TECHNICAL MANUAL

TIMEX SINCLAIR
2068
PERSONAL
COLOR COMPUTER

Published by The Time Designs Magazine Co.
TIMEX SINCLAIR 2068
PERSONAL COLOR COMPUTER

TECHNICAL REFERENCE MANUAL

Prepared by

V. C. Corcoran
and
M. H. Branigin

TIMEX COMPUTER CORPORATION
Waterbury CT 06720
© May 1984
Second Edition Printing
Published Exclusively by:
TIME DESIGNS MAGAZINE CO.
COLTON, OREGON 97017
© JANUARY 1986
PREFACE

This manual is dedicated to the many individuals associated with the Timex Computer Corporation in the development and production of the TS2068. Our special thanks to Nan Parsons who prepared the TS2068 Schematic and other drawings used in this manual.

While every effort has been made to make this document complete and accurate, use of the technical information contained herein is at user's sole risk. The Timex Corp. or its affiliates, and Time Designs Magazine Company assume no responsibility or liability for the safety or performance of any product manufactured relying on the technical data contained herein, or any liability, loss, damage, or expense sustained by reason of any claim that such products infringe any patent or other industrial property right.

The Second Edition of this Technical Manual has been re-edited by Tim Woods. Special thanks to Bob Orrfelt and Dave Clifford for technical assistance.

If you would like to receive information on a magazine and other publications for the Timex Sinclair 2068, direct your inquiry to: Time Designs Magazine Company, 29722 Hult Rd., Colton, OR 97017.


This manual was printed by Toad'l Litho Printing and Composition, Oregon City, OR 97045.
# TABLE OF CONTENTS

1.0 INTRODUCTION  1

1.1 TS 2068 Overview  1
   1.1.1 Hardware Overview  
   1.1.2 System Software Overview  
   1.1.3 Cartridge Software Overview  

2.0 HARDWARE GUIDE  7

2.1 Major Hardware Functions  7
   2.1.1 AC Adapter  
   2.1.2 Voltage Regulation  
   2.1.3 Z80A CPU  
   2.1.4 ROM  
   2.1.5 32K RAM  
   2.1.6 Programmable Sound Generator  
   2.1.7 Joystick Port  
   2.1.8 Control Logic  
   2.1.9 Keyboard  
   2.1.10 16K Video Display RAM  
   2.1.11 Video Generation  
   2.1.12 Cassette I/O  
   2.1.13 Port Map  

2.2 Schematic (see inside back cover and Appendix D)  

2.3 Unit Absolute Ratings  53  

2.4 Interfaces and Connectors  53
   2.4.1 System Bus Connector - P1  
   2.4.2 Cartridge Connector - J4  
   2.4.3 Cassette I/O  
   2.4.4 Joystick  
   2.4.5 Composite Monitor Output  
   2.4.6 RF Output  
   2.4.7 Keyboard Interface Connector - J9  
   2.4.8 AC Adapter Power Plug  

3.0 SYSTEM SOFTWARE GUIDE  65

3.1 Identifier  65
TABLE OF CONTENTS
(continued)

3.2 ROM Organization and Services 65
  3.2.1 Home ROM
    3.2.1.1 Fixed Entry Points
    3.2.1.2 BASIC AROS Support
    3.2.1.3 General
  3.2.2 Extension ROM
    3.2.2.1 Fixed Entry Points
    3.2.2.2 General
    3.2.2.3 Video Mode Change Service
    3.2.2.4 Extension ROM Interface Routine

3.3 RAM Organization and Services 72
  3.3.1 System Variables
  3.3.2 System Configuration Table
  3.3.3 Machine Stack
  3.3.4 OS RAM Routines
    3.3.4.1 RAM Interruption Handler
    3.3.4.2 RAM Service Routines
    3.3.4.3 Function Dispatcher

4.0 SYSTEM I/O GUIDE 91
  4.1 I/O Channels 91
    4.1.1 Keyboard
    4.1.2 Video Screen
    4.1.3 2040 Dot Matrix Printer

4.2 Cassette Tape 102
  4.3 Joysticks 104
  4.4 Software Generated Sound (BEEP) 105
  4.5 Programmable Sound Chip (SOUND) 105

5.0 ADVANCED CONCEPTS 106
  5.1 Cartridge Software/Hardware 106
  5.2 Advanced Video Modes 117
  5.3 Other 125
# TABLE OF CONTENTS

(continued)

<table>
<thead>
<tr>
<th>6.0</th>
<th>KNOWN &quot;BUGS&quot; AND CORRECTIONS</th>
<th>126</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>LROS and Machine Code AROS</td>
<td>126</td>
</tr>
<tr>
<td>6.2</td>
<td>Machine Code AROS</td>
<td>126</td>
</tr>
<tr>
<td>6.3</td>
<td>BASIC AROS</td>
<td>127</td>
</tr>
<tr>
<td>6.4</td>
<td>Video Mode Change Service</td>
<td>127</td>
</tr>
<tr>
<td>6.5</td>
<td>OS RAM Routines</td>
<td>129</td>
</tr>
<tr>
<td>6.6</td>
<td>General</td>
<td>134</td>
</tr>
</tbody>
</table>

## APPENDICES

Appendix A - System ROM Maps/OS RAM Module 136
Appendix B - System Variables Definition File 150
Appendix C - Application Development Library 158

| C-1  | 64-Column Mode                  |
| C-2  | 80-Column Mode                  |
| C-3  | 40-Column Mode                  |
| C-4  | Dual Screen Mode                |
| C-5  | Sprites                         |

Appendix D - 288

| D-1  | TS2068 PCB Assembly Drawing    |
| D-2  | TS2068 Parts List              |
| D-3  | TS2068 Schematic Diagram       |

Appendix E - Expansion Buss Comparisons 295
Appendix F - Modifications for EPROMs 296
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE NO.</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1-1</td>
<td>TS 2068 Block Diagram</td>
</tr>
<tr>
<td>1.1-2</td>
<td>Memory Configuration</td>
</tr>
<tr>
<td>1.1-3</td>
<td>RAM Mapping</td>
</tr>
<tr>
<td>1.1-4</td>
<td>System Initialization Flowchart</td>
</tr>
<tr>
<td>2.1.3-1</td>
<td>CPU Timing</td>
</tr>
<tr>
<td>2.1.3-2</td>
<td>Op Code Fetch Timing</td>
</tr>
<tr>
<td>2.1.3-3</td>
<td>Memory Read/Write Timing</td>
</tr>
<tr>
<td>2.1.3-4</td>
<td>I/O Read/Write Timing</td>
</tr>
<tr>
<td>2.1.3-5</td>
<td>Interrupt Request/Ack. Cycle</td>
</tr>
<tr>
<td>2.1.4-1</td>
<td>Rework for EPROM's</td>
</tr>
<tr>
<td>2.1.6-1</td>
<td>PSG Register Block Diagram</td>
</tr>
<tr>
<td>2.1.6-2</td>
<td>Tone Period Registers</td>
</tr>
<tr>
<td>2.1.6-3</td>
<td>Noise Period Register</td>
</tr>
<tr>
<td>2.1.6-4</td>
<td>Mixer Control-I/O Enable Reg.</td>
</tr>
<tr>
<td>2.1.6-5</td>
<td>D/A Converter Signal Generation</td>
</tr>
<tr>
<td>2.1.6-6</td>
<td>Amplitude Control Registers</td>
</tr>
<tr>
<td>2.1.6-7</td>
<td>Variable Amplitude Control</td>
</tr>
<tr>
<td>2.1.6-8</td>
<td>Envelope Period Registers</td>
</tr>
<tr>
<td>2.1.6-9</td>
<td>Envelope Shape/Cycle Control Reg.</td>
</tr>
<tr>
<td>2.1.6-10</td>
<td>Envelope Generator Output</td>
</tr>
<tr>
<td>2.1.6-11</td>
<td>Envelope Generator Output Detail</td>
</tr>
<tr>
<td>2.1.7-1</td>
<td>Joystick Port Operation</td>
</tr>
<tr>
<td>2.1.8-1</td>
<td>Bank Selection Logic</td>
</tr>
<tr>
<td>2.1.8-2</td>
<td>Video RAM Address Generation</td>
</tr>
<tr>
<td>2.1.9-1</td>
<td>Keyboard Schematic</td>
</tr>
<tr>
<td>2.1.10-1</td>
<td>Video RAM Data Organization</td>
</tr>
<tr>
<td>2.1.11-1</td>
<td>Composite Video Signal</td>
</tr>
<tr>
<td>2.4.1-1</td>
<td>P1 Mating Connector Mechanical Requirements</td>
</tr>
<tr>
<td>2.4.1-2</td>
<td>P1 Signal Layout</td>
</tr>
<tr>
<td>2.4.1-3</td>
<td>RGB Monitor Connection Schematic</td>
</tr>
<tr>
<td>2.4.2-1</td>
<td>J4 Mating Connector Mechanical Requirements</td>
</tr>
<tr>
<td>2.4.2-2</td>
<td>J4 Signal Layout</td>
</tr>
<tr>
<td>2.4.4-1</td>
<td>Joystick Connector</td>
</tr>
<tr>
<td>2.4.8-1</td>
<td>AC Adapter Plug</td>
</tr>
<tr>
<td>3.2.2-1</td>
<td>Ext.ROM Interruption Fielder</td>
</tr>
<tr>
<td>3.2.2-2</td>
<td>Ext.ROM Interface Routine</td>
</tr>
<tr>
<td>4.1.1-1</td>
<td>Keyboard Mode Control</td>
</tr>
<tr>
<td>4.1.1-2</td>
<td>Keyboard Support Routines Flowcharts</td>
</tr>
<tr>
<td>4.1.2-1</td>
<td>Standard Character Table Locations</td>
</tr>
<tr>
<td>4.1.2-2</td>
<td>Screen Row/Column Designations</td>
</tr>
<tr>
<td>4.2-1</td>
<td>Tape Header Formats</td>
</tr>
<tr>
<td>4.3-1</td>
<td>Joystick Data Format</td>
</tr>
</tbody>
</table>
LIST OF FIGURES
(continued)

