is hit. The OS has a get character function which converts the keycode to ATASCII (Atari ASCII).
### KEYCODE TO ATASCII CONVERSION

<table>
<thead>
<tr>
<th>KEY CODE</th>
<th>KEY CODE</th>
<th>L.C.</th>
<th>U.C.</th>
<th>CTRL</th>
<th>KEY CODE</th>
<th>KEY CODE</th>
<th>L.C.</th>
<th>U.C.</th>
<th>CTRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>L</td>
<td>6C</td>
<td>4C</td>
<td>0C</td>
<td>20</td>
<td>,</td>
<td>2C</td>
<td>5B</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>J</td>
<td>6A</td>
<td>4A</td>
<td>0A</td>
<td>21</td>
<td>SPACE</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>02</td>
<td>;</td>
<td>3B</td>
<td>3A</td>
<td>7B</td>
<td>22</td>
<td>*</td>
<td>2E</td>
<td>5D</td>
<td>60</td>
</tr>
<tr>
<td>03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>23</td>
<td>N</td>
<td>6E</td>
<td>4E</td>
<td>0E</td>
</tr>
<tr>
<td>04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>K</td>
<td>6B</td>
<td>4B</td>
<td>0B</td>
<td>25</td>
<td>M</td>
<td>6D</td>
<td>4D</td>
<td>0D</td>
</tr>
<tr>
<td>06</td>
<td>+</td>
<td>2B</td>
<td>5C</td>
<td>1E</td>
<td>26</td>
<td>/</td>
<td>2F</td>
<td>3F</td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>*</td>
<td>2A</td>
<td>5E</td>
<td>1F</td>
<td>27</td>
<td>A</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>08</td>
<td>0</td>
<td>6F</td>
<td>4F</td>
<td>0F</td>
<td>28</td>
<td>R</td>
<td>72</td>
<td>52</td>
<td>12</td>
</tr>
<tr>
<td>09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>P</td>
<td>70</td>
<td>50</td>
<td>10</td>
<td>2A</td>
<td>E</td>
<td>65</td>
<td>45</td>
<td>05</td>
</tr>
<tr>
<td>0B</td>
<td>U</td>
<td>75</td>
<td>55</td>
<td>15</td>
<td>2B</td>
<td>Y</td>
<td>79</td>
<td>59</td>
<td>19</td>
</tr>
<tr>
<td>0C</td>
<td>RET</td>
<td>9B</td>
<td>9B</td>
<td>9B</td>
<td>2C</td>
<td>TAB</td>
<td>7F</td>
<td>9F</td>
<td>9E</td>
</tr>
<tr>
<td>0D</td>
<td>I</td>
<td>69</td>
<td>49</td>
<td>09</td>
<td>2D</td>
<td>T</td>
<td>74</td>
<td>54</td>
<td>14</td>
</tr>
<tr>
<td>0E</td>
<td>-</td>
<td>2D</td>
<td>5F</td>
<td>1C</td>
<td>2E</td>
<td>W</td>
<td>77</td>
<td>57</td>
<td>17</td>
</tr>
<tr>
<td>0F</td>
<td>=</td>
<td>3D</td>
<td>7C</td>
<td>1D</td>
<td>2F</td>
<td>Q</td>
<td>71</td>
<td>51</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>V</td>
<td>76</td>
<td>56</td>
<td>16</td>
<td>30</td>
<td>9</td>
<td>39</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>63</td>
<td>43</td>
<td>03</td>
<td>32</td>
<td>0</td>
<td>30</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
<td>7</td>
<td>37</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>34</td>
<td>BACKS</td>
<td>7E</td>
<td>9C</td>
<td>FE</td>
</tr>
<tr>
<td>15</td>
<td>B</td>
<td>62</td>
<td>42</td>
<td>02</td>
<td>35</td>
<td>8</td>
<td>38</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>X</td>
<td>78</td>
<td>58</td>
<td>18</td>
<td>36</td>
<td>&lt;</td>
<td>3C</td>
<td>7D</td>
<td>7D</td>
</tr>
<tr>
<td>17</td>
<td>Z</td>
<td>7A</td>
<td>5A</td>
<td>1A</td>
<td>37</td>
<td>&gt;</td>
<td>3E</td>
<td>9D</td>
<td>FF</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>34</td>
<td>24</td>
<td></td>
<td>38</td>
<td>F</td>
<td>66</td>
<td>46</td>
<td>06</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>39</td>
<td>H</td>
<td>68</td>
<td>48</td>
<td>08</td>
</tr>
<tr>
<td>1A</td>
<td>3</td>
<td>33</td>
<td>23</td>
<td></td>
<td>3A</td>
<td>D</td>
<td>64</td>
<td>44</td>
<td>04</td>
</tr>
<tr>
<td>1B</td>
<td>6</td>
<td>36</td>
<td>26</td>
<td></td>
<td>3B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1C</td>
<td>ESC</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>3C</td>
<td>CAPS</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>1D</td>
<td>5</td>
<td>35</td>
<td>25</td>
<td></td>
<td>3D</td>
<td>G</td>
<td>67</td>
<td>47</td>
<td>07</td>
</tr>
<tr>
<td>1E</td>
<td>2</td>
<td>32</td>
<td>22</td>
<td></td>
<td>3E</td>
<td>S</td>
<td>73</td>
<td>53</td>
<td>13</td>
</tr>
<tr>
<td>1F</td>
<td>1</td>
<td>31</td>
<td>21</td>
<td></td>
<td>3F</td>
<td>A</td>
<td>61</td>
<td>41</td>
<td>01</td>
</tr>
</tbody>
</table>

* = special handling
H. SERIAL PORT (see peripheral connector on console)

SKCTL (Serial Port control)(D20F): This address writes data into the register that controls the configuration of the serial port, and also the Fast Pot Scan and Keyboard Enable.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

(Bits are normally zero and perform the functions shown below when true.)

- D7 Force Break (force serial output to zero (space))*
- D6 Serial Port Mode Control (see mode chart at end of Serial port description, page II.34).
- D5 Two Tone (Serial output transmitted as two tone signal instead of logic true/false.)
- D4 Fast Pot (Fast Pot Scan. The Pot Scan Counter completes its sequence in two TV line times instead of one frame time. The capacitor dump transistors are completely disabled.)
- D3 Enable Key Scan (Enables Keyboard Scanning circuit)
- D2 Enable Debounce (Enables Keyboard Debounce circuits)
- D0-D1 (Both Zero) Initialize (State used for testing and initializing chip) **

OS SHADOW: SSKCTL (hex 232)

The OS enables key scan and debounce and may change the other bits for different I/O operations. In particular, an aborted cassette operation may leave the two tone bit in the true state, causing undesirable audio signals. This may be corrected by writing hex 13 to both SKCTL and SSKCTL after doing I/O and/or before modifying the audio registers.

* NOTE: When powered on, serial port output may stay low even if this bit is cleared. To get S.P. high (mark), send a byte out (recommend 00 or FF).

**NOTE: There is no original power on state. Pokey has no reset pin.
SKSTAT (Serial Port-Keyboard Status)(D20F): This address reads the status register giving information about the serial port and keyboard.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

(Bits are normally true and provide the following information when zero.)

- D7 = 0 = Serial Data Input Frame Error
- D6 = 0 = Serial Data Input Over-run
- D5 = 0 = Keyboard Over-run
- D4 = 0 = Direct from Serial Input Port
- D3 = 0 = Shift Key Depressed
- D2 = 0 = Last Key is Still Depressed
- D1 = 0 = Serial Input Shift Register Busy
- D0 = 1 Not Used (Logic True)

SKRES (Reset above Status Register)(D20A): This write address resets bits 7, 6, and 5 of the Serial Port-Keyboard Status Register to 1.

not used

SERIN (Serial Input Data)(D20D): This address reads the 8 bit parallel holding register that is loaded when a full byte of serial input data has been received. This address is usually read in response to a serial data in interrupt (IRQ and bit 5 of IRQST). Also see IRQEN.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Serial I/O Port Connector Pinout:

1. Clock In
2. Clock Out
3. Data In to computer
4. GND
5. Data Out of Computer
6. GND
7. Command
8. Motor Control
9. Proceed
10. +5 / Ready
11. Audio In
12. +12
13. Interrupt

See serial port description in OS manual for more details.
SEROUT (Serial Output Data) (D20D): This address writes to the 8 bit parallel holding register that is transferred to the output serial shift register when a full byte of serial output data has been transmitted. This address is usually written in response to a serial data out interrupt (IRQ and bit 4 of IRQST).

```
D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0
```

I. CONTROLLER PORTS (front of console)

PORTA (Port A) (D300): This address reads or writes data from Player 0 and Player 1 controller jacks if bit 2 of PACTL is true. This address writes to the direction control register if bit 2 of PACTL is zero. I/O for both ports (A and B) goes through a 6520/6820 Data Register-Addressed if bit 2 of PACTL is 1.

**Joystick Operation**

```
D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0
```

<table>
<thead>
<tr>
<th>Right</th>
<th>Back</th>
<th>Right</th>
<th>Back</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>Fwd.</td>
<td>Left</td>
<td>Fwd.</td>
</tr>
</tbody>
</table>

Stick1 (Jack 2) Stick0 (Jack 1)

**Paddle Operation**

```
D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0
```

| PTRIG2 | PTRIGO | PTRIG3 | PTRIG1 |

**Keyboard Controller Operation**

```
D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0
```

<table>
<thead>
<tr>
<th>Top Row</th>
<th>2nd Row</th>
<th>3rd Row</th>
<th>4th Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Row</td>
<td>2nd Row</td>
<td>3rd Row</td>
<td>4th Row</td>
</tr>
<tr>
<td>Jack 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Direction Control Register-Addressed if bit 2 of PBCTL is 0

```
D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0
```

Each bit corresponds to a jack pin

0=input
1=output

OS SHADOWS: STICK0 (hex 278), STICK1 (279), PTRIGO-3 (27C-27F)
PACTL (Port A Control) (D302): This address writes or reads data from the Port A Control Register.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

Port A Control Register
Set up register as shown
(X = described below)

D7 - (Read only) Peripheral A Interrupt Status Bit. Serial bus Proceed line. (Reset by reading Port A Register. Set by Peripheral A Interrupt.)

D3 - Peripheral Motor Control line on serial bus (write).
(0 = On 1 = Off)

D2 - Controls Port A addressing described above (write).
(1 = Port A Register 0 = Direction Control Register).

D0 - Peripheral A Interrupt Enable Bit. (Write) 1 = Enable.
Reset by power turn-on or processor. Set by Processor.