<table>
<thead>
<tr>
<th>FIGURE NO.</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1-1</td>
<td>EPROM Cartridge Board Schematic</td>
</tr>
<tr>
<td>5.1-2</td>
<td>Ctdg.Bd.Component Side Artwork</td>
</tr>
<tr>
<td>5.1-3</td>
<td>Ctdg.Bd.Solder Side Artwork</td>
</tr>
<tr>
<td>5.1-4</td>
<td>EPROM Cartridge Bd. Solder Mask</td>
</tr>
<tr>
<td>6.5-1</td>
<td>GET STATUS Corrections</td>
</tr>
<tr>
<td>6.5-2</td>
<td>PUTWORD Corrections</td>
</tr>
<tr>
<td>6.5-3</td>
<td>BANK_ENABLE and RESTORE_STATUS Corrections</td>
</tr>
</tbody>
</table>

LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE NO.</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Z80A Control Signals</td>
</tr>
<tr>
<td>2.1.6-1</td>
<td>PSG I/O Enable Truth Table</td>
</tr>
<tr>
<td>2.1.6-2</td>
<td>PSG I/O Port Truth Table</td>
</tr>
<tr>
<td>2.1.8-1</td>
<td>SCLD I/O Pin Function Definitions</td>
</tr>
<tr>
<td>2.1.13-1</td>
<td>I/O Port Map</td>
</tr>
<tr>
<td>2.4.1-1</td>
<td>P1 Signal Definitions</td>
</tr>
<tr>
<td>2.4.1-2</td>
<td>P1 Signal Electrical Characteristics</td>
</tr>
<tr>
<td>2.4.2-1</td>
<td>J4 Signal Definitions</td>
</tr>
<tr>
<td>2.4.2-2</td>
<td>J4 Signal Electrical Characteristics</td>
</tr>
<tr>
<td>2.4.4-1</td>
<td>Joystick Connector Signal Assignment</td>
</tr>
<tr>
<td>3.2.2-1</td>
<td>Inputs to Video Mode Change Service</td>
</tr>
<tr>
<td>3.3.4-1</td>
<td>OS RAM Service Routines</td>
</tr>
<tr>
<td>3.3.4-2</td>
<td>Function Dispatcher Services</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

This manual provides detailed technical information on the Timex Sinclair 2068 Personal Color Computer. In conjunction with the TS2068 User Manual, it is intended to assist the reader in understanding the architecture, hardware and software features, programming techniques and I/O techniques pertaining to the TS2068.

1.1 TS2068 Overview

1.1.1 Hardware Overview

Figure 1.1-1 is a block diagram of the TS2068 showing the major functional components and their logical connections. These components are:

Control Logic - SCLD (Standard Cell Logic Device)
CPU - Z80A Microprocessor
RAM - 48K Random Access Memory
ROM - 24K System Read-Only Memory
(16K plus 8K Extension)

System Bus Connector
Cartridge Connector
Sound Generator/Speaker
Video Circuits
Cassette READ/WRITE
Joystick Connectors

The TS2068 Cartridge Connector provides for the plug-in of cartridges containing programmed ROM's with up to 64K of addressable memory. The full 64K is not normally utilized (e.g., due to need for access to RAM for the machine stack). See Section 5.1 for details.

Figure 1.1-2 shows the standard TS2068 memory configuration comprised of the Home Bank, the ROM Extension Bank and the Dock (Cartridge) Bank. This memory is selectable as eight 8K 'chunks' with the Home Bank being enabled by default, i.e., any chunk not selected in the Extension or Dock Bank is automatically enabled in the Home Bank.

Memory selection and I/O are controlled via the I/O ports. These topics are covered in detail in later sections.
1.1.2 System Software Overview

The TS2068 System Software resides in the Home ROM, the Extension ROM, and dedicated RAM. It supports the following functions:

- System Initialization

- BASIC Interpreter (including BASIC cartridge support)

- BASIC I/O for Standard Peripherals
  - keyboard
  - video screen
  - 2040 32-col. dot matrix printer
  - cassette tape
  - joysticks
  - software generated sound (BEEP)
  - programmable sound chip (SOUND)

- Video Mode Change Service

- Interruption Servicing (Z80 Int. Mode 1)

- Bank Switching/Data Transfer Services

- Function Dispatcher (provides access to selected system routines via a Service Code input)

In addition, portions of the Home Bank RAM are used for the machine stack, the BASIC system variables, the Printer Buffer and the Display Files. Figure 1.1-3 shows the standard mapping of the Home Bank RAM and the mapping necessary when the second display file is to be used with the BASIC interpreter still functional. The Video Mode Change Service routine makes these memory modifications. Note that there is no direct support of the second display file via BASIC or in the system ROM I/O routines.

Figure 1.1-4 is a Flowchart of the System Initialization process.
FIGURE 1.1-3

STANDARD MAPPING OF

HOME BANK RAM

A) 1 Display File

B) 2 Display File
FIGURE 1.1-4
SYSTEM INITIALIZATION

POWER ON

HOME ROM
SET MAX.
ADDRESS = 64K

EXTENSION ROM
SET MAX.
ADDRESS = 64K

Switch to Home Rom

"NEW"

SET MAX. ADDRESS = (RAMTOP)

SET "NEW" FLAG

SAVE (P-RAMT),
(UDG), (PIP) &
(RASP)

RESTORE (P-RAMT),
(UDG), (PIP) &
(RASP)

INITIALIZE RAM

BUILD SYSCON TABLE

LROS?

Y

To LROS
Start ADRS.

N

AROS?

Y

To Basic
Interpreter

N

AUTO-START?

N

To BASIC
AROS
Support Code

Y

"NEW"

N

INITIALIZE P-RAMPT,
UDG, PIP, &
RASP

INITIALIZE SYSTEM
VARIABLES

INITIALIZE MACHINE
STACK

INITIALIZE
CHANS/STREAMS
& PRINTER BUFFER

CLEAR SCREEN
& OUTPUT
© MSG.

COPY OS RAM
CODE TO CHUNK 3

Switch to Extension
1.1.3 Cartridge Software Overview

The TS2068 supports two basic types of Cartridge or ROM-Oriented Software designated as LROS (Language ROM-ORiented Software) and AROS (Application ROM-Oriented Software) which plug into the cartridge connector. They are identified via overhead bytes at Location 0 for an LROS or 32768 (8000H) for an AROS. The fundamental difference is that an LROS contains Z80 machine code in memory chunk 0 and is in total control of the TS2068 hardware including the RESTART implementation and Interruption Mode setting and handling, while an AROS is dependent on the System ROM or an LROS for these functions if needed. An AROS written in BASIC, which may also include machine code accessed via the USR function, is supported from the System ROM BASIC Interpreter and is mapped beginning in memory chunk 4. An AROS may also be written entirely in Z80 machine code. An AROS written in any other high-level language would require an LROS supporting that language and would have to be integrated with the LROS in a single cartridge.

See Sections 3.2.1.2, BASIC AROS Support and 5.1, Cartridge Software/Hardware, for additional details.
2.0 HARDWARE GUIDE

2.1 Description of Major Hardware Functions

Figure 1.1-1 shows a simplified block diagram of the TS2068. The following functional units are described in the following sections:

<table>
<thead>
<tr>
<th>SECTION</th>
<th>FUNCTIONAL UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1</td>
<td>AC Adapter</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Voltage Regulation</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Z-80A CPU</td>
</tr>
<tr>
<td>2.1.3.1</td>
<td>Address Bus</td>
</tr>
<tr>
<td>2.1.3.2</td>
<td>Data Bus</td>
</tr>
<tr>
<td>2.1.3.3</td>
<td>Control Signals</td>
</tr>
<tr>
<td>2.1.3.4</td>
<td>OP Code Fetch</td>
</tr>
<tr>
<td>2.1.3.5</td>
<td>Memory READ/WRITE</td>
</tr>
<tr>
<td>2.1.3.6</td>
<td>I/O READ/WRITE</td>
</tr>
<tr>
<td>2.1.3.7</td>
<td>Maskable Interruption</td>
</tr>
<tr>
<td>2.1.3.8</td>
<td>Non-Maskable Interruption (NMI)</td>
</tr>
<tr>
<td>2.1.4</td>
<td>ROM</td>
</tr>
<tr>
<td>2.1.5</td>
<td>32K RAM</td>
</tr>
<tr>
<td>2.1.6</td>
<td>Sound Generator</td>
</tr>
<tr>
<td>2.1.7</td>
<td>Joystick Port</td>
</tr>
<tr>
<td>2.1.8</td>
<td>Control Logic</td>
</tr>
<tr>
<td>2.1.8.1</td>
<td>Bank Selection Logic</td>
</tr>
<tr>
<td>2.1.8.2</td>
<td>Z80 Clock Generator</td>
</tr>
<tr>
<td>2.1.8.3</td>
<td>Display File Access</td>
</tr>
<tr>
<td>2.1.8.4</td>
<td>Interruption Generation</td>
</tr>
<tr>
<td>2.1.9</td>
<td>Keyboard</td>
</tr>
<tr>
<td>2.1.10</td>
<td>16K Video Display RAM</td>
</tr>
<tr>
<td>2.1.11</td>
<td>Video Generation</td>
</tr>
<tr>
<td>2.1.11.1</td>
<td>Composite Video</td>
</tr>
<tr>
<td>2.1.11.2</td>
<td>RF Modulator</td>
</tr>
<tr>
<td>2.1.12</td>
<td>Cassette I/O</td>
</tr>
<tr>
<td>2.1.13</td>
<td>Port Map</td>
</tr>
</tbody>
</table>
2.1.1 AC Adapter

The AC Adapter transforms 117V AC (Nominal) to filtered DC via a step down transformer, full-wave bridge rectifier, and filter capacitor to supply from 14 to 25 volts at 1 amp over the AC voltage variation range of 105 to 130 V AC. Transformer isolation exceeds 1500 volts.

2.1.2 Voltage Regulation

Unregulated DC from the AC Adapter is supplied for regulation through a bi-filar toroidal inductor which reduces conducted line emanation for FCC compliance and through the power-ON/OFF switch located on the left side of the TS2068. This switch voltage is supplied to the System Bus Connector (see Section 2.4) and for regulation to the +12 V regulator and the +5 V regulator. Characteristics are as follows:

<table>
<thead>
<tr>
<th>SUPPLY</th>
<th>VOLTAGE RANGE</th>
<th>CURRENT RANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>4.75 - 5.25V</td>
<td>200ma - 1.0 A</td>
</tr>
<tr>
<td>12V</td>
<td>11.5 - 12.5V</td>
<td>20ma - 100ma</td>
</tr>
</tbody>
</table>

+20v  
+5v   
+12v  
GND  .
The 12V regulator is a 78L12 series regulator while the 5V regulator is a switching supply utilizing the 78S40 circuit.

2.1.3 Z-80A CPU

The Z-80A CPU of the TS2068 operates at a clock frequency of 3.53 MHz. Primary features of this CPU are:

- 158 instructions
- Dual register set
- Two index registers
- On-chip refresh logic

The Z-80 CPU executes instructions by proceeding through a sequence of operations that include:

a) instruction Op code fetching
b) READ or WRITE memory
c) READ or WRITE I/O
d) Acknowledge an interruption

The basic clock period is referred to as a T time or state and three or more T states make up a machine cycle. In the TS2068, each T-time is approximately 283 nanoseconds (2.83 X 10^-7 seconds). Figure 2.1.3-1 illustrates the basic timing.

**FIGURE 2.1.3-1**

**BASIC CPU TIMING EXAMPLE**

![Diagram of basic CPU timing example]
2.1.3.1 Address Bus

Output from the Z-80 are 16-bits of address information, A0 - A15, which are high-active tri-state signals and address for memory data and I/O device exchanges.

2.1.3.2 Data Bus

These input/output signals from the Z-80, D0 - D7, constitute an 8-bit bi-directional, high-active, tri-state data bus used for data exchanges with memory and I/O devices.

2.1.3.3 Control Bus

Associated with the Z-80 are 13 control lines which are provided by or used by the Z-80 to control system operation. These signals are detailed in Table 2-1.

2.1.3.4 Op Code Fetch

The timing during an M1 cycle (Op Code Fetch) is shown in Figure 2.1.3-2. At the beginning of the M1 cycle the PC (Program Counter) is placed onto the address bus, then one-half clock time later the MREQ signal goes active indicating that the memory address is stable. The RD signal is activated to indicate that memory read data should be gated onto the data bus. At the rising clock edge during the T3 state, the CPU samples the data on the data bus and deactivates the RD and MREQ signals. During the T3 and T4 states, the CPU decodes and executes the fetched instruction and the CPU places on the lower 7 bits of the address bus a memory refresh address and activates the RFSH signal indicating a refresh read is to begin when MREQ is activated.

2.1.3.5 Memory READ/WRITE

Memory read or write cycles other than Op Code Fetches are 3 clock periods long with the MREQ and RD signals used as in the fetch cycle. During a write cycle the WR signal is activated when the write data is stable on the data bus. The address and data bus contents remain stable for one-half T state after the WR signal goes active. Figure 2.1.3-3 illustrates.
FIGURE 2.1.3-2

INSTRUCTION OP CODE FETCH

FIGURE 2.1.3-3

MEMORY READ OR WRITE CYCLES
2.1.3.6 I/O READ/WRITE

During I/O operations RORQ and RD or WR are activated on the leading edge of the T2 clock and a single WAIT state is automatically inserted as illustrated in Figure 2.1.3-4. The RD and WR signals are used to enable data from the addressed port onto the data bus and to, on the rising edge of WR, clock data to the I/O port, respectively. Note that external I/O may stretch the activation period of the WAIT line to extend the I/O cycles.

FIGURE 2.1.3-4

INPUT OR OUTPUT CYCLES

*Inserted by Z80 CPU
2.1.3.7 Maskable Interruption

When enabled by software, when BUSREQ is not active and when INT is active at the rising edge of the last clock of any instruction, a maskable interruption occurs during the subsequent MI cycle, as illustrated in Figure 2.1.3-5.

**FIGURE 2.1.3-5**

**INTERRUPT REQUEST/ACKNOWLEDGE CYCLE**

![Diagram of interrupt request/acknowledge cycle]

In Interruption Mode 0, the interrupting I/O device places any instruction on the data bus during the IORQ activation and the CPU executes that instruction. The RESTART instruction is commonly used for this purpose. RESET will automatically set Interruption Mode 0.

In Interruption Mode 1, the CPU executes a RESTART to Location 0038H. This is the mode normally used by the TS 2068 software.

In Interruption Mode 2, the CPU concatenates the 8-bit argument, which must be a 2-byte boundary address, with the 8-bit I Register contents to form a 16-bit pointer to a memory table entry containing the 16-bit service routine address - the first byte in the table.
being the low order portion of the address. Once the interrupting device supplies the lower portion of the pointer (for concatenation), the CPU automatically pushes the PC onto the stack, obtains the starting address from the table, and does a jump to that address. 19 clock periods are required to complete this sequence.

2.1.3.8 Non-Maskable Interruption (NMI)

A pulse on the NMI input to the Z80 sets the internal latch which is tested by the CPU at the end of each instruction. The NMI has priority over the maskable interruption and its response is identical to the maskable interruption (Mode 1) except that the call location is 0066H instead of 0038H.

NOTES: 1. The NMI is not used by the TS 2068.

2. Comments in the ROM listing claiming to "mask the NMI" via the DI instruction are incorrect. The DI instruction masks only the maskable interruption.
<table>
<thead>
<tr>
<th>ACRONYM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Machine Cycle 1 - Output, active low. This signal indicates that the current machine cycle is the OP code fetch cycle. During execution of instructions having a 2-byte OP code, this signal is generated as each OP code byte is fetched. M is also used with IORQ to indicate an interrupt acknowledge cycle.</td>
</tr>
<tr>
<td>MREQ</td>
<td>Memory Request - Tri-state output, active low. This signal indicates that the Address Bus holds a valid address for a memory read or write operation.</td>
</tr>
<tr>
<td>IORQ</td>
<td>I/O Request - Tri-state output, active low. This signal indicates that the lower half of the Address Bus holds a valid I/O address for an I/O read or write operation. This signal is also used with M in connection with acknowledging an interruption, indicating that an interrupt response vector can be placed on the data bus. I/O operations never occur during M time.</td>
</tr>
<tr>
<td>RD</td>
<td>Memory Read - Tri-state output, active low. This signal indicates that the CPU wants to read data from memory or an I/O device. The addressed memory or device should use this signal to gate the requested data onto the CPU data bus.</td>
</tr>
<tr>
<td>WR</td>
<td>Memory Write - Tri-state output, active low. This signal indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.</td>
</tr>
<tr>
<td>RFSH</td>
<td>Refresh - Output, active low. This signal indicates that the lower 7 bits of the Address Bus contain a refresh address for dynamic memories and the current MREQ signal should be used to do a refresh read to all dynamic memories. A7 is a logic zero and the upper 8 bits of the Address Bus contain the contents of the I Register.</td>
</tr>
<tr>
<td>ACRONYM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>CPU CONTROL</td>
<td></td>
</tr>
<tr>
<td>HALT</td>
<td>Halt State - Output, active low. This signal indicates that the CPU has executed a HALT instruction. CPU operations are suspended until a Non-Maskable or a Maskable Interruption (with the mask enabled) occurs. While halted, the CPU executes NOP's to maintain memory refresh.</td>
</tr>
<tr>
<td>WAIT</td>
<td>Wait - Input, active low. This signal indicates to the CPU that the addressed memory or I/O device is not ready for a data transfer. The CPU will continue to enter wait states as long as this signal is active. This allows for synchronization of the CPU to external devices of varying speeds.</td>
</tr>
<tr>
<td>INT</td>
<td>Interrupt Request - Input, active low. This signal is generated by external devices and is honored at the end of the current instruction if the interrupt is not masked by the software and if the BUSRQ signal is not active. When the CPU accepts the interruption, an acknowledge signal is sent out at the beginning of the next instruction cycle (TORQ at MI time). There are three interruption modes selectable by the software.</td>
</tr>
<tr>
<td>NMI</td>
<td>Non-Maskable Interruption - Input, negative edge triggered. This signal has a higher priority than INT and is always recognized at the end of the current instruction (cannot be masked). The CPU is forced to restart to location 0066H with the program counter saved in the external stack. NOTE: The NMI is not used in the TS2068 ROM software design.</td>
</tr>
</tbody>
</table>
### TABLE 2-1

#### Z80 CONTROL SIGNALS

(continued)

<table>
<thead>
<tr>
<th>ACRONYM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RESET</strong></td>
<td>Reset - Input, active low. This signal forces the program counter to zero and initializes the CPU. Address and data buses go to their high impedance state and control output signals to their inactive state. No refresh occurs. Initialization includes: Disable the interrupt enable flip-flop and set Register I, Register R and the Interrupt Mode all to Zero.</td>
</tr>
</tbody>
</table>

**CPU BUS CONTROL**

| BUSRQ | Bus Request - Input, active low. This signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state permitting other devices to control these buses. The CPU sets these buses to a high impedance state at the termination of the current Machine cycle. |

| BUSAK | Bus Acknowledge - Output, active low. This signal is used to indicate to the requesting device that the CPU has set its address, data and control bus signals to a high impedance state in response to BUSRQ. |

#### Figure 2.1.4-1

REWORK TO REPLACE ROM's with EPROM's

[Diagram of rework process]

![Diagram](image-url)
2.1.4 ROM

The system includes both a 16K byte ROM and an 8K byte ROM mapped into the address space as shown below.