PORTB (Port B) (D301): This address reads or writes data from Player 2 and Player 3 controller jacks if bit 2 of PBCTL is true. This address writes to the direction control register if bit 2 of PBCTL is zero. I/O for both ports (A and B) goes through a 6520/6820.

Data Register-Addressed if bit 2 of PBCTL is 1

Joystick Operation

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Right</td>
<td>Back</td>
<td>Right</td>
<td>Back</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Left</td>
<td>Fwd.</td>
<td>Left</td>
<td>Fwd.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stick3
(Jack 4)

Stick2
(Jack 3)

Paddle Operation

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTRIG6</td>
<td>PTRIG7</td>
<td>PTRIG4</td>
<td>PTRIG5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Keyboard Controller Operation

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Row</td>
<td>2nd Row</td>
<td>3rd Row</td>
<td>4th Row</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top Row</td>
<td>2nd Row</td>
<td>3rd Row</td>
<td>4th Row</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Jack 3

Jack 4
Direction Control Register—Addressed if bit 2 of PBCTL is 0

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

Each bit corresponds to a jack pin

0 = input
1 = output

**OS SHADOWS:** STICK2 (hex 27A), STICK3 (27B), PTRIG4-7 (280-283)

PBCTL (Port B Control)(D303): This address writes or reads data from the Port B Control Register.

<table>
<thead>
<tr>
<th>Read Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
</tr>
<tr>
<td>x</td>
</tr>
</tbody>
</table>

Port B Control Register
Set up register as shown (X = Described below)

**D7** (Read only) Peripheral B Interrupt Status Bit. Serial bus Interrupt line. Reset by Reading Port B Register. Set by Peripheral B Interrupt.

**D3** Peripheral Command Identification. Serial bus Command Line.

**D2** Controls Port B addressing described above.
(1 = Port B Register 0 = Direction Control Register)

**D0** Peripheral B Interrupt Enable Bit. 1 = Enable.
Reset by power turn-on or processor. Set by processor.
(Set to hex 3C by OS IRQ code)

**POTO - POT7 (Pot Values)**(D200-D207): These addresses read the value (0 to 228) of 8 pots (paddle controllers) connected to the 8 lines pot port. The paddle controllers are numbered from left to right when facing the console keyboard. Turning the paddle knob clockwise results in decreasing pot values. The values are valid only after 228 TV lines following the "POTGO" command described below or after ALLPOT changes.

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

Each Pot Value (0-228)

**OS SHADOWS:** PADDLO - 7 (hex 270-277)
ALLPOT (All Pot Lines Simultaneously)(D208): This address reads the present state of the 8 line pot port.

Capacitor dump transistors must be turned off by either going to fast pot scan mode (bit 2 of SKCTL) or starting pot scan (POTGO).

<table>
<thead>
<tr>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
</table>

Pot number: 7 6 5 4 3 2 1 0
0 = Pot register value is valid.
1 = Pot register value is not valid.

8 Pot Line States

POTGO (Start Pot Scan)(D20B):

No Data Bits Used

This write address starts the pot scan sequence. The pot values (POTO - POT7) should be read first. This write strobe is then used causing the following sequence.

1. Scan Counter cleared to zero.
2. Capacitor dump transistors turned off.
3. Scan Counter begins counting.
4. Counter value captured in each of 8 registers (POTO - POT7) as each pot line crosses trigger voltage.
5. Counter reaches 228, capacitor dump transistors turned on.

(Written to by OS vertical blank code)

TRIGO, TRIG1, TRIG2, TRIG3 (Trigger Ports)(0 D010, 1 D011, 2 D012, 3 D013): These addresses read port pins normally connected to the joystick controller trigger buttons.

<table>
<thead>
<tr>
<th>Not Used</th>
<th>(Zero Forced)</th>
<th>D0</th>
</tr>
</thead>
</table>

0 = button pressed
1 = button not pressed

OS SHADOWS: STRIGO-3 (hex 284-287)

NOTE: TRIGO thru TRIG3 are normally read directly by the microprocessor. However, if bit 2 of GRACLTL is 1, these inputs are latched whenever they go to logic zero. These latches are reset (true) when bit 2 of GRACLTL is set to 0.
PENH (Light Pen Horizontal Color Clock Position) (D40C): This address reads the Horizontal Light Pen Register (based on the horizontal color clock counter in hardware). The values range from 0 to decimal 227. Wraparound occurs when the pen is near the right edge of a standard-width screen. PENH and PENV are modified when any of the joystick trigger lines is pulled low.

H7 H6 H5 H4 H3 H2 H1 H0

OS SHADOW: LPENH (hex 234)

PENV (Light Pen Vertical TV Line Position) (D40D): This address reads the Vertical Light Pen Register (8 most significant bits, same as VCOUNT).

LP8 7 6 5 4 3 2 1 0 LP0 not read. Two line resolution supplied.

OS SHADOW: LPENV (hex 235)

Front Panel (Controller) Jacks as I/O Parts:

PIA (6520/6820)
Out: TTL levels, l load
In: TTL levels, l load

---

Port A Circuit (typical):

Port 6520 (A) 220 .001 Jack

---

Port B Circuit (typical):

Port 6520 (B) 4.7K .001 Jack

---

"Trigger" Port Circuit (typical):

CTIA Trig 220 .001 Jack
### Controller Port Pinout:

#### Male (console)

<table>
<thead>
<tr>
<th>PIN</th>
<th>JOYSTICK</th>
<th>PADDLE (POT)</th>
<th>KEYBOARD</th>
<th>HARDWARE REGISTERS</th>
<th>OS VARIABLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Forward</td>
<td>Top Row*</td>
<td>Bit 0 or 4**</td>
<td>Bit 0***</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Back</td>
<td>2nd Row*</td>
<td>Bit 1 or 5**</td>
<td>Bit 1***</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Left</td>
<td>3rd Row*</td>
<td>Bit 2 or 6**</td>
<td>PTRIGO,2,4,6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Right</td>
<td>Bottom Row*</td>
<td>Bit 3 or 7**</td>
<td>PTRIG1,3,5,7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Trigger Button</td>
<td>1st Column</td>
<td>POT 1,3,5,7</td>
<td>Bit 3***</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>POT B(Right)</td>
<td>3rd Column</td>
<td>TRIGO,1,2,3</td>
<td>PADDL1,3,5,7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>+5</td>
<td>+5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>GND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>POTA (Left)</td>
<td>2nd Column</td>
<td>POT 0,2,4,6</td>
<td>PADDL0,2,4,6</td>
<td></td>
</tr>
</tbody>
</table>

#### Female (connector)

- * Write
- ** PORTA or PORTB
- *** STICK 0, 1, 2 or 3
IV. SAMPLE PROGRAM

This assembly language program illustrates the use of players, missiles, and display lists. The diagram on the next page shows what the display looks like and which objects are used. The comments in the program listing describe how it works.
CHECKERS DISPLAY

Player/Missile Horizontal Position Register Values (hex).

TV LINES AFTER START OF DISPLAY

End of Vertical Blank

24 Blank Lines

20 Characters across by 8 lines high

16 Characters across by 16 lines high

8 Rows

16 Characters across by 16 lines high

16 Characters across by 8 lines high
TITLE "ATARI 80's CHECKERS DISPLAY NYX - HAN 2/31/80"

; COPYRIGHT ATARI 1980

; THIS IS AN EXAMPLE OF A DISPLAY LIST WHICH USES CHARACTER MAPPING TO
; PRODUCE THE CHECKERS AND THE TOP AND BOTTOM BORDERS OF THE BOARD.
; PLAYERS ARE USED FOR THE RED SQUARES. THIS GIVES 6 COLORS WITHOUT
; CHANGING THE COLOR REGISTERS.
; MISSILES ARE USED FOR THE LEFT AND RIGHT FORDER.
; THE PROGRAM STARTS AT THE LOCATION SPECIFIED BY PME.
; A FEW TRICKS ARE USED TO SAVE RAM, BUT FURTHER OPTIMIZATION IS POSSIBLE.
; THIS IS A RAM BASED PROGRAM WHICH RUNS WITH THE ASSEMBLER CARTRIDGE, NOT A
; ROM CARTRIDGE.
; COLLEEN (ATARI 800) EQUATES

D409
0170 CHEASE = $D409
D400
0100 DMACLT = $D400
022F
0190 SMMCLT = $D400
D400
0200 HPOSP0 = $0000
D400
0210 SIZEP0 = $0008
D400
0220 PCOLR0 = $02C0
D400
0230 SDLSTL = $0230
D400
0240 SDLSTH = $0231
D400
0250 GRNCLT = $001D
D407
0260 PMBASE = $0407
026F
0270 GPRIGR = $026F
D400
0280 VDSLST = $0200
D40E
0290 NMIEN = $040E
D300
; DISPLAY LIST EQUATES

D320
0000 INT = $00 ; DISPLAY LIST INTERRUPT (BIT 7 OF NMI STATUS)
0041 JMPWT = $41 ; JUMP AND WAIT UNTIL END OF NEXT VERTICAL BLANK (2 BYTES)
0040 JUMP = $40 ; RELOAD MEN SCAN COUNTER (3 BYTES)
0020 VSC = $20 ; VERTICAL SCROLL ENABLE
0010 MSC = $10 ; HORIZONTAL SCROLL ENABLE
0001 JUMP = $1 ; JUMP INSTRUCTION (3 BYTES)
0000 BLANK1 = $0 ; 1 BLANK TV LINE
0010 BLANK2 = $10 ; 2 BLANK LINES
0020 BLANK3 = $20 ; 3 BLANK LINES
0030 BLANK4 = $30 ; 4 BLANK LINES
0040 BLANK5 = $40 ; 5 BLANK LINES
0050 BLANK6 = $50 ; 6 BLANK LINES
0060 BLANK7 = $60 ; 7 BLANK LINES
0070 BLANK8 = $70 ; 8 BLANK TV LINES
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