Section 2.1.8.1 describes the selection of the Home Bank and Expansion Bank via the control logic.

The devices involved are a 23128 and a 2364 for the 16K byte (128K-bit) and the 8K byte (64K-bit) ROM's respectively. Direct replacement of these devices with 27128 and 2764 EPROM's is not possible since pins 1 and 27 must be maintained in the high state for those devices (see schematic in Section 2.2). To replace U16 and U20 with 27128 and 2764 EPROM's requires the rework shown in Figure 2.1.4-1.

(1) Cut input to pin 27 on each chip.

(2) Wire +5V to pins 1 and 27 on each chip to pull high.

If U20 is to be a 27128, then replace the RD input to pin 26 with address A13 from pin 26 on U16.

2.1.5 32K RAM (Address 8000-FFFFH)

The upper 32K of RAM is composed of four 200ns 4416's (16K x 4 dynamic RAMs).
2.1.6 Sound Generator

The Programmable Sound Generator (GI 8912) is accessed via Ports OF5H (Address) and OF6H (Data). The basic registers in the PSG which produce the programmed sounds include:

Tone Generators: Produce the basic square wave tone frequencies for each channel (A, B, C).

Noise Generator: Produces a frequency modulated pseudo-random pulse width square wave output.

Mixers: Combine the outputs of the Tone Generators and the Noise Generator. One for each channel (A, B, C).

Amplitude Control: Provides the D/A Converters with either a fixed or variable amplitude pattern. The fixed amplitude is under direct CPU control; the variable amplitude is accomplished by using the output of the Envelope Generator.

Envelope Generator: Produces an envelope pattern which can be used to amplitude modulate the output of each Mixer.

D/A Converters: The three D/A Converters each produce up to a 16-level output signal as determined by the Amplitude Control.

An additional register is shown in the PSG Block Diagram (Figure 2.1.6-1) which has nothing directly to do with the production of sound -- this is the I/O Port (A). Data to/from the CPU may be read/written to/from the 8-bit I/O Port without affecting any other function of the PSG. The TS 2068 uses the I/O Port to access the joysticks.

2.1.6.1 Tone Generator Control (Registers RO-R5)

The frequency of each square wave generated by the three Tone Generators (one each for Channels A, B, and C) is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 12-bit Tone Period value. Each 12-bit value is obtained in the PSG by combining the contents of the relative Coarse and Fine Tune registers, as illustrated by Figure 2.1.6-2.

Note that the 12-bit value programmed in the combined Coarse and Fine Tune registers is a period value -- the higher the value in the registers, the lower the resultant tone frequency.

Note also that due to the design technique used in the Tone Period countdown, the lowest period value is 0000000000001 (divide by 1) and the highest period value is 111111111111 (divide by 4095).
FIGURE 2.1.6-1

PSG REGISTER BLOCK DIAGRAM

<table>
<thead>
<tr>
<th>REGISTER</th>
<th>DEC</th>
<th>HEX</th>
<th>OCT</th>
<th>BIT</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>IN/OUT</th>
<th>NOISE</th>
<th>TUNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8 Bit Fine Tune</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>R1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4 Bit Coarse Tune</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>R2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>8 Bit Fine Tune</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>R3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4 Bit Coarse Tune</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>R4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8 Bit Fine Tune</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>R5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4 Bit Coarse Tune</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>R6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5 Bit Period Control</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>R7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>Enable</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>R8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>Channel A Amplitude</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>R9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>Channel B Amplitude</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>R10</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>Channel C Amplitude</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>R11</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>Envelope</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>R12</td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>Period</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>R13</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>Shape/Cycle</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>R14</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>I/O Port A</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>R15</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>Data Store</td>
<td>//</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 2.1.6-2

12-BIT TONE PERIOD (TP) TO TONE GENERATOR

COARSE TUNE REGISTER  | CHANNEL  | FINE TUNE REGISTER
----|-----|----
R1 | A   | R0
R3 | B   | R2
R5 | C   | R4

B7 B6 B5 B4 B3 B2 B1 B0

NOT USED

B7 B6 B5 B4 B3 B2 B1 B0

TP11 TP10 TP9 TP8 TP7 TP6 TP5 TP4 TP3 TP2 TP1 TP0

20
2.1.6.1 (continued)

The equations describing the relationship between the desired output tone frequency and the input clock frequency and Tone Period value are:

\[
\begin{align*}
(a) \quad FT &= \frac{fCLOCK}{16TP} \\
(b) \quad TP &= 256CT + FT \\
\end{align*}
\]

Where:

\[
\begin{align*}
ft &= \text{Desired tone frequency} \\
fCLOCK &= \text{Input clock frequency} \\
TP &= \text{Decimal equivalent of the Tone Period bits TP11 to TPO} \\
CT &= \text{Decimal equivalent of the Coarse Tune register bits B3 to B0 (TP11 to TP8)} \\
FT &= \text{Decimal equivalent of the Fine Tune register bits B7 to B0 (TP7 to TPO)} \\
\end{align*}
\]

From the above equations, it can be seen that the tone frequency can range from a low of:

\[
fCLOCK/65520 \quad(\text{wherein TP} = 4095) \\
\]

to a high of:

\[
fCLOCK/16 \quad(\text{wherein TP} = 1). \\
\]

The TS 2068 uses a 1.76475 MHZ input clock, so it can produce a range of 26.9 Hz to 110 kHz.
2.1.6.1 (continued)

To calculate the values for the contents of the Tone Period Coarse and Fine Tune registers, given the input clock and the desired output tone frequencies, we simply rearrange the above equations, yielding:

(a) \( TP = \frac{f_{\text{CLOCK}}}{16 \text{ ft}} \)  
(b) \( CT + \frac{FT}{10} = \frac{TP}{256} \)

Example 1: \( f_T = 1 \text{ kHZ} \)  \( f_{\text{CLOCK}} = 1.76475 \text{ MHz} \)

\[
TP = \frac{1.76475 \times 10}{16(1 \times 10)} = 110.3
\]

Substituting this result into equation (b):

\[
\frac{CT}{10} + \frac{FT}{10} = \frac{110.3}{256} \]

resulting in:

\[
CT = 0 \quad 0000 \text{ (B3-B0)}
\]

\[
FT = 110 \quad 01101110 \text{ (B7-B0)}
\]

Example 2: \( f_T = 100 \text{ Hz} \)  \( f_{\text{CLOCK}} = 1.76475 \text{ MHz} \)

\[
TP = \frac{1.76475 \times 10}{16(1 \times 10)} = 1103
\]

Substituting this result into equation (b):

\[
\frac{CT}{10} + \frac{FT}{10} = \frac{1103}{256} = 4 + \frac{79}{256}
\]

resulting in:

\[
CT = 4 \quad 0100 \text{ (B3-B0)}
\]

\[
FT = 79 \quad 01001111 \text{ (B7-B0)}
\]
2.1.6.2 Noise Generator Control (Register R6)

The frequency of the noise source is obtained in the PSG by first counting down the input clock by 16, then by further counting down the result by the programmed 5-bit Noise Period value. This 5-bit value consists of the lower 5 bits (B4-B0) of Register R6 as illustrated by Figure 2.1.6-3.

FIGURE 2.1.6-3

NOISE PERIOD REGISTER R6

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT USED</td>
<td>5-BIT NOISE PERIOD (NP)</td>
<td>TO NOISE GENERATOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the 5-bit value in R6 is a period value -- the higher the value in the register, the lower the resultant noise frequency. Note also that, as with the Tone Period, the lowest period value is 00001 (divide by 1); the highest period value is 11111 (divide by 31).

\[
f_N = \frac{f_{\text{CLOCK}}}{16 \times \text{NP}}
\]

Where:

- \( f_N \) = Desired noise frequency
- \( f_{\text{CLOCK}} \) = Input clock frequency
- \( \text{NP} \) = Decimal equivalent of the Noise Period register bits B4-B0.

From the above equation it can be seen that the noise frequency can range from a low of \( f_{\text{CLOCK}}/496 \) (wherein \( \text{NP} = 31 \))

\[ \frac{10}{10} \]

to a high of \( f_{\text{CLOCK}}/16 \) (wherein \( \text{NP} = 1 \)). Using a 1.76475 MHz

\[ \frac{10}{10} \]
clock, for example, would produce a range of noise frequencies from 3.6 kHz to 110.3 kHz.

To calculate the value for the contents of the Noise Period register, given the input clock and the desired output noise frequencies, we simply rearrange the above equation, yielding:

\[
\text{NP} = \frac{f_{\text{CLOCK}}}{16f_N} \frac{10}{10}
\]
2.1.6.3 Mixer Control I/O Enable (Register R7)

Register 7 is a multi-function Enable register which controls the three Noise/Tone Mixers and the two general purpose I/O ports.

The Mixers, as previously described, combine the noise and tone frequencies for each of the three channels. The determination of combining neither/either/both noise and tone frequencies on each channel is made by the state of bits B5 thru B0 of R7.

The direction (input or output) of the two general purpose I/O ports (IOA and IOB) is determined by the state of bits B7 and B6 of R7. Note that in the TS 2068 there is no second I/O Port B.

These functions are illustrated by Figure 2.1.6-4 and Tables 2.1.6-1 and 2.1.6-2 below.

**FIGURE 2.1.6-4**

**MIXER CONTROL - I/O ENABLE REGISTER R7**

**TABLE 2.1.6-1**

**I/O ENABLE TRUTH TABLE**

<table>
<thead>
<tr>
<th>R7 BITS</th>
<th>Noise Enabled on Channel</th>
<th>R7 BITS</th>
<th>Tone Enabled on Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>B5 B4 B3</td>
<td>Noise Enabled on Channel</td>
<td>B2 B1 B0</td>
<td>Tone Enabled on Channel</td>
</tr>
<tr>
<td>0 0 0</td>
<td>C B A</td>
<td>0 0 0</td>
<td>C B A</td>
</tr>
<tr>
<td>0 0 1</td>
<td>C B -</td>
<td>0 0 1</td>
<td>C B -</td>
</tr>
<tr>
<td>0 1 0</td>
<td>C - A</td>
<td>0 1 0</td>
<td>C - A</td>
</tr>
<tr>
<td>0 1 1</td>
<td>C - -</td>
<td>0 1 1</td>
<td>C - -</td>
</tr>
<tr>
<td>1 0 0</td>
<td>- B A</td>
<td>1 0 0</td>
<td>- B A</td>
</tr>
<tr>
<td>1 0 1</td>
<td>- B -</td>
<td>1 0 1</td>
<td>- B -</td>
</tr>
<tr>
<td>1 1 0</td>
<td>- - A</td>
<td>1 1 0</td>
<td>- - A</td>
</tr>
<tr>
<td>1 1 1</td>
<td>- - -</td>
<td>1 1 1</td>
<td>- - -</td>
</tr>
</tbody>
</table>
TABLE 2.1.6-2
I/O PORT TRUTH TABLE

<table>
<thead>
<tr>
<th>R7 BITS</th>
<th>I/O Port Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>B6</td>
<td>IOA</td>
</tr>
<tr>
<td>0</td>
<td>Input</td>
</tr>
<tr>
<td>1</td>
<td>Output</td>
</tr>
</tbody>
</table>

NOTE
Disabling noise and tone does not turn off a channel. Turning a channel off can only be accomplished by writing all zeroes into the corresponding Amplitude Control register, R8, R9 or R10 (refer to Paragraph 2.1.6.4).

2.1.6.4 Amplitude Control (Registers R8, R9, R10)
The amplitudes of the signals generated by each of the three D/A Converters (one each for Channels A, B, and C) is determined by the contents of the lower 5 bits (B4-B0) of Registers R8, R9 and R10 as illustrated by Figure 2.1.6-5.

FIGURE 2.1.6-5
D/A CONVERTER SIGNAL GENERATION

<table>
<thead>
<tr>
<th>AMPLITUDE CONTROL REGISTER #</th>
<th>CHANNEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8</td>
<td>A</td>
</tr>
<tr>
<td>R9</td>
<td>B</td>
</tr>
<tr>
<td>R10</td>
<td>C</td>
</tr>
</tbody>
</table>

B7 B6 B5 B4 B3 B2 B1 B0
NOT USED
M
L3 L2 L1 L0
Amplitude 'Mode'
4-Bit Fixed Amplitude Level

25
2.1.6.4 (continued)

The amplitude 'mode' (Bit M) selects either fixed level amplitude (M=0) or variable level amplitude (M=1). It follows then that Bits L3-L0 defining the value of a 'fixed' level amplitude, are only active when M=0. When fixed level amplitude is selected, it is 'fixed' only in the sense that the amplitude level is under the direct control of the system processor (via bits L3-L0). Varying the amplitude when in this 'fixed' amplitude mode requires in each instance the direct intervention of the system processor via an address latch/write data sequence to modify the L3-L0 data.

When M=1 (select 'variable' level amplitudes), the amplitude of each channel is determined by the envelope pattern as defined by the Envelope Generator's 4-bit output E3-E0 (refer to Paragraph 2.1.6.5).

The amplitude 'mode' (Bit M) can be thought of as an 'envelope enable' bit, i.e. when M=0 the envelope is not used, and when M=1 the envelope is enabled.

Figure 2.1.6-6 illustrates all combination of the 5-bit Amplitude Control.

FIGURE 2.1.6-6

AMPLITUDE CONTROL REGISTERS

<table>
<thead>
<tr>
<th>AMPLITUDE CONTROL REGISTER #</th>
<th>CHANNEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>R8</td>
<td>A</td>
</tr>
<tr>
<td>R9</td>
<td>B</td>
</tr>
<tr>
<td>R10</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B7 B6 B5 B4 B3 B2 B1 B0</th>
<th>Amplitude Control Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT USED M L3 L2 L1 L0</td>
<td></td>
</tr>
<tr>
<td>0 0 0 0 0 *0 0 0 0</td>
<td>The amplitude is fixed at 1 of 16 levels as determined by L3-L0.</td>
</tr>
<tr>
<td>0 .. .. .. ..</td>
<td></td>
</tr>
<tr>
<td>0 .. .. .. ..</td>
<td></td>
</tr>
<tr>
<td>0 .. .. .. ..</td>
<td></td>
</tr>
<tr>
<td>0 .. .. .. ..</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>1 X X X X E3 E2 E1 E0</td>
<td>The amplitude is variable at 16 levels as determined by the output of the Envelope Gen.</td>
</tr>
</tbody>
</table>

(X=Don't Care)

*The all zeros code is used to turn a channel "off".
2.1.6.4 (continued)

Figure 2.1.6-7 graphically illustrates a selection of variable level (envelope-controlled) amplitude where the 16 levels directly reflect the output of the Envelope Generator. A fixed level amplitude would correspond to only one of the levels shown, with the level directly determined by the decimal equivalent of Bits L3-L0.

FIGURE 2.1.6-7

VARIABLE AMPLITUDE CONTROL (M=1)

2.1.6.5 Envelope Generator Control (Registers R11, R12, R13)

To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG; first, it is possible to vary the frequency of the envelope using registers R11 and R12; and second, the relative shape and cycle pattern of the envelope can be varied using register R13. The following paragraphs explain the details of the envelope control functions, describing first the envelope period control and then the envelope shape/cycle control.

2.1.6.5.1 Envelope Period Control (Registers R11, R12)

The frequency of the envelope is obtained in the PSG by first counting down the input clock by 256, then by further counting down the result by the programmed 16-bit Envelope Period value. This 16-bit value is obtained in the PSG by combining the contents of the Envelope Coarse and Fine Tune registers, as illustrated by Figure 2.1.6-8.
FIGURE 2.1.6-8

16-BIT ENVELOPE PERIOD (EP) TO
ENVELOPE GENERATOR

ENVELOPE
COARSE TUNE
REGISTER R12

ENVELOPE
FINE TUNE
REGISTER R11

\[ \begin{align*}
B7 & \ B6 \ B5 \ B4 \ B3 \ B2 \ B1 \ B0 \\
EPI5 & \ EPI4 \ EPI3 \ EPI2 \ EPI1 \ EPI0 \ EP9 \ EP8 \ EP7 \ EP6 \ EP5 \ EP4 \ EP3 \ EP2 \ EP1 \ EP0
\end{align*} \]

Note that the 16-bit value programmed in the combined Coarse and
Fine Tune registers is a period value - the higher the value in
the registers, the lower the resultant envelope frequency.

Note also that, as with the Tone Period, the lowest period value
is 0000000000000001 (divide by 1); the highest period value is
1111111111111111 (divide by 65,535).