0000 0470   PAGE
0020 0490 INTOFF = $20  USED TO GET INTERNAL CODE FOR UPPER CASE ALPHANUMERICS
          0500 ;
          0510 /INTERNAL CHARACTER CODES
          0520 ;
0000 0530 SPI = 'I-INTOFF
0021 0540 AI = '!A-INTOFF
0023 0550 CI = '!C-INTOFF
0024 0560 DI = '!O-INTOFF
0025 0570 EI = '!E-INTOFF
0027 0580 GI = '!O-INTOFF
0028 0590 HI = '!H-INTOFF
0029 0600 II = '!I-INTOFF
002F 0610 DI = '!D-INTOFF
0030 0620 PI = '!P-INTOFF
0032 0630 RI = '!R-INTOFF
0034 0640 TI = '!T-ENTOFF
0039 0650 VI = '!V-INTOFF
003A 0660 NI = '!N-INTOFF
003B 0670 NB1 = '!B-INTOFF
0039 0680 NB2 = '!B-INTOFF
0010 0690 NO1 = '!O-INTOFF
0700 ;
0710 ;CHECKERS EQUATES
0720 ;
0730 ;CODES FOR SPECIAL CHECKERS CHARACTER SET
0740 ;
0000 0750 EMPT = 0  EMPTY SQUARE
0001 0760 CHECKER= 1  ORDINARY CHECKER
0002 0770 KING = 2  
0003 0780 CUS = 3  CURSOR (X)
0004 0790 BORDER = 4  USED FOR TOP AND BOTTOM BORDERS OF BOARD
0008 0810 CLPO = 0  PLAYER 0 (HUMAN)
0009 0820 CPL = 90  PLAYER 1 (COMPUTER)
000C 0830 LDR = $C0  BORDER COLOR USED TO SET UP 2 MSB'S OF CHAR
5000 0840 PMB = $5000  PLAYER MISSILE BASE ADDRESS & PROGRAM LOCATION
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

0000  0050  PAGE
  0060  ; PAM VARIABLES
  0070  ;
  0080  0590  == PMB
  5000  0900 BOARD  ==  **32  ; CHECKER BOARD (ONLY 32 BLACK SQUARES ARE USED)
  5020  0910 T0  ==  **1  ; TEMP FOR MOVING BOARD TO MEM MAP
  0920  ;
  0930  ; PLAYER AND MISSILE GRAPHS
  0940  ; PLAYERS ARE USED FOR SQUARES, MISSILES FOR LEFT AND RIGHT BORDERS

5021  0960  == PMB+$180
5100  0970 GRP0  ==  **$80  ; MISSILE GRAPHICS
5200  0980 GRP1  ==  **$80  ; PLAYER 0 GRAPHICS
5300  0990 GRP2  ==  **$80  ; PLAYER 1
5300  1000 GRP2  ==  **$80  ; 2
5300  1010 GRP3  ==  **$80  ; 3
5400  1020 TITL  ==  **20  ; TOP LINE OF CHARS — ASCII MESSAGE
5414  1040 TOPBRD  ==  **16  ; TOP BORDER OF BOARD
5424  1050 BRDSP  ==  $16++  ; BOARD DISPLAY
5444  1060 BOTBRD  ==  **16  ; BOTTOM BORDER
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

5484  1870  .PAGE
      1880  .
      1890  ;G# -- SPECIAL CHECKERS CHARACTER SET (ONLY CODES 0-4 ARE USED).
      1900  .
      5484  1110  == PBS=#600
      1120 GR
      5600  00  1130  .BYTE 0.0.0.0.0.0.0.0 ; BLANK (0)
      5601  00  
      5602  00  
      5603  00  
      5604  00  
      5605  00  
      5606  00  
      5607  00  
      5608  3C  1140  .BYTE $3C,$7E,$FF,$FF,$FF,$7E,$3C ; CHECKER (1)
      5609  7E  
      560A FF  
      560B FF  
      560C FF  
      560D FF  
      560E 7E  
      560F 3C  
      5610 2C  1150  .BYTE $3C,$7E,$A5,$A5,$C3,$C3,$7E,$3C ; KING (2)
      5611  7E  
      5612 A5  
      5613 A5  
      5614 C3  
      5615 C3  
      5616 7E  
      5617 3C  
      5618 C3  1160  .BYTE $C3,$66,$3C,$18,$18,$3C,$66,$C2 ; CURSOR (3)
      5619  66  
      561A 3C  
      561B 18  
      561C 18  
      561D 3C  
      561E 66  
      561F C3  
      5620  00  1170  .BYTE 0,$FF,$FF,$FF,$FF,$FF,0 ; BORDER (4)
      5621  FF  
      5622 FF  
      5623 FF  
      5624 FF  
      5625 FF  
      5626 FF  
      5627 00  

PAGE SPECIAL CHARACTERS (0-4 USE ONLY).

BYTE $3C,$7E,$FF,$FF,$7E,$3C ; CHECKER (1)
BYTE $3C,$7E,$A5,$A5,$C3,$C3,$7E,$3C ; KING (2)
BYTE $C3,$66,$3C,$18,$18,$3C,$66,$C2 ; CURSOR (3)
BYTE 0,$FF,$FF,$FF,$FF,0 ; BORDER (4)
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/80

5628 1180 PAGE
1180 :
1200 :
1210 : DISPLAY LIST
1220 :
1230 DSP

5628 70 1240 .BYTE BLANK8, 24 BLANK LINES
5629 70 1250 .BYTE BLANK8
5630 70 1260 .BYTE BLANK8
5631 46 1270 .BYTE RELOAD+6, LINES 0-7. MESSAGE LINE: 20 ACROSS X 5 COLOR X 1 LINE RESOLUTION CHARACTERS
5632 0054 1280 .WORD TITL
5633 80 1290 .BYTE INT+BLANK1, 8 INTERRUPT TO CHANGE CHARACTER BASE ADDRESS AND CHANGE TO NARROW SCREEN
5634 86 1300 .BYTE 6, .9-16. TOP BORDER. 16 X 5 X 1 CHAR. LAST LINE IS TOP OF 1ST ROW OF SQUARES.
5635 10 1310 .BYTE BLANK2, 17-18. TOP OF FIRST ROW OF SQUARES
5636 07 1320 .BYTE 7, CHECKERBOARD (8 LINES OF CHAR WITH SPACES INBETWEEN - 22 LINES/SQUARE)
5637 10 1330 .BYTE 7, 19-24. 16X5X2 LINE RESOLUTION CHAR.
5638 50 1340 .BYTE BLANK6, 25-30. FIRST 3 LINES=BOTTOM OF PREVIOUS SQUARE.
5639 50 1350 .BYTE BLANK6, 31-36. LAST 3 LINES=TOP OF NEXT SQUARE.
5640 50 1360 .BYTE BLANK6, 37-42. NEXT THREE LINES ARE BOTTOM OF PREVIOUS SQUARE.
5641 50 1370 .BYTE BLANK2, 43-48. END OF NORMAL DISPLAY (SHOULD BE ON SCREEN ON ALL TV'S).
5642 50 1380 .BYTE BLANK2, 49-54. BOTTOM BORDER (MAY OVERSCAN, BUT NOT ESSENTIAL TO GAME PLAY)
5643 50 1390 .BYTE BLANK2, 55-60. WAIT FOR NEXT VBLANK, THEN START OVER.
5644 50 1400 .WORD DSP
5645 50 1410 NCHR
5646 4956 1420 LDA #$2D
5647 8009D4 1430 STA CHBASE
5648 1620 PLA
5649 1630 JMPWT
564A 1550 .DSP -- DISPLAY LIST INTERRUPT HANDLER.
564B 1560 .CHANGES CHARACTER BASE AND WIDTH OF DISPLAY FOR SPECIAL CHECKERS GRAPHICS.
564C 1570 .THE OS WILL CHANGE CHBASE BACK TO NORMAL DURING VERTICAL BLANK.
564D 1580 NCHR
564E 48 1600 PHA
564F 4956 1610 LDA #GR/256
5650 8009D4 1620 STA CHBASE
5651 1630 :
5652 41 1640 .INSTRUCTION FETCH DMA ENABLE, P/M 2 LINE RES, P/M DMA ENABLE, NARROW SCREEN (128 CLOCKS)
ATARI 600 CHECKERS DISPLAY BY C. SHAW 3/31/80

5652 1600 PAGE
1700 ;
1710 ;INITIALIZATION CODE -- START EXECUTION HERE
1720 ;
5652 1730 ** PMB+1700
1740 ;
1750 ;INIT OS'S DMACTL VARIABLE
1760 ;INSTRUCTION FETCH DMA ENABLE, P/M 2 LINE RES; P/M DMA ENABLE, STANDARD SCREEN (160 CLOCKS)
1770 ;
5700 A92E 1780 LDA #2E
5702 802F02 1790 STA SOMCTL
1800 ;
1810 ;CLEAR RAM
1820 ;
5705 A900 1830 LDA #0
5707 A9 1840 TRX
1850 INITLP
5709 900050 1860 STA PMB.X
570B 900051 1870 STA PMB+100.X
570E 900052 1880 STA PMB+200.X
5711 900053 1890 STA PMB+300.X
5714 900054 1900 STA PMB+400.X
5717 E8 1910 INX
5718 D0EE 1920 BNE INITLP
1930 ;
1940 ;INITIALIZE MISSILE GRAPHICS FOR BORDERS
1950 ;
571A A90E 1960 LDA #0E
571C A90E 1970 LDY #0E
571E 999451 1980 COPZ STA GRP0+$14.Y
5721 80 1990 DEV
5722 D0FA 2000 BNE LGPZ
2010 ;
2020 ;INITIALIZE TOP AND BOTTOM BORDERS
2030 ;
5724 A010 2040 LDY #16
5726 A9C4 2050 LDA #CLBOR+BORDER
5728 991354 2060 TBLP STA TOPBD+$1.Y
572B 999354 2070 STA BOTBD+$1.Y
572E 80 2080 DEV
572F D0F7 2090 BNE TBLP ;CONTINUE UNTIL Y=0
2100 ;
2110 ;INITIALIZE PLAYER GRAPHICS FOR SQUARES (CHECKER BOARD) Y=0
2120 ;
5731 A9F0 2130 LDA #$F0
5733 A20A 2140 IN2 LDX #10
5735 991852 2150 IN3 STA GRP0+$16.Y
5738 991852 2160 STA GRP1+$16.Y
573B 991853 2170 STA GRP2+$16.Y
573E 999653 2180 STA GRP3+$16.Y
2190 ;
$ATARI 800 CHECKERS DISPLAY BY C. SHAW 2/31/80$

5741 48 2200PHA
5742 9A0A 2210LDA #38A
5744 999951 2220STA GRAM+18,Y; REST OF MISSILE GRAPHICS
5747 68 2230PLA
5748 CB 2240INV
5749 CA 2250DEX
574A 10E9 2260BPL IN3
574C 49FF 2270EDR #FF ;FILL IN OPPOSITE SQUARES
574E 9098 2280CPY #8
5750 90E1 2290BCC IN2
5752 A008 2300LDV #8

;INITIALIZE PLAYER AND MISSILE POSITIONS AND COLORS
2310 ;
5754 B90857 2348 IN4 LDA ITBL,Y
5757 990000 2350STA HP0SP0,Y
5759 BA 2360TXA ;#FF
575B 990000 2370STA SIZEP0,Y ;#93 INDICATES 4 TIMES NORMAL SIZE (REST IS DON'T CARE)
575E 990057 2380LDA ITBL1,Y
5761 990002 2390STA PCOLR0,Y
5764 98 2400DEV
5765 10ED 2410BPL IN4