2 10

The envelope frequency equations are:

(a) \[ fe = \frac{fCLOCK}{256 \ EP} \]
(b) \[ EP = 256 \ CT + FT \]

\[ \begin{align*}
10 & 10 & 10
\end{align*} \]

Where:

\[ \begin{align*}
fe & = \text{Desired envelope frequency} \\
fCLOCK &= \text{Input clock frequency} \\
EP & = \text{Decimal equivalent of the Envelope} \\
CT & = \text{Decimal equivalent of the Coarse} \\
10 & = \text{Period bits EP15-EP0} \\
10 & = \text{Tune register bits B7-B0 (EP15-EP8)} \\
10 & = \text{Decimal equivalent of the Fine} \\
10 & = \text{Tune register bits B7-B0 (EP7-EP0)}
\end{align*} \]

From the above equation it can be seen that the envelope frequency can
range from a low of \[ fCLOCK/16,766,960 \] (wherein EP = 65,535)
\[ \begin{align*}
10 & 10 & 10
\end{align*} \]

to a high of \[ fCLOCK/256 \] (wherein EP = 1). Using a 1.76475 MHz clock,
\[ \begin{align*}
10 & 10 & 10
\end{align*} \]

for example, would produce a range of envelope frequencies from 0.105
Hz to 6893.6 Hz.
To calculate the values for the contents of the Envelope Period Coarse and Fine Tune registers, given the input clock and the desired envelope frequencies, we rearrange the above equations, yielding:

(a) \[ \frac{\text{EP}}{10} = \frac{\text{fCLOCK}}{256\text{fE}} \]

(b) \[ \frac{\text{CT} + \text{FT}}{10} = \frac{\text{EP}}{256} = \frac{10}{256} \]

Example:

\[ \text{fE} = 0.5 \text{ Hz} \]
\[ \text{fCLOCK} = 1.76475 \text{ MHz} \]

\[ \frac{\text{EP}}{10} = \frac{1.76475 \times 10}{256(0.5)} = 13787 \]

Substituting this result into equation (b):

\[ \frac{\text{CT} + \text{FT}}{10} = \frac{13787}{256} = \frac{53 + 219}{256} \]

\[ \text{CT} = 53 = 00110101 \quad \text{(B7-B0)} \]
\[ 2 \]

\[ \text{FT} = 219 = 11011011 \quad \text{(B7-B0)} \]
\[ 2 \]

2.1.6.5.2 Envelope Shape/Cycle Control (Register R13)

The Envelope Generator further counts down the envelope frequency by 16, producing a 16-state per cycle envelope pattern as defined by its 4-bit counter output, E3-E0. The particular shape and cycle pattern of any desired envelope is accomplished by controlling the count pattern (count up/count down) of the 4-bit counter and by defining a single-cycle or repeat-cycle pattern. This envelope shape/cycle control is contained in the lower 4 bits (B3-B0) of register R13. Each of these 4 bits controls a function in the envelope generator, as illustrated in Figure 2.1.6-9.
The definition of each function is as follows:

**HOLD**  When set to logic "1", limits the envelope to one cycle, holding the last count of the envelope counter (E3-E0 = either 0000 or 1111, depending on whether the envelope counter was in countdown or countup mode respectively.

**ALTERNATE**  When set to logic "1", the envelope counter reverses count direction (up-down) after each cycle.

**NOTE**  When both the Hold bit and the Alternate bit are ones, the envelope counter is reset to its initial count before holding.

**ATTACK**  When set to logic "1", the envelope counter will count up (attack) from E3-E0 = 0000 to E3-E0 = 1111; when set to logic "0", the envelope counter will count down (decay) from 1111 to 0000.

**CONTINUE**  When set to logic "1", the cycle pattern will be as defined by the Hold bit; when set to logic "0", the envelope generator will reset to 0000 after one cycle and hold at that count.
To further describe the above functions, numerous charts of the binary count sequence of E3-E0 could be used, showing each combination of Hold, Alternate, Attack and Continue. However, since these outputs are used (when selected by the Amplitude Control registers) to amplitude modulate the output of the Mixers, a better understanding of their effect can be accomplished via a graphic representation of their value for each condition selected, as illustrated in Figures 2.1.6-10 and 2.1.6-11.

FIGURE 2.1.6-10

ENVELOPE GENERATOR OUTPUT
2.1.6.6 I/O Port Data Store (Register R14)

Register R14 functions as an intermediate data storage register between the PSG/CPU data bus (DA7-DA0) and the I/O Port (IOA7-IOA0). This port is available for reading the joysticks. Using register R14 for the transfer of I/O data has no effect at all on sound generation.

To output data from the CPU bus to a peripheral device connected to I/O Port A would require the following steps:

1. Latch address R7 (select Enable register)
2. Write data to PSG (setting R7, B6=1)
3. Latch address R14 (select IOA register)
4. Write data to PSG (data to be output on I/O Port A)

To input data from I/O Port A to the CPU bus would require the following:

1. Latch address R7 (select Enable register)
2. Write data to PSG (setting R7, B6=0)
3. Latch address R14 (select IOA register)
4. Read data from PSG (data from I/O Port A)

Note that once loaded with data in the output mode, the data will remain on the I/O port until changed either by loading different data, by applying a reset (grounding the Reset pin), or by switching to the input mode.
Note also that when in the input mode, the contents of register R14 will follow the signals applied to the I/O port. However, transfer of this data to the CPU bus requires a "read" operation as described above.

2.1.7 Joystick Port Operation

The joystick port (Register 14 of the Sound Chip - Section 2.1.5.6) is read via an IN-instruction directed at port F6H with selection of activating data from the left (player 1) or right (player 2) determined by Address bits 8 and 9 as shown in Figure 2.1.7-1. In order to address Register 14, a OEH must be written to port F5H (Sound Generator Address) prior to reading joystick data. Section 4.4 describes the software sequence necessary to control this hardware.

In the example of Figure 2.1.7-1, the joystick, shown schematically in the lower left of the drawing, is composed of a movable center stick which is pushed up to touch the up-contact and, therefore, electronically connects pin-8 to pin-1. In this state, a read of port F6H with address bit A8 high, causes actions as follows:

1. Address A8 high turns on transistor Q8
2. Q8 drives cable pin-8 low
3. The movable center stick of the joystick in contact with the up-contact results in a conductive path from cable pin-8 to cable pin-1.
4. Pin-1 low results in a 0 in bit position 0 of the I/O register via the isolation diode.

The various positions of the stick similarly result in various bits being read from the I/O register.

Note that +5 volts and ground are available on the connector so +5V logic could be attached to the joystick port.
FIGURE 2.1.7-1
JOYSTICK PORT OPERATION
2.1.8 Control Logic

The control logic of the TS2068 is primarily a Standard Cell Logic Device in a 68-pin JEDEC leaded carrier package and includes the following major functions:

<table>
<thead>
<tr>
<th>SECTION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.8.1</td>
<td>Bank Selection Logic</td>
</tr>
<tr>
<td>2.1.8.2</td>
<td>Z-80 Clock Generation</td>
</tr>
<tr>
<td>2.1.8.3</td>
<td>Display Timing, DMA Display File Access, Attribute Control, and Pixel Data Serial Shift</td>
</tr>
<tr>
<td>2.1.8.4</td>
<td>Interruption Generation</td>
</tr>
</tbody>
</table>

BEEP Output (See Section 2.1.13.2)
CASSETTE I/O (See Section 2.1.12).

Additionally, Table 2.1.8-1 provides a description of the function of each SCLD I/O pin. See the System Schematic in Appendix D for pin numbering.

2.1.8.1 Bank Selection Logic

The TS2068 is a Z-80 based computer, therefore it can directly address only 64K bytes of memory via its 16-bit address. Additionally, since the Z-80 has no relocation or indirection capability, the conventional technique of extending the memory space available to the Z-80 is bank switching. The TS2068 provides extended bank switching by allowing selection of memory in 8K "chunks" which are identified by bank number and chunk number as illustrated in Figure 2.1.8-1 for the internal bank selection logic.

The externally sourced BE⁻ (Bank Enable) signal can be used by external logic to disable the internally controlled memories.

As shown in Figure 2.1.8-1:

1. The cartridge is selected on a memory access with:
   a. Port FF bit 7 = 0
   b. The HSR at port F4h has a "1" in the bit selected by a decode of Address bits A13-A15. and
   c. BE⁻ is high

causing activation of ROSCS⁻ (ROS Chip Select).
(2) The EXROM bank is selected on a memory access with:
   a. Port FF bit 7 = 1
   b. The HSR at port F4H has a "1" in the bit selected by a decode of Address bits A13 - A15.
   c. BE is high

causing the activation of EXROM (Ext. ROM Enable)

(3) The Home Bank is selected on a memory access with:
   a. The HSR at Port F4H has a "0" in the bit selected by a decode of Address bits A13 - A15.
   b. BE is high.

causing the activation of the appropriate enable signal as detailed below.

To understand the details of the schematic of Section 2.2 (Appendix D):

(1) SELECT CARTRIDGE of Figure 2.1.8-1 involves activating RSCS to its low active state

(2) SELECT EXROM of Figure 2.1.8-1 involves activating EXROM to its low active state

(3) SELECT HOME BANK of Figure 2.1.8-1 involves
   a. Activating ROMCS to its low active state when A15=0 and A14=0
   b. Activating CAS1 to its low active state when A15=0 and A14=1
   c. Activating CAS2 to its low active state when A15=1 and A14=0
   d. Activating CAS3 to its low active state when A15=1 and A14=1.
FIGURE 2.1.8-1
BANK SELECTION LOGIC
### TABLE 2.1.8-1

**SCLD I/O PIN FUNCTION DEFINITIONS**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>NAME</th>
<th>DIRECTION OF SCLD</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO-A7</td>
<td>Address Bus</td>
<td>In</td>
<td>Address Bus lines Input from Z80A</td>
</tr>
<tr>
<td>A13-A15</td>
<td></td>
<td></td>
<td>Data Bus inputs/outputs from/to Z80A through U9-74LS245 or inputs from display RAM (16K) - U6 and U7</td>
</tr>
<tr>
<td>DO-D7</td>
<td>Data Bus</td>
<td>In/Out</td>
<td>Inputs from 5 lines of keyboard matrix - goes low at one of 8 address line (active low) sequences on I/O Request</td>
</tr>
<tr>
<td>KBO-KB4</td>
<td>Keyboard Outputs</td>
<td>In</td>
<td>To refresh and address 8th bit address line input of RAM memory (not display) of 32K of 4416 RAM's (Home Bank 8000H to FFFFH)</td>
</tr>
<tr>
<td>MAO-MA7</td>
<td>Muxed Adrs.Bus</td>
<td>Out</td>
<td>Display memory muxed address bus and refresh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>Tri-State</td>
<td>Out</td>
<td>Tri-State control for address and data buffers when CPU is addressing display memory at same time display controller is addressing the display memory</td>
</tr>
<tr>
<td></td>
<td>Display Memory Ctl.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCPU</td>
<td>Clock to CPU</td>
<td>Out</td>
<td>CLK - Clock to Z80A CPU which is interrupted to stop CPU when CPU wants to address display RAM at same time as display controller</td>
</tr>
<tr>
<td>RDEF</td>
<td>Read Direction</td>
<td>Out</td>
<td>To control read/write direction of 74LS245 Data Bus Buffer between CPU and SCLD</td>
</tr>
<tr>
<td></td>
<td>Control to SCLD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROMCS</td>
<td>Home ROM Chip Select</td>
<td>Out</td>
<td>To activate the 16K Home ROM (first 16K) when memory selection (MS) is set to Home Bank</td>
</tr>
<tr>
<td>RAST</td>
<td>Row Address Strobe #1</td>
<td>Out</td>
<td>To activate row address strobe for display memory only during memory read/write, refresh and display read</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>NAME</td>
<td>DIRECTION</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------</td>
<td>------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>CAST</td>
<td>Column Address Strobe #1</td>
<td>Out</td>
<td>To activate column address strobe for display memory only (2nd 16K) during memory read/write and display read</td>
</tr>
<tr>
<td>CAS2</td>
<td>Column Address Strobe #2</td>
<td>Out</td>
<td>To activate column address strobe for Home Bank RAM (3rd 16K)</td>
</tr>
<tr>
<td>CAS3</td>
<td>Column Address Strobe #3</td>
<td>Out</td>
<td>To activate column address strobe for Home Bank RAM (4th 16K)</td>
</tr>
<tr>
<td>DRAMWE</td>
<td>Dynamic RAM Write Enable</td>
<td>Out</td>
<td>When active low, enables a write into the display RAM only</td>
</tr>
<tr>
<td>MUX</td>
<td>Mux Control of RAM Address</td>
<td>Out</td>
<td>Mux control to 74LS157 (U10 &amp; U11) to multiplex the row and column addresses to all dynamic RAM's</td>
</tr>
<tr>
<td>V</td>
<td>Chroma Vector V</td>
<td>Out</td>
<td>Color vector level for quadrature (R-Y) input to video modulator</td>
</tr>
<tr>
<td>Y</td>
<td>Luminance Y</td>
<td>Out</td>
<td>Luminance (brightness) control level</td>
</tr>
<tr>
<td>RD</td>
<td>Read to CPU</td>
<td>In</td>
<td>CPU is reading from a memory or I/O location</td>
</tr>
<tr>
<td>WR</td>
<td>Write from CPU</td>
<td>In</td>
<td>CPU is writing to a memory or I/O location</td>
</tr>
<tr>
<td>NREQ</td>
<td>Memory Request</td>
<td>In</td>
<td>CPU is requesting access to a memory location to read or write</td>
</tr>
<tr>
<td>TREQ</td>
<td>I/O Request</td>
<td>In</td>
<td>CPU is requesting access to an I/O location to read or write</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>NAME</td>
<td>DIRECTION OF SCLD</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------</td>
<td>-------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RFSH</td>
<td>Refresh</td>
<td>In</td>
<td>CPU is generating a refresh address to refresh dynamic RAM's</td>
</tr>
<tr>
<td>Tape In</td>
<td>Tape Input</td>
<td>In</td>
<td>Magnetic tape signal input</td>
</tr>
<tr>
<td>BE</td>
<td>Bank Enable</td>
<td>In</td>
<td>When active low, indicates that internal memory is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Home, Extension and Dock Banks) and an external memory is in use</td>
</tr>
<tr>
<td>EXROM</td>
<td>Extension ROM Select</td>
<td>Out</td>
<td>Active low chip select signal for Extension ROM</td>
</tr>
<tr>
<td>VCC</td>
<td>+5 Volt Power</td>
<td>In</td>
<td>Power (+5V) input to SCLD</td>
</tr>
<tr>
<td>INT</td>
<td>Interrupt to CPU</td>
<td>Out</td>
<td>Interrupts CPU to handle keyboard strobing and timer for PAUSE command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Open drain N channel with internal pull-up</td>
</tr>
<tr>
<td>ROSCS</td>
<td>ROS Chip Select</td>
<td>Out</td>
<td>ROM-Oriented Software (Cartridge Bank) Chip Select</td>
</tr>
<tr>
<td>SPKR/TAPE OUT</td>
<td>Speaker and Tape Output</td>
<td>Out</td>
<td>Digital output to magnetic tape and to sound amplifier for speaker output</td>
</tr>
<tr>
<td>OC</td>
<td>Clock &quot;C&quot;</td>
<td>Out</td>
<td>Clock for sound chip @1.764 MHz.</td>
</tr>
<tr>
<td>BDIR</td>
<td>Bus Direction to Sound Chip</td>
<td>Out</td>
<td>A bus direction control signal to the PSG. When high the sound chip</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>either receives a write to PSG or latches addresses from the data bus</td>
</tr>
<tr>
<td>BCI</td>
<td>Bus Control to Sound Chip</td>
<td>Out</td>
<td>A bus control signal to the PSG. When high the sound chip either is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>read to data bus or latches addresses from the data bus</td>
</tr>
<tr>
<td>SYMBOL</td>
<td>NAME</td>
<td>DIRECTION OF SCLD IN/OUT</td>
<td>FUNCTION</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>OSC Out</td>
<td>Oscillator Out</td>
<td>Out</td>
<td>Xtal Oscillator amplifier output to drive crystal</td>
</tr>
<tr>
<td>OSC In</td>
<td>Oscillator In</td>
<td>In</td>
<td>Xtal Oscillator amplifier input to sense crystal signal</td>
</tr>
<tr>
<td>U</td>
<td>Chroma Vector U</td>
<td>Out</td>
<td>Color vector level for quadrature (B-Y) input to video modulator</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
<td>In</td>
<td>Ground return of SCLD</td>
</tr>
<tr>
<td>Ø</td>
<td>Buffered Clock</td>
<td>Out</td>
<td>Buffered CPU clock to outside (J1 connector)</td>
</tr>
<tr>
<td>R</td>
<td>Red Color Output</td>
<td>Out</td>
<td>Produce color signals to RGB monitor (TTL level)</td>
</tr>
<tr>
<td>G</td>
<td>Green Color Output</td>
<td>Out</td>
<td>Produce color signals to RGB monitor (TTL level)</td>
</tr>
<tr>
<td>B</td>
<td>Blue Color Output</td>
<td>Out</td>
<td>Produce color signals to RGB monitor (TTL level)</td>
</tr>
</tbody>
</table>
2.1.8.2 Z-80 Clock Generation

The oscillator circuit utilizes an AT-cut quartz crystal at 14.112 MHz. This oscillator feeds a divide by 4 chain to generate the 3.528 MHz clock for the CPU (0 CPU). This clock runs continuously except when the CPU addresses the 16K bytes of RAM containing the video display file at the same time the video display processor logic requires access to that same RAM. For this contention case the CPU clock is stopped in the high state until the video display processor access has been completed, then the CPU clock continues in its normal manner.

2.1.8.3 Display File H/W Control and Timing

The 14.112 MHz oscillator is also used to drive the counter chain deriving video timing. By dividing the 14.112 MHz signal by 896 a 15.75 KHz horizontal sweep frequency is generated. The 15.75 KHz signal feeds a 9-stage counter which counts from 0 to 106H (262 decimal) developing the 60.1145 Hz vertical sync. See Figure 2.1.8-2.

During each horizontal scan the video display processor accesses, in the standard video mode, 32 bytes of pixel data plus 32 bytes of attributes by 32 memory accesses reading 2 bytes per access in RAM page mode, i.e. the low order address bits are provided to the RAM once via RAS activation, then the data byte is read during the first activation of CAS and the attribute byte is read during the second activation of CAS. The page mode operation is completed by deactivating RAS. (See Fig. 2.1.8-2.)

The accessed pixel data is serially shifted out to the video generation circuitry at a rate of 1 bit each 142 nanoseconds (7.056 MHz) resulting in the need to fetch a new data/attribute pair each 1.134 microseconds during the horizontal scan time. The shifted out pixel information is used to control the selection of the 3 paper color (pixel=0) or 3 ink color (pixel=1) bits to be gated out as the R, G, and B signals. When FLASH is enabled by the attribute byte, the INK and PAPER field information is swapped at the 1.879 Hz. flash rate. The R, G, and B signals control the D-to-A converter which generates the proper U, V, and Y outputs for use by the 1889 to create composite video.

The address information provided to the RAM's during RAS and CAS times is as shown in Figure 2.1.8-2. This address generation logic explains the non-sequential nature of the video display as described in Section 2.1.10.
FIGURE 2.1.8-2
VIDEO DISPLAY PROCESSOR RAM ADDRESS GENERATION
(Normal Video Mode)

DISPLAY PIXEL DATA ADDRESS
Address Bit: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Range=
4000H-
57FFH

DISPLAY ATTRIBUTE ADDRESS
Address Bit: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Range=
5800H-
5AFFH

VIDEO TIMING COUNTER CHAIN

60.1145 Hz
Vert.Sync.

15.75 KHz
Horiz.Sweep

1.764 MHz

(14.112 MHz/8)
2.1.3.4 Interruption Generation (17 ms)

During the vertical blanking interval (once each 16.635 ms) the SCLD, if enabled by the INTEN bit (Bit 6) of I/O Port FFH, activates the INT signal which directly connects to the INT input to the Z80. A CPU maskable interruption can then occur, as described in Section 2.1.3.7, if enabled.