;OS, ANTIC, POKEY INITIALIZATION
2420 ;
5767 A928 2450LDA #DP4,#FF ;DISPLAY LIST START ADDRESS (LSB)
5769 BD3002 2460STA SIDS
576C A956 2470LDA #DSP/256,MSB OF ADDRESS
576E BD3102 2480STA SIDSH
5771 A903 2490LDA #3 ;ENABLE PLAYER/MISSILE DMA TO GRAPHICS REGS.
5773 BD1000 2500STA GRACTL
5776 A950 2510LDA #PMB/256,MSB OF ADDRESS OF PLAYER/MISSILE GRAPHICS
5778 BD0704 2520STA PMBASE
577B A914 2530LDA #1145TH PLAYER ENABLE (USE PF3 FOR MISSILE COLOR), PF TAKES PRIOR OVER PLAYERS
577D 06F802 2540STA GPRIO ;OS PRIORITY REG
577F A945 2550LDA #CHR,#FF ;DISPLAY LIST INTERRUPT VECTOR (LSB)
5780 BD0002 2560STA VDSLST
5785 A956 2570LDA #CHR/256
5787 BD0102 2580STA VDSLST+1
578A BE0E04 2590STX NHIEF,X=#00 #C0 ENABLES DISPLAY LIST & VBLANK INTERRUPTS

2600 ;INITIALIZE BOARD DISPLAY
2610 ;LDX #11
579D A20B 2620BRLP
579F A901 2630LDA #CHECKER+CLP0,HUMAN PIECES ON SQUARES 0-11
57A1 909950 2640STA BPMPD,X
57A4 A901 2650LDA #CHECKER+CLP1,COMPUTER PIECES ON SQUARES 20-31
57A6 901450 2660STA BPMPD+20,X
57A9 CA 2670DEX
579A 10F3 2680BPL BRLP
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/21/80

2720 .MOVE COPYRIGHT MESSAGE TO MESSAGE DISPLAY LINE
2730 ,
573C A213 2740 LDX #14
573E 80E957 2750 IN6 LDA COPY.X
5741 900054 2760 STA TITLE.X
5744 CA 2770 DEX
5745 10F7 2780 BPL IN6
2790 ;
2800 ; LOOP TO MOVE BOARD TO GRAPHICS AREA.
2810 ; THE CHECKERS PROGRAM LOGIC COULD BE ADDED HERE OR A VBLANK INTERRUPT COULD BE USED.
2820 ;
2830 LOOP
5747 20A957 2840 JSR UPCHR
574A 4C7B57 2850 JMP LOOP
2860 ;
2870 ;
2880 ;
2890 ;
2900 ; UPCHR -- SUBROUTINE TO MOVE 32 BYTES OF CHECKER BOARD TO DISPLAY RAM
2910 ;
2920 UPCHR
574D A21F 2930 LDX #31 ; SQUARE 31 = UPPER LEFT
574F A000 2940 LDY #0
2950 UPLP1
5751 A903 2960 LDA #4-1 ; 4 SQUARES/LINE
5753 8D0050 2970 STA T0
2980 UPLP2
5756 8D0050 2990 LDA BOARD.X
5759 992654 3000 STA BRDSP+2.Y ; FOR ROWS SHIFTED TO RIGHT
575C BDFF4F 3010 LDA BOARD-4.X
5760 993454 3020 STA BRDSP+16.Y ; FOR ROWS SHIFTED TO LEFT
5763 CB 3030 INV
5765 CB 3040 INV
5767 CB 3050 INV
5769 CB 3060 INV
576B CA 3070 DEX
576D CE0250 3080 DEC T0
576F 10EA 3090 BPL UPLP2
3100 ;
5770 96 7110 TYA
5772 16 3120 CLC
5774 6910 3130 ADC #10
5776 AB 3140 TYR
5778 1A 3150 TXA
577A E903 3160 SBC #4-1 ; CARRY IS CLEAR (SUBLTACT 4)
577C AA 3170 TRX
577E 800A 3180 BCS UPLP1
5780 60 3190 RTS
3200 ;
3210 ;
3220 ;
3230 ;
ATARI 800 CHECKERS DISPLAY BY C. SHAW 3/31/81

.3240 ;DATA
.3250 ;HORIZONTAL POSITION OF PLAYERS (SQUARES) AND MISSILES (SIDE BORDERS).
.3260 ;M0=RIGHT CORNER, M1=LEFT CORNER
.3270 ;M2 & M3 ARE PLACED WITH M1
.3280 ;F0, F1, F2, F3, M0, M1, M2, M3

5700 3C 3300 .BYTE $3C,$5C,$7C,$9C,$BC,$38,$38,$38
5709 5C
570A 7C
570B 9C
570C BC
570D 38
570E 38
570F 38

3310 ;
3320 ;COLOR TABLE
3330 ITBL1

57E0 34 3340 .BYTE $34,$34,$34,$34 ;4 PLAYERS (RED SQUARES)
57E1 34
57E2 34
57E3 34
57E4 36
57E5 88 3350 .BYTE $36 ;PF0: RED CHECKERS AND MESSAGES
57E6 0E 3360 .BYTE $88 ;PF1: BLUE CHARACTERS
57E7 26 3370 .BYTE $0E ;PF2: WHITE CHECKERS AND MESSAGES
57E8 00 3380 .BYTE $26 ;PF3: YELLOW BORDER (CHARS & MISSILES)
57E9 00 3390 .BYTE 0 ;BK: BLACK BACKGROUND
3400 ;
3410 ;"COPYRIGHT ATARI 1980" MESSAGE
3420 ;
0000 3430 OF = $00 ;FOR PF0 COLOR (RED)
0000 3440 OF2 = $00 ;FOR PF2 COLOR (WHITE)
0040 3450 OF3 = $40 ;FOR PF1 COLOR (BLUE)
3460 TOTBL

57E9 00 3470 COPY .BYTE SP1,CI+OF,DI+OF,P1+OF,V1+OF,R1+OF,II+OF,G1+OF,H1+OF,T1+OF
57EA 23
57EB 2F
57EC 39
57ED 39
57EE 32
57EF 29
57F0 27
57F1 28
57F2 34
57F3 41 3480 .BYTE A1+OF2,T1+OF2,A1+OF2,R1+OF2,II+OF2,N1+OF3,N91+OF3,N81+OF3,N81+OF3
V. HARDWARE REGISTER LISTS

A. ADDRESS ORDER

<table>
<thead>
<tr>
<th>Address</th>
<th>Write</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0FF</td>
<td>CONSO</td>
<td>CONSO</td>
</tr>
<tr>
<td>D020</td>
<td>REPEA AS BELOW</td>
<td>7 MORE TIMES</td>
</tr>
<tr>
<td>D01F</td>
<td>CONSOL Write Consol SW. Port</td>
<td>CONSOL Read Consol SW. Port</td>
</tr>
<tr>
<td>D01E</td>
<td>HITCLR Collision Clear</td>
<td></td>
</tr>
<tr>
<td>D01D</td>
<td>GRACTL Graphic Control</td>
<td></td>
</tr>
<tr>
<td>D01C</td>
<td>VDELAY Vert. Delay</td>
<td></td>
</tr>
<tr>
<td>D91B</td>
<td>PRIOR Priority Select</td>
<td></td>
</tr>
<tr>
<td>D01A</td>
<td>COLBK Column Bkgnd</td>
<td></td>
</tr>
<tr>
<td>D019</td>
<td>COLP3 Color-lum of 3</td>
<td></td>
</tr>
<tr>
<td>D018</td>
<td>COLP2 Playfield 2</td>
<td></td>
</tr>
<tr>
<td>D017</td>
<td>COLP1 Playfield 1</td>
<td></td>
</tr>
<tr>
<td>D016</td>
<td>COLP0 Playfield 0</td>
<td></td>
</tr>
<tr>
<td>D015</td>
<td>COLP Color-lum of 3</td>
<td></td>
</tr>
<tr>
<td>D014</td>
<td>COLPM2 Player-Missile 2</td>
<td>PAL READ PAL/NTSC bits</td>
</tr>
<tr>
<td>D013</td>
<td>COLPM1 Player-Missile 1</td>
<td>TRIG3</td>
</tr>
<tr>
<td>D012</td>
<td>COLPM0 Player-Missile 0</td>
<td>TRIG2</td>
</tr>
<tr>
<td>D011</td>
<td>GRAFM Graphics All Missiles</td>
<td>TRIG1 Buttons</td>
</tr>
<tr>
<td>D010</td>
<td>GRAFP3 Graphics Player 3</td>
<td>TRIG0</td>
</tr>
<tr>
<td>D00F</td>
<td>GRAFP2 Graphics Player 2</td>
<td>P3PL</td>
</tr>
<tr>
<td>D00E</td>
<td>GRAFP1 Graphics Player 1</td>
<td>P2PL</td>
</tr>
<tr>
<td>D00D</td>
<td>GRAFP0 Graphics Player 0</td>
<td>P1PL</td>
</tr>
<tr>
<td>D00C</td>
<td>SIZEM Size All Missiles</td>
<td>POPL</td>
</tr>
<tr>
<td>D00B</td>
<td>SIZE3 Size Player 3</td>
<td>M3PL</td>
</tr>
<tr>
<td>D00A</td>
<td>SIZE2 Size Player 2</td>
<td>M2PL</td>
</tr>
<tr>
<td>D009</td>
<td>SIZE1 Size Player 1</td>
<td>M1PL</td>
</tr>
<tr>
<td>D008</td>
<td>SIZE0 Size Player 0</td>
<td>MOPL</td>
</tr>
<tr>
<td>D007</td>
<td>HPSOM3 Horz. Posit. Missile 3</td>
<td>P3PF</td>
</tr>
<tr>
<td>D006</td>
<td>HPSOM2 Horz. Posit. Missile 2</td>
<td>P2PF</td>
</tr>
<tr>
<td>D005</td>
<td>HPSOM1 Horz. Posit. Missile 1</td>
<td>P1PF</td>
</tr>
<tr>
<td>D004</td>
<td>HPSOM0 Horz. Posit. Missile 0</td>
<td>P0PF</td>
</tr>
<tr>
<td>D003</td>
<td>HPOS3 Horz. Posit. Player 3</td>
<td>M3PF</td>
</tr>
<tr>
<td>D002</td>
<td>HPOS2 Horz. Posit. Player 2</td>
<td>M2PF</td>
</tr>
<tr>
<td>D001</td>
<td>HPOS1 Horz. Posit. Player 1</td>
<td>M1PF</td>
</tr>
<tr>
<td>D000</td>
<td>HPOS0 Horz. Posit. Player 0</td>
<td>M0PF</td>
</tr>
</tbody>
</table>
## ANTIC ADDRESSES

<table>
<thead>
<tr>
<th>Address</th>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>D4FF</td>
<td></td>
<td>REPEAT (AS BELOW)</td>
</tr>
<tr>
<td>D410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D40F</td>
<td>NMIRES</td>
<td>Reset NMI Interrupt Status</td>
</tr>
<tr>
<td>D40E</td>
<td>NMIEN</td>
<td>ENABLE</td>
</tr>
<tr>
<td>D40D</td>
<td>PENV</td>
<td>Light Pen Register Vertical</td>
</tr>
<tr>
<td>D40C</td>
<td>PENH</td>
<td>Light Pen Register Horizontal</td>
</tr>
<tr>
<td>D40B</td>
<td>VCOUNT</td>
<td>Vertical Line Counter</td>
</tr>
<tr>
<td>D40A</td>
<td>WSYNC</td>
<td>Wait for HBLANK Synchronism</td>
</tr>
<tr>
<td>D409</td>
<td>CHBASE</td>
<td>Character Base Address Red</td>
</tr>
<tr>
<td>D408</td>
<td>PMBASE</td>
<td>Player-Missile Base Address Register</td>
</tr>
<tr>
<td>D406</td>
<td>VSCROL</td>
<td>Vertical Scroll Register</td>
</tr>
<tr>
<td>D405</td>
<td>HSCROL</td>
<td>Horizontal Scroll Register</td>
</tr>
<tr>
<td>D404</td>
<td>DLISTH</td>
<td>Display List Pointer (High Byte)</td>
</tr>
<tr>
<td>D403</td>
<td>DLISTL</td>
<td>Display List Pointer (Low Byte)</td>
</tr>
<tr>
<td>D402</td>
<td>CHACTL</td>
<td>Character Control Register</td>
</tr>
<tr>
<td>D401</td>
<td>DMACTL</td>
<td>DMA Control Register</td>
</tr>
<tr>
<td>D400</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## POKEY ADDRESSES

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>D2FF</td>
<td>REPEAT (AS BELOW)</td>
</tr>
</tbody>
</table>

### D210

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D20F</td>
<td>SKCTLS: Serial Port 4 Key Control</td>
</tr>
<tr>
<td>D20E</td>
<td>IRQEN: IRQ Interrupt Enable</td>
</tr>
<tr>
<td>D20D</td>
<td>SEROUT: Serial Port Output Reg.</td>
</tr>
</tbody>
</table>

### D20C

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D20B</td>
<td>POTGO: Start Pot Scan Sequence</td>
</tr>
<tr>
<td>D20A</td>
<td>SKRES: (SKSTAT) Random Numb Generator</td>
</tr>
<tr>
<td>D209</td>
<td>STIMER: Start Timers</td>
</tr>
<tr>
<td>D208</td>
<td>AUDCTL: Audio Control</td>
</tr>
</tbody>
</table>

### D207

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D207</td>
<td>AUDC4: Audio Channel 4 Control</td>
</tr>
<tr>
<td>D206</td>
<td>AUDF4: Audio Channel 4 Frequency</td>
</tr>
</tbody>
</table>

### D205

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D205</td>
<td>AUDC3: Audio Channel 3 Control</td>
</tr>
<tr>
<td>D204</td>
<td>AUDF3: Audio Channel 3 Frequency</td>
</tr>
</tbody>
</table>

### D203

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D203</td>
<td>AUDC2: Audio Channel 2 Control</td>
</tr>
</tbody>
</table>

### D202

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D202</td>
<td>AUDF2: Audio Channel 2 Frequency</td>
</tr>
</tbody>
</table>

### D201

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D201</td>
<td>AUDC1: Audio Channel 1 Control</td>
</tr>
</tbody>
</table>

### D200

<table>
<thead>
<tr>
<th>WRITE</th>
<th>READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>D200</td>
<td>AUDF1: Audio Channel 1 Frequency</td>
</tr>
<tr>
<td>Address</td>
<td>Name</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>D3FF</td>
<td></td>
</tr>
<tr>
<td>D304</td>
<td></td>
</tr>
<tr>
<td>D303</td>
<td>PBCTL</td>
</tr>
<tr>
<td>D302</td>
<td>PACTL</td>
</tr>
<tr>
<td>D301</td>
<td>PORTB</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>D300</td>
<td>PORTA</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* NOTE: Output data is retained in Jack Output Registers. 
If direction bits are true, a read of the jacks will read old data from these registers.
## B. ALPHABETICAL ORDER

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Address</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLPOT</td>
<td>Read 8 line Pot Port State</td>
<td>D208</td>
<td>XX</td>
<td>53768</td>
</tr>
<tr>
<td>AUDC1</td>
<td>Audio Channel 1 Control</td>
<td>D201</td>
<td>XX</td>
<td>53761</td>
</tr>
<tr>
<td>AUDC2</td>
<td>Audio Channel 2 Control</td>
<td>D203</td>
<td>XX</td>
<td>53763</td>
</tr>
<tr>
<td>AUDC3</td>
<td>Audio Channel 3 Control</td>
<td>D205</td>
<td>XX</td>
<td>53765</td>
</tr>
<tr>
<td>AUDC4</td>
<td>Audio Channel 4 Control</td>
<td>D207</td>
<td>XX</td>
<td>53767</td>
</tr>
<tr>
<td>AUDCTL</td>
<td>Audio Control</td>
<td>D208</td>
<td>XX</td>
<td>53768</td>
</tr>
<tr>
<td>AUDF1</td>
<td>Audio Channel 1 Frequency</td>
<td>D200</td>
<td>XX</td>
<td>53760</td>
</tr>
<tr>
<td>AUDF2</td>
<td>Audio Channel 2 Frequency</td>
<td>D202</td>
<td>XX</td>
<td>53762</td>
</tr>
<tr>
<td>AUDF3</td>
<td>Audio Channel 3 Frequency</td>
<td>D204</td>
<td>XX</td>
<td>53764</td>
</tr>
<tr>
<td>AUDF4</td>
<td>Audio Channel 4 Frequency</td>
<td>D206</td>
<td>XX</td>
<td>53766</td>
</tr>
<tr>
<td>CHACTL</td>
<td>Character Control</td>
<td>D401</td>
<td>XX</td>
<td>54273</td>
</tr>
<tr>
<td>CHBASE</td>
<td>Character base address</td>
<td>D409</td>
<td>XX</td>
<td>54281</td>
</tr>
<tr>
<td>COLBK</td>
<td>Color-Luminance of Background</td>
<td>D01A</td>
<td>XX</td>
<td>53274</td>
</tr>
<tr>
<td>COLPF0</td>
<td>Color Luminance of Playfield 0</td>
<td>D016</td>
<td>XX</td>
<td>53270</td>
</tr>
<tr>
<td>COLPF1</td>
<td>Color Luminance of Playfield 1</td>
<td>D017</td>
<td>XX</td>
<td>53271</td>
</tr>
<tr>
<td>COLPF2</td>
<td>Color Luminance of Playfield 2</td>
<td>D018</td>
<td>XX</td>
<td>53272</td>
</tr>
<tr>
<td>COLPF3</td>
<td>Color Luminance of Playfield 3</td>
<td>D019</td>
<td>XX</td>
<td>53273</td>
</tr>
<tr>
<td>COLPM0</td>
<td>Color Luminance of Player-Missile 0</td>
<td>D012</td>
<td>XX</td>
<td>53266</td>
</tr>
<tr>
<td>COLPM1</td>
<td>Color Luminance of Player-Missile 1</td>
<td>D013</td>
<td>XX</td>
<td>53267</td>
</tr>
<tr>
<td>COLPM2</td>
<td>Color Luminance of Player-Missile 2</td>
<td>D014</td>
<td>XX</td>
<td>53268</td>
</tr>
<tr>
<td>COLPM3</td>
<td>Color Luminance of Player-Missile 3</td>
<td>D015</td>
<td>XX</td>
<td>53269</td>
</tr>
<tr>
<td>CONSOL</td>
<td>Console Switch Port</td>
<td>D01F</td>
<td>XX</td>
<td>53279</td>
</tr>
<tr>
<td>DLISTH</td>
<td>Display List Pointer (high byte)</td>
<td>D01B</td>
<td>XX</td>
<td>54275</td>
</tr>
<tr>
<td>DLISTL</td>
<td>Display List Pointer (low byte)</td>
<td>D02</td>
<td>XX</td>
<td>54274</td>
</tr>
<tr>
<td>DMACTL</td>
<td>Direct Memory Access (DMA) Control</td>
<td>D00</td>
<td>XX</td>
<td>54272</td>
</tr>
<tr>
<td>GRACTL</td>
<td>Graphic Control</td>
<td>D01D</td>
<td>XX</td>
<td>53277</td>
</tr>
<tr>
<td>GRAFM</td>
<td>Graphics for all Missiles</td>
<td>D011</td>
<td>XX</td>
<td>53265</td>
</tr>
<tr>
<td>GRAFP0</td>
<td>Graphics for Player 0</td>
<td>D00D</td>
<td>XX</td>
<td>53261</td>
</tr>
<tr>
<td>GRAFP1</td>
<td>Graphics for Player 1</td>
<td>D00E</td>
<td>XX</td>
<td>53262</td>
</tr>
<tr>
<td>GRAFP2</td>
<td>Graphics for Player 2</td>
<td>D00F</td>
<td>XX</td>
<td>53263</td>
</tr>
<tr>
<td>GRAFP3</td>
<td>Graphics for Player 3</td>
<td>D010</td>
<td>XX</td>
<td>53264</td>
</tr>
<tr>
<td>HITCLR</td>
<td>Collision Clear</td>
<td>D01E</td>
<td>XX</td>
<td>53278</td>
</tr>
<tr>
<td>HPOS0</td>
<td>Horizontal Position of Missile 0</td>
<td>D004</td>
<td>XX</td>
<td>53252</td>
</tr>
<tr>
<td>HPOS1</td>
<td>Horizontal Position of Missile 1</td>
<td>D005</td>
<td>XX</td>
<td>53253</td>
</tr>
<tr>
<td>HPOS2</td>
<td>Horizontal Position of Missile 2</td>
<td>D006</td>
<td>XX</td>
<td>53254</td>
</tr>
<tr>
<td>HPOS3</td>
<td>Horizontal Position of Missile 3</td>
<td>D007</td>
<td>XX</td>
<td>53255</td>
</tr>
<tr>
<td>HPOS0</td>
<td>Horizontal Position of Player 0</td>
<td>D000</td>
<td>XX</td>
<td>53248</td>
</tr>
<tr>
<td>HPOS1</td>
<td>Horizontal Position of Player 1</td>
<td>D001</td>
<td>XX</td>
<td>53249</td>
</tr>
<tr>
<td>HPOS2</td>
<td>Horizontal Position of Player 2</td>
<td>D002</td>
<td>XX</td>
<td>53250</td>
</tr>
<tr>
<td>HPOS3</td>
<td>Horizontal Position of Player 3</td>
<td>D003</td>
<td>XX</td>
<td>53251</td>
</tr>
<tr>
<td>HSCROL</td>
<td>Horizontal Scroll</td>
<td>D01B</td>
<td>XX</td>
<td>53277</td>
</tr>
<tr>
<td>IRQEN</td>
<td>Interrupt Request (IRQ) Enable</td>
<td>D02E</td>
<td>XX</td>
<td>53774</td>
</tr>
<tr>
<td>IROST</td>
<td>IRQ Status</td>
<td>D02E</td>
<td>XX</td>
<td>53774</td>
</tr>
<tr>
<td>KBCODE</td>
<td>Keyboard Code</td>
<td>D01A</td>
<td>XX</td>
<td>53274</td>
</tr>
<tr>
<td>MOPF</td>
<td>Missile 0 to Playfield Collisions</td>
<td>D008</td>
<td>XX</td>
<td>53248</td>
</tr>
<tr>
<td>MOP0L</td>
<td>Missile 0 to Player Collisions</td>
<td>D008</td>
<td>XX</td>
<td>53256</td>
</tr>
<tr>
<td>M1PF</td>
<td>Missile 1 to Playfield Collisions</td>
<td>D001</td>
<td>XX</td>
<td>53249</td>
</tr>
<tr>
<td>M1PL</td>
<td>Missile 1 to Player Collisions</td>
<td>D009</td>
<td>XX</td>
<td>53257</td>
</tr>
<tr>
<td>M2PF</td>
<td>Missile 2 to Playfield Collisions</td>
<td>D002</td>
<td>XX</td>
<td>53250</td>
</tr>
<tr>
<td>M2PL</td>
<td>Missile 2 to Player Collisions</td>
<td>D00A</td>
<td>XX</td>
<td>53258</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Address</td>
<td>Hex</td>
<td>Dec</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------</td>
<td>---------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>M3PF</td>
<td>Missile 3 to Playfield Collisions</td>
<td>D003</td>
<td>53251</td>
<td></td>
</tr>
<tr>
<td>M3PL</td>
<td>Missile 3 to Player</td>
<td>D00B</td>
<td>53259</td>
<td></td>
</tr>
<tr>
<td>NMIEN</td>
<td>Non-Maskable Interrupt (NMI) Enable</td>
<td>D40E</td>
<td>54286</td>
<td></td>
</tr>
<tr>
<td>NMIRES</td>
<td>NMI reset</td>
<td>D40F</td>
<td>54287</td>
<td></td>
</tr>
<tr>
<td>NMIST</td>
<td>NMI Status</td>
<td>D40F</td>
<td>54287</td>
<td></td>
</tr>
<tr>
<td>POPF</td>
<td>Player 0 to Playfield Collisions</td>
<td>D004</td>
<td>53252</td>
<td></td>
</tr>
<tr>
<td>POPL</td>
<td>Player 0 to Player Collisions</td>
<td>D00C</td>
<td>53260</td>
<td></td>
</tr>
<tr>
<td>P1PF</td>
<td>Player 1 to Playfield Collisions</td>
<td>D005</td>
<td>53253</td>
<td></td>
</tr>
<tr>
<td>P1PL</td>
<td>Player 1 to Player Collisions</td>
<td>D00D</td>
<td>53261</td>
<td></td>
</tr>
<tr>
<td>P2PL</td>
<td>Player 2 to Playfield Collisions</td>
<td>D00E</td>
<td>53262</td>
<td></td>
</tr>
<tr>
<td>P2PF</td>
<td>Player 2 to Player Collisions</td>
<td>D00F</td>
<td>53263</td>
<td></td>
</tr>
<tr>
<td>P3PF</td>
<td>Player 3 to Playfield Collisions</td>
<td>D007</td>
<td>53255</td>
<td></td>
</tr>
<tr>
<td>P3PL</td>
<td>Player 3 to Player Collisions</td>
<td>D00F</td>
<td>53263</td>
<td></td>
</tr>
<tr>
<td>PACTL</td>
<td>Port A Control</td>
<td>D302</td>
<td>54018</td>
<td></td>
</tr>
<tr>
<td>PAL</td>
<td>PAL/NTSC Indicator</td>
<td>D014</td>
<td>53268</td>
<td></td>
</tr>
<tr>
<td>PBCTL</td>
<td>Port B Control</td>
<td>D303</td>
<td>54019</td>
<td></td>
</tr>
<tr>
<td>PENH</td>
<td>Light Pen Horizontal Position</td>
<td>D40C</td>
<td>54284</td>
<td>LPENH</td>
</tr>
<tr>
<td>PENV</td>
<td>Light Pen Vertical Position</td>
<td>D40D</td>
<td>54285</td>
<td>LPENV</td>
</tr>
<tr>
<td>PMBASE</td>
<td>Player Missile Base Address</td>
<td>D407</td>
<td>54279</td>
<td></td>
</tr>
<tr>
<td>PORTA</td>
<td>Port A</td>
<td>D300</td>
<td>54016</td>
<td>STICK0,1</td>
</tr>
<tr>
<td>PORTB</td>
<td>Port A</td>
<td>D301</td>
<td>54017</td>
<td>STICK2,3</td>
</tr>
<tr>
<td>POT0</td>
<td>Pot 0</td>
<td>D200</td>
<td>53760</td>
<td></td>
</tr>
<tr>
<td>POT1</td>
<td>Pot 1</td>
<td>D201</td>
<td>53761</td>
<td></td>
</tr>
<tr>
<td>POT2</td>
<td>Pot 2</td>
<td>D202</td>
<td>53762</td>
<td></td>
</tr>
<tr>
<td>POT3</td>
<td>Pot 3</td>
<td>D203</td>
<td>53763</td>
<td></td>
</tr>
<tr>
<td>POT4</td>
<td>Pot 4</td>
<td>D204</td>
<td>53764</td>
<td></td>
</tr>
<tr>
<td>POT5</td>
<td>Pot 5</td>
<td>D205</td>
<td>53765</td>
<td></td>
</tr>
<tr>
<td>POT6</td>
<td>Pot 6</td>
<td>D206</td>
<td>53766</td>
<td></td>
</tr>
<tr>
<td>POT7</td>
<td>Pot 7 (right paddle controller)</td>
<td>D207</td>
<td>53767</td>
<td></td>
</tr>
<tr>
<td>POTGO</td>
<td>Start POT Scan Sequence</td>
<td>D20B</td>
<td>53771</td>
<td></td>
</tr>
<tr>
<td>PRIOR</td>
<td>Priority Select</td>
<td>D01B</td>
<td>53275</td>
<td></td>
</tr>
<tr>
<td>RANDOM</td>
<td>Random number generator</td>
<td>D20A</td>
<td>53770</td>
<td></td>
</tr>
<tr>
<td>SERIN</td>
<td>Serial Port Input</td>
<td>D20E</td>
<td>53774</td>
<td></td>
</tr>
<tr>
<td>SEROUT</td>
<td>Serial Port output</td>
<td>D20D</td>
<td>53773</td>
<td></td>
</tr>
<tr>
<td>SIZEM</td>
<td>Sizes for all missiles</td>
<td>D00C</td>
<td>53260</td>
<td></td>
</tr>
<tr>
<td>SIZEP0</td>
<td>Size of Player 0</td>
<td>D008</td>
<td>53256</td>
<td></td>
</tr>
<tr>
<td>SIZEP1</td>
<td>Size of Player 1</td>
<td>D009</td>
<td>53257</td>
<td></td>
</tr>
<tr>
<td>SIZEP2</td>
<td>Size of Player 2</td>
<td>D00A</td>
<td>53258</td>
<td></td>
</tr>
<tr>
<td>SIZEP3</td>
<td>Size of Player 3</td>
<td>D00B</td>
<td>53259</td>
<td></td>
</tr>
<tr>
<td>SKCTL</td>
<td>Serial Port Control</td>
<td>D20F</td>
<td>53775</td>
<td></td>
</tr>
<tr>
<td>SKREST</td>
<td>Reset Serial Port Status (SKSTAT)</td>
<td>D20A</td>
<td>53770</td>
<td></td>
</tr>
<tr>
<td>SKSTAT</td>
<td>Serial Port Status</td>
<td>D20F</td>
<td>53775</td>
<td></td>
</tr>
<tr>
<td>STIMER</td>
<td>Start Timer</td>
<td>D209</td>
<td>53769</td>
<td></td>
</tr>
<tr>
<td>TRIG0</td>
<td>Joystick Controller Trigger 0</td>
<td>D010</td>
<td>53264</td>
<td></td>
</tr>
<tr>
<td>TRIG1</td>
<td>Joystick Controller Trigger 1</td>
<td>D011</td>
<td>53265</td>
<td></td>
</tr>
<tr>
<td>TRIG2</td>
<td>Joystick Controller Trigger 2</td>
<td>D012</td>
<td>53266</td>
<td></td>
</tr>
<tr>
<td>TRIG3T</td>
<td>Joystick Controller Trigger 3</td>
<td>D013</td>
<td>53267</td>
<td></td>
</tr>
<tr>
<td>VCOUNT</td>
<td>Vertical Line Counter</td>
<td>D40B</td>
<td>54283</td>
<td></td>
</tr>
<tr>
<td>VDELAY</td>
<td>Vertical Delay</td>
<td>D01C</td>
<td>54276</td>
<td></td>
</tr>
<tr>
<td>VSCROL</td>
<td>Vertical Scroll</td>
<td>D405</td>
<td>54277</td>
<td></td>
</tr>
<tr>
<td>WSYNC</td>
<td>Wait for Horizontal Sync</td>
<td>D40A</td>
<td>54282</td>
<td></td>
</tr>
</tbody>
</table>
VI. FIGURES

A. MEMORY MAP

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>FUNCTION</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFF</td>
<td>Operating System And Math Routines</td>
<td>10K</td>
</tr>
<tr>
<td>D800</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D000-D7FF</td>
<td>Hardware Addresses</td>
<td>2K</td>
</tr>
<tr>
<td>CFFF</td>
<td>Reserved for Future O.S. expansion</td>
<td>4K</td>
</tr>
<tr>
<td>BFFF</td>
<td>ROM Cartridge</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Colleen left and right slot and Candy single slot all address to this space)</td>
<td>16K</td>
</tr>
<tr>
<td>8000</td>
<td>RAM Expansion *</td>
<td></td>
</tr>
<tr>
<td>7FFF</td>
<td>RAM</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1FFF</td>
<td>RAM initially supplied in the product</td>
<td>8K</td>
</tr>
</tbody>
</table>

* RAM expansion can actually extend to BFFF. However, the ROM cartridges will deselect the RAM. Deselection occurs on 8K boundaries. Atari 400 units are RAM expandable only at the factory. They can accept RAM up to 2FFF (16K) when fully extended.
POWER SUPPLY BOARD
ATARI 800

(VOLTAGE BOOST)

(INTERLOCK SWITCH)

(POWER SWITCH)

(VOLTAGE BOOST)

(RIPPLE FILTER)

(12 VOLT REGULATOR)

(+ 5A)

(+ 5B)

(-5 VOLT REGULATOR)

(POWER ON LITE)

(CONTROL SWITCHES)

(J201)

(J205)

(J202)

(J204)

(TO MOTHER BOARD)

(TO TELEVISION)

(TO PERIPHERAL JACK)

(COMPOSITE VIDEO (TO MONITOR JACK))
SCHEMATIC ROM CARTRIDGE
APPENDIX A
USE OF PLAYER/MISSILE GRAPHICS WITH BASIC

The ATARI® 400/800™ Hardware Manual should be read first to understand the details of the Player/Missile Graphics.

To enable the P/M Graphics from BASIC the following procedure can be used:

1. Generate the playfield, either with a GRAPHICS call or build a custom display list with a series of POKE statements.
2. Enable P/M DMA control by a POKE 559 with either a 62 for single line resolution players or a 46 for double line resolution players.
3. There are four players and four missiles (or five players if the four missiles are combined into one player). Each of these has a horizontal position register that controls its horizontal position on the screen. The registers and their locations are as follows:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>HORIZONTAL POSITION OF</th>
</tr>
</thead>
<tbody>
<tr>
<td>53248</td>
<td>Player 0</td>
</tr>
<tr>
<td>53249</td>
<td>Player 1</td>
</tr>
<tr>
<td>53250</td>
<td>Player 2</td>
</tr>
<tr>
<td>53251</td>
<td>Player 3</td>
</tr>
<tr>
<td>53252</td>
<td>Missile 1</td>
</tr>
<tr>
<td>53253</td>
<td>Missile 2</td>
</tr>
<tr>
<td>53254</td>
<td>Missile 3</td>
</tr>
<tr>
<td>53255</td>
<td>Missile 4</td>
</tr>
</tbody>
</table>

The horizontal positions can range on the playfield between 41 and 200. So POKE 53249,120 will move Player 1 to the middle of the screen.

*NOTE: All number references are decimal.
Use of Player/Missile Graphics with BASIC, cont.

4. Each player (and its missile) has a color register which determines its color. These registers can be controlled by poking to the following locations:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>COLOR OF</th>
</tr>
</thead>
<tbody>
<tr>
<td>704</td>
<td>P/M 0</td>
</tr>
<tr>
<td>705</td>
<td>P/M 1</td>
</tr>
<tr>
<td>706</td>
<td>P/M 2</td>
</tr>
<tr>
<td>707</td>
<td>P/M 3</td>
</tr>
<tr>
<td>711</td>
<td>fifth player (if enabled)</td>
</tr>
</tbody>
</table>

Thus a POKE 706,200 will color player 2 green.

5. The P/M bit information (those bytes which actually describe the shape of the player) must be stored in an area where it will not interfere with BASIC or the operating system. It must also start at a 2K memory boundary if single line resolution players are used, or a 1K boundary for double line resolution players.

6. The page number (i.e. number of 256 byte sections of memory) for the starting address of the P/M information obtained in step 5 is poked into location 54279.

7. Enable the P/M DMA by a POKE 53277,3.

8. The starting address of each player is obtained by multiplying the number obtained in step 6 by 256 and then adding the offset indicated in P/M memory configuration table.

9. The vertical position of the player is determined by its location in memory. After the initial offset is obtained in step 8, its height may be defined. Its range on the playfield is from 32 to 223 in single line resolution and from 16 to 111 in double line resolution. By adding the desired height to the initial offset, the absolute address of each player is found. The appropriate bit information for the player can now be poked into this address.
Use of Player/Missile Graphics with BASIC, cont.

(9, cont.)

Example to Generate a rectangular box player, eight color clocks wide and four lines high in immediate mode.

<table>
<thead>
<tr>
<th>STEP</th>
<th>TYPE</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GRAPHICS 8</td>
<td>Setup Mode 8 Playfield</td>
</tr>
<tr>
<td>2</td>
<td>POKE 559, 62</td>
<td>Enable P/M DMA single line</td>
</tr>
<tr>
<td>3</td>
<td>POKE 53248,120</td>
<td>Set horizontal position</td>
</tr>
<tr>
<td>4</td>
<td>POKE 704,88</td>
<td>Set color to pink</td>
</tr>
<tr>
<td>5</td>
<td>I = PEEK(106)-8</td>
<td>Get P/M base address</td>
</tr>
<tr>
<td>6</td>
<td>POKE 54279,I</td>
<td>Store in base register</td>
</tr>
<tr>
<td>7</td>
<td>POKE 53277,3</td>
<td>Enable P/M DMA</td>
</tr>
<tr>
<td>8</td>
<td>J = I * 256 + 1024</td>
<td>Get player starting address</td>
</tr>
<tr>
<td>9</td>
<td>POKE J + 125,255</td>
<td>Draw player on screen</td>
</tr>
<tr>
<td></td>
<td>POKE J + 126,129</td>
<td></td>
</tr>
<tr>
<td></td>
<td>POKE J + 127,129</td>
<td></td>
</tr>
<tr>
<td></td>
<td>POKE J + 128,255</td>
<td></td>
</tr>
</tbody>
</table>
Use of Player/Missile Graphics with BASIC, cont.

<table>
<thead>
<tr>
<th>DMCTL bit D4=0</th>
<th>DMCTL bit D4=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>start at PMBASE*1024</td>
<td>start at PMBASE*2048</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>8 bits wide</th>
<th>0</th>
<th>8 bits wide</th>
</tr>
</thead>
<tbody>
<tr>
<td>+384</td>
<td>384 bytes unused</td>
<td>+768</td>
<td>768 bytes unused</td>
</tr>
<tr>
<td>+512</td>
<td>M3 M2 M1 M0</td>
<td>+768</td>
<td>Player 1</td>
</tr>
<tr>
<td>+640</td>
<td>Player 0</td>
<td>+1024</td>
<td>Player 2</td>
</tr>
<tr>
<td>+768</td>
<td>Player 1</td>
<td>+1280</td>
<td>Player 3</td>
</tr>
<tr>
<td>+896</td>
<td>Player 2</td>
<td>+1536</td>
<td>256 bytes per player</td>
</tr>
<tr>
<td>+1024</td>
<td>Player 3</td>
<td>+1792</td>
<td>single line resolution</td>
</tr>
<tr>
<td>+1536</td>
<td>128 bytes per player</td>
<td>+2048</td>
<td>double line resolution</td>
</tr>
</tbody>
</table>

PLAYER-MISSILE Memory Configuration

Absolute address determined by PMBASE.

Relative address shown along sides of maps.

Each Player-Missile section (128 bytes in single line, 256 bytes in double line) maps directly onto the total height of TV screen.
APPENDIX B

MIXING GRAPHICS MODES

I. GENERAL

This procedure describes how to mix several graphic modes on the TV screen at the same time using BASIC commands. Each graphics mode has a different number of scan lines per "Mode Line" (one line of a graphics mode). The TV screen must consist of 192 scan lines, so when mixing modes, they must be combined in such a way as to get 192 scan lines. This is accomplished by modifying the Display List.

When a graphics mode is set on the computer, the O/S allocates RAM space for the graphics mode, then builds the display list adjacent to the graphics RAM, and sets a pointer to the beginning of the display list. Each "mode line" is constructed from a "mode byte" in the display list that determines how many scan lines in each mode line. The display list describes the screen display from top to bottom.

A Display List must be built for the "max RAM mode" (the graphics mode that requires the most RAM) then modified with POKEs to mix the other modes with it. This "max RAM mode" cannot be a split screen mode (text window), therefore "max RAM mode" +16 must be used. If the max RAM mode will be at the top of the screen, then the "LMS byte" (load memory scan byte) at the top of the Display List will already be correct. If not, the "LMS byte" will have to be modified.

The Display List is modified by POKEING a new mode byte for each mode line that is not a max RAM mode line. At the end of the display list is a JUMP instruction pointing to the top of the Display list. When the Display List is modified, the JUMP instruction must be placed immediately after the last mode byte.

Example #1 will be used throughout this procedure to illustrate each step.
MODE 7
96 LINES

MOOD 7
56 LINES

MODE 2
2 LINES

TOTAL = 192

Example #1
Mixing Graphics Modes, cont.

II. PROCEDURE TO SET UP SCREEN IN MIXED MODES:

1. Select modes desired, then look up which mode is the max RAM mode from table #2.

   example: modes selected - mode 1, mode 7, mode 2
   mode 7 = max RAM mode

2. Use table #1 to calculate the number of mode lines such that the total number of scan lines = 192.

   example:
   
<table>
<thead>
<tr>
<th>mode</th>
<th># mode per scan line</th>
<th>scan lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>56</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3. If the max RAM mode is at the top of the screen, then skip this step: Calculate the LMS byte by setting the left nibble to 4, then use table #1 to find the right nibble for the graphics mode at the top of the screen.

   example: 1. left nibble = 4
   2. right nibble for mode 1 = 6
   3. LMS byte = 46 (HEX)

4. Calculate the mode byte for each mode. Set the left nibble to 0, use table #1 to find the right nibble for each mode.

   example:
   
<table>
<thead>
<tr>
<th>Mode</th>
<th>Left Nibble</th>
<th>Right Nibble</th>
<th>Mode Byte (HEX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>6</td>
<td>06</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>D</td>
<td>0D</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>7</td>
<td>07</td>
</tr>
</tbody>
</table>
II. PROCEDURE TO SET UP SCREEN IN MIXED MODES, cont.:

5. Convert all bytes to decimal.

<table>
<thead>
<tr>
<th>example: Byte</th>
<th>(HEX)</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMS</td>
<td>46</td>
<td>70</td>
</tr>
<tr>
<td>Mode 1</td>
<td>06</td>
<td>6</td>
</tr>
<tr>
<td>Mode 7</td>
<td>0D</td>
<td>13</td>
</tr>
<tr>
<td>Mode 2</td>
<td>07</td>
<td>7</td>
</tr>
</tbody>
</table>

6. Execute a graphics call on the computer using the max RAM mode (+16).

example: GRAPHICS 7 + 16

7. PEEK the Display List pointer and use it to calculate a variable labelled "START".

example: START = PEEK(560) + PEEK(561) * 256 + 4

8. If the max RAM mode is at the top of the screen, then skip this step: Poke the LMS byte to location START-1.

example: POKE START-1,70

9. Every mode line requires a mode byte in the Display List in the same order as the mode lines appear on the screen. The mode bytes must be POKED into the Display List at location START + offset, where offset = mode line #.

<table>
<thead>
<tr>
<th>Example:</th>
<th>MODE LINE #</th>
<th>POKE INSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE 1</td>
<td>2</td>
<td>POKE START + 2,6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>POKE START + 3,6</td>
</tr>
<tr>
<td>MODE 7</td>
<td>4</td>
<td>POKE START + 4,6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>POKE START + 5,6</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>POKE START + 6,6</td>
</tr>
<tr>
<td>MODE 2</td>
<td>63</td>
<td>POKE START + 63,7</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>POKE START + 64,7</td>
</tr>
</tbody>
</table>

NOTE: The Display List will already be correct for the max RAM mode, therefore its mode bytes do not need to be POKEd.
II. PROCEDURE TO SET UP SCREEN IN MIXED MODES, cont.:  

10. POKE the JUMP instruction followed by the LO byte, then the HI byte into the Display List. The offset for the JUMP POKE is the last mode line # + 1, for LO byte it is + 2, for HI byte it is + 3.

example: (last mode line # was 64)

<table>
<thead>
<tr>
<th>REMARK</th>
<th>POKE INSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUMP</td>
<td>POKE START + 65,65</td>
</tr>
<tr>
<td>LO BYTE</td>
<td>POKE START + 66, PEEK(560)</td>
</tr>
<tr>
<td>HI BYTE</td>
<td>POKE START + 67, PEEK(561)</td>
</tr>
</tbody>
</table>

III. PROCEDURE TO PRINT AND PLOT IN MIXED MODES

1. If the mode line #’s of a mode on the screen fall within the range of that mode’s normal mode line #’s then use the following procedure:
   a. POKE 87 with the mode #
   b. Determine the Y coordinate by counting the # of mode lines from the top of the screen to the current position.
   c. Determine the X position in the normal manner for that mode.
   d. Depending on the mode, either PLOT and DRAWTO, or POSITION and PRINT.
   e. These steps must be done for each mode on the screen that meets the condition in step 1.

example: MODE 1  
POKE 87,1
POSITION 2,1:PRINT #6;"TEXT"

MODE 7  
POKE 87,7
COLOR 1:PLOT 20,20:DRAWTO 30,30

MODE 2  
See step 2
III. PROCEDURE TO PRINT AND PLOT IN MIXED MODES, cont.

2. Some modes may have mode line #'s outside of their normal range.

   example: Mode 2 normally has mode line #'s 1 through 12
   (full screen). These are modified to #63 and #64 in example #1.

To prevent the computer from giving a "cursor out of range" error message the following procedure can be used:
   a. Set a variable labelled "MEMST" to be the display
      memory start pointer.
         MEMST = PEEK(START) + PEEK(START + 1) * 256
   b. Set a variable labelled CHRPOS to position characters
      to be printed on the target line.

         CHRPOS = MEMST + [(M₁ -1)*R-M₂ *(R-20) -M₃ *(R-10)] +X

Where:

   X = horizontal position of character on the target line.
   R = the RAM per line of the Max RAM Mode (table #1).
   M₁ = the Mode Line # of the target line.
   M₂ = the number of mode lines of 20 bytes of RAM per
   line above the target line.
   M₃ = the number of mode lines of 10 bytes of RAM per
   line above the target line.

Example: calculate CHRPOS for Mode Line #64 (the
last line of the Mode 2 area) at horizontal position 5.

   X = 5
   R = 40
   M₁ = 64
   M₂ = 7 (6 from Mode 1 area, 1 from Mode 2 area).
   M₃ = 0

   CHRPOS = MEMST + [(64-1)*40-7*(40-20)-0*(40-10)] +5
   CHRPOS = MEMST + [(63)*40-7*(20)-0*(30)] +5
   CHRPOS = MEMST + [2520 - 140] + 5
   CHRPOS = MEMST + 2380 + 5
   CHRPOS = MEMST + 2385
Mixing Graphics Modes, cont.

III. PROCEDURE TO PRINT AND PLOT IN MIXED MODES, cont.

2. cont.
   c. If few characters will be printed, then each character’s internal value may be looked up in the Internal Character Set Table (Table 9.6), in the new BASIC Reference Manual. This value is then POKEd into CHRPOS.
   d. If strings are to be output, and if the ATASCII values of all the characters lie within one of the ranges shown in the table below, then do the following:
      1) Obtain the appropriate ATASCII value range for the characters
      2) Do the OPERATION the table indicates on the ATASCII value of each character.
      3) POKE this value into CHRPOS.

<table>
<thead>
<tr>
<th>ATASCII VALUE RANGE</th>
<th>OPERATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-31</td>
<td>Value + 64</td>
</tr>
<tr>
<td>32-95</td>
<td>Value - 32</td>
</tr>
<tr>
<td>96-127</td>
<td>NONE</td>
</tr>
<tr>
<td>128-159</td>
<td>Value + 64</td>
</tr>
<tr>
<td>160-223</td>
<td>Value - 32</td>
</tr>
<tr>
<td>224-255</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Example: 1) assume we want to print the word "TEXT" in the mode 2 area of example #1 using the CHRPOS calculated previously.
2) these characters are in the ATASCII VALUE RANGE of "32 - 95".
3) the OPERATION for this range is "Value-32", so 32 must be subtracted from each ATASCII value.
4) the program statements would now look like this:

```
T$(1,4) = "TEXT"
CHRPOS = MEMST + 2385
FOR X = 1 TO LEN(T$)
   POKE CHRPOS + X - 1, ASC[T$(X,X)] - 32
NEXT X
```

(OPERATION: value - 32)

5) the FOR/NEXT loop POKEs the first character of T$, ASC[T$(X,X)]-32, into CHRPOS + 0.
6) the next iteration POKEs the next character of T$ into the next CHRPOS, and so on.
Mixing Graphics Modes, cont.

### TABLE #1

<table>
<thead>
<tr>
<th>REMARK</th>
<th>LEFT NIBBLE (HEX)</th>
<th>RIGHT NIBBLE (HEX)</th>
<th>SCAN LINES</th>
<th>C.C. PER PIXEL</th>
<th>MODE COLORS</th>
<th>RAM PER LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>4</td>
<td>2</td>
<td>(\frac{1}{2})</td>
<td>8</td>
<td>1(\frac{1}{2})</td>
<td>0</td>
</tr>
<tr>
<td>MODES</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>16</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>MODES</td>
<td>0</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>BLANK</td>
<td>0-7</td>
<td>4</td>
<td>0</td>
<td>BLANK</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JUMP</td>
<td>4 SPECIAL</td>
<td>1</td>
<td>JUMP</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. When the max RAM mode is not at the top of the screen, the left nibble of the LMS byte must be changed to a 4.

2. Left nibble for all mode bytes after the LMS byte.

3. Color & Lum for the field is controlled by Setcolor 2, and Lum for characters or graphics from Setcolor 1.

4. JUMP - used to end the display list and return to the beginning.

BLANK - to output selected number of background lines.
Mixing Graphics Modes, cont.

**TABLE #2**

<table>
<thead>
<tr>
<th>GRAPHICS MODES</th>
<th>RAM REQUIREMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 8 + 16</td>
<td>8138 Bytes</td>
</tr>
<tr>
<td>8</td>
<td>8112</td>
</tr>
<tr>
<td>7 + 16</td>
<td>4200</td>
</tr>
<tr>
<td>7</td>
<td>4190</td>
</tr>
<tr>
<td>6 + 16</td>
<td>2184</td>
</tr>
<tr>
<td>6</td>
<td>2174</td>
</tr>
<tr>
<td>5 + 16</td>
<td>1176</td>
</tr>
<tr>
<td>5</td>
<td>1174</td>
</tr>
<tr>
<td>4 + 16</td>
<td>696</td>
</tr>
<tr>
<td>4</td>
<td>694</td>
</tr>
<tr>
<td>3 + 16</td>
<td>432</td>
</tr>
<tr>
<td>3</td>
<td>434</td>
</tr>
<tr>
<td>2 + 16</td>
<td>420</td>
</tr>
<tr>
<td>2</td>
<td>424</td>
</tr>
<tr>
<td>1 + 16</td>
<td>672</td>
</tr>
<tr>
<td>1</td>
<td>674</td>
</tr>
<tr>
<td>0</td>
<td>992</td>
</tr>
</tbody>
</table>

These values include the display list and any imbedded unused memory blocks.
MEMORY CONFIGURATIONS
FOR MODES 5 - 8

MODE: 5 5+16 6 6+16 7 7+16 8 8+16

8192 Bytes
6144 Bytes
4096 Bytes
2048 Bytes

80 unused
8 unused

6400 bytes
7680 bytes

176 bytes DL
202 bytes DL

94 bytes DL
104 bytes DL

96 unused
96 unused

3200 bytes
3840 bytes

800 bytes
960 bytes

bit map
bit map

1600 bytes
1920 bytes

bit
bit

320 bytes
640 bytes

bit
bit

1280 bytes
16 unused

unused

160 text
160 unused

160 text
160 unused

160 text
160 unused

160 text
160 unused

160 text
160 unused

160 text
160 unused

160 text
160 unused

160 text
160 unused
Monitor Jack (800 only)

D.I.N. 5 Jack

Audio Output
Composite Chroma
Ground

1. Clock Input
2. Clock Output
3. Data Input
4. Ground
5. Data Output
6. Ground
7. Command
8. Motor Control
9. Proceed
10. +5/Ready
11. Audio Input
12. +12 volts
13. Interrupt

Serial I/O Jack

Controller Jack

1. (Joystick) Forward Input
2. (Joystick) Back Input
3. (Joystick) Left Input
4. (Joystick) Right Input
5. B Potentiometer Input
6. Trigger Input
7. +5 volts
8. Ground
9. A Potentiometer Input