2.1.9 Keyboard

The keyboard for the TS 2068 has forty-two (42) hard keys (typewriter style) with tactile feel utilizing an over-dead-center type of rubber spring pad and a carbon pill that hits the P.C. board, just under the keyboard, to short-out a pair of closely placed precious metal contacts. The read-out matrix is an eight by five cross point switching as shown in Figure 2.1.9-1.

Each switch closure connects one of the eight high order address lines (by going low through a diode) to one of the five input lines to the SCLD (K80 through KB4).

Scanning is by software algorithm as described in Section 4.1.1. During the IN instruction, address bits A0-A7=FEH select the Keyboard I/O port while bits A8-A15 select the particular 5 keys to be sampled during the particular IN instruction execution. For example, an IN instruction directed at the keyboard I/O port with address bit A8 low and A9-A15 high will supply 0's on K80, K81, K82, K83, and/or KB4 if the CAP SHIFT, Z, X, C, and/or V keys are respectively depressed.

Note that when reading the I/O port FEH, data bits D5-D7 are not part of the keyboard information.

Section 2.4.7 details the connection of the keyboard to the main P.C. board.

2.1.10 16K Video Display RAM

The 16K-byte video display RAM, composed of two 4416's, is isolated from the Z80A CPU by the SCLD control logic and buffers to allow the video display processor to access pixel and attribute data from the display files independent of the CPU (see Section 2.1.8.3).

The Video Display RAM is located in Chunks 2 and 3 of the Home Bank, beginning at 4000H and 6000H respectively. Figure 2.1.10-1 illustrates the organization of the Primary Display File located at 4000H. The second display file utilizes the same organization. Based on the video mode set via Port FFH, the video hardware accesses the RAM for pixel data and attribute control information.
Figure 2.10. KEYBOARD SCHEMATIC
# FIG. 2.1.10-1

**DISPLAY FILE ORGANIZATION (NORMAL MODE)**

<table>
<thead>
<tr>
<th>B Scan</th>
<th>0</th>
<th>32 BYTES</th>
<th>32 BYTES</th>
<th>32 BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LINE 0</td>
<td>LINE 1</td>
<td>LINE 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4000</td>
<td>401F</td>
<td>4020</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>4100</td>
<td>411F</td>
<td>4120</td>
</tr>
<tr>
<td>O</td>
<td>2</td>
<td>4200</td>
<td>421F</td>
<td>4220</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>4300</td>
<td>431F</td>
<td>4320</td>
</tr>
<tr>
<td>K</td>
<td>4</td>
<td>4400</td>
<td>441F</td>
<td>4420</td>
</tr>
<tr>
<td>O</td>
<td>5</td>
<td>4500</td>
<td>451F</td>
<td>4520</td>
</tr>
<tr>
<td>O</td>
<td>6</td>
<td>4600</td>
<td>461F</td>
<td>4620</td>
</tr>
<tr>
<td>O</td>
<td>7</td>
<td>4700</td>
<td>471F</td>
<td>4720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR.</td>
<td>CHAR.</td>
<td>CHAR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/0</td>
<td>0/1</td>
<td>0/31</td>
</tr>
<tr>
<td>B Scan</td>
<td>0</td>
<td>32 BYTES</td>
<td>32 BYTES</td>
<td>32 BYTES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINE 8</td>
<td>LINE 9</td>
<td>LINE 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4800</td>
<td>481F</td>
<td>4820</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>4900</td>
<td>491F</td>
<td>4920</td>
</tr>
<tr>
<td>O</td>
<td>2</td>
<td>4A00</td>
<td>4A1F</td>
<td>4A20</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>4B00</td>
<td>4B1F</td>
<td>4B20</td>
</tr>
<tr>
<td>K</td>
<td>4</td>
<td>4C00</td>
<td>4C1F</td>
<td>4C20</td>
</tr>
<tr>
<td>L</td>
<td>5</td>
<td>4D00</td>
<td>4D1F</td>
<td>4D20</td>
</tr>
<tr>
<td>O</td>
<td>6</td>
<td>4E00</td>
<td>4E1F</td>
<td>4E20</td>
</tr>
<tr>
<td>O</td>
<td>7</td>
<td>4F00</td>
<td>4F1F</td>
<td>4F20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR.</td>
<td>CHAR.</td>
<td>CHAR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/0</td>
<td>0/1</td>
<td>0/31</td>
</tr>
<tr>
<td>B Scan</td>
<td>0</td>
<td>32 BYTES</td>
<td>32 BYTES</td>
<td>32 BYTES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINE 16</td>
<td>LINE 17</td>
<td>LINE 23</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5000</td>
<td>501F</td>
<td>5020</td>
</tr>
<tr>
<td>L</td>
<td>1</td>
<td>5100</td>
<td>511F</td>
<td>5120</td>
</tr>
<tr>
<td>O</td>
<td>2</td>
<td>5200</td>
<td>521F</td>
<td>5220</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>5300</td>
<td>531F</td>
<td>5320</td>
</tr>
<tr>
<td>K</td>
<td>4</td>
<td>5400</td>
<td>541F</td>
<td>5420</td>
</tr>
<tr>
<td>O</td>
<td>5</td>
<td>5500</td>
<td>551F</td>
<td>5520</td>
</tr>
<tr>
<td>O</td>
<td>6</td>
<td>5600</td>
<td>561F</td>
<td>5620</td>
</tr>
<tr>
<td>O</td>
<td>7</td>
<td>5700</td>
<td>571F</td>
<td>5720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHAR.</td>
<td>CHAR.</td>
<td>CHAR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/0</td>
<td>0/1</td>
<td>0/31</td>
</tr>
</tbody>
</table>

**ATTRIBUTE FILE:**

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>LINE 0</th>
<th>LINE 1</th>
<th>LINES 2 - 6</th>
<th>LINES 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5800</td>
<td>581F</td>
<td>5820</td>
<td>583F</td>
</tr>
<tr>
<td>1</td>
<td>5900</td>
<td>591F</td>
<td>5920</td>
<td>593F</td>
</tr>
<tr>
<td>2</td>
<td>5A00</td>
<td>5A1F</td>
<td>5A20</td>
<td>5A3F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>LINE 8</th>
<th>LINE 9</th>
<th>LINES 10-14</th>
<th>LINES 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5800</td>
<td>581F</td>
<td>5820</td>
<td>583F</td>
</tr>
<tr>
<td>1</td>
<td>5900</td>
<td>591F</td>
<td>5920</td>
<td>593F</td>
</tr>
<tr>
<td>2</td>
<td>5A00</td>
<td>5A1F</td>
<td>5A20</td>
<td>5A3F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BLOCK</th>
<th>LINE 16</th>
<th>LINE 17</th>
<th>LINES 18-22</th>
<th>LINES 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5800</td>
<td>581F</td>
<td>5820</td>
<td>583F</td>
</tr>
<tr>
<td>1</td>
<td>5900</td>
<td>591F</td>
<td>5920</td>
<td>593F</td>
</tr>
<tr>
<td>2</td>
<td>5A00</td>
<td>5A1F</td>
<td>5A20</td>
<td>5A3F</td>
</tr>
</tbody>
</table>

46
2.1.11 Video Generation

2.1.11.1 Composite Video

The U, V, and Y signals from the SCLD are supplied to the LM1889 and associated circuitry to produce composite video and modulated RF. This circuitry produces color vectors at approximately the following angles:

<table>
<thead>
<tr>
<th>PHASE</th>
<th>TS 2068 (Degrees)</th>
<th>NTSC STANDARD (Degrees)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>350</td>
<td>350</td>
</tr>
<tr>
<td>Magenta</td>
<td>64</td>
<td>62</td>
</tr>
<tr>
<td>Red</td>
<td>116</td>
<td>112</td>
</tr>
<tr>
<td>Green</td>
<td>242</td>
<td>240</td>
</tr>
<tr>
<td>Cyan</td>
<td>284</td>
<td>284</td>
</tr>
<tr>
<td>Yellow</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>Reference</td>
<td>224</td>
<td>180</td>
</tr>
</tbody>
</table>

The Front Porch, Sync Pulse, Back Porch, and Color Burst portions of the composite video signal are illustrated in Figure 2.1.11-1. In proper adjustment the following should be observed:

- Sync Pulse = 40 +/- 2 IRE units
- Color Burst = 35 to 45 IRE units
- Color Burst Freq. = 3.579545 MHz +/- 70 Hz

The following three facts may aid in understanding problems with certain monitors.

1. The color burst is not synchronous with the waveform since it is generated from the 3.579545 MHz crystal and the waveform is derived from the 14.112 MHz crystal. The result is observed ripples at color boundaries, e.g. green to magenta.

2. The color burst duration is 8 cycles while standard TV broadcast stations provide 9 cycles. This "short" burst is a problem for some monitors.

3. The color burst starts 6.4 microseconds from the leading edge of sync. Many monitors are designed to expect this start as early as 5.3 microseconds, thus these monitors may not produce color when attached to the TS 2068.
2.11.2 RF Modulator

The composite video information is used to AM modulate the selected channel frequency via the LM1889 and associated Channel 2/3 tank circuitry. The modulated output is filtered through the output filter network to reduce harmonic generation to comply with FCC requirements. The RF circuitry is physically contained inside the RF-can at the rear left corner of the PCB (at the RF output jack). 75 ohms is the output impedance.
2.1.12 Cassette I/O

See Sections 2.1.13.2, 2.4.3 and 4.2.

2.1.13 Port Map

Table 2.1.13-1 summarizes the I/O addressing of ports utilized by the TS 2068. Details of the data bits of each of these ports is provided by the following sections.

2.1.13.1 Display Enhancement Control (Port FFH)

The display enhancement control register within the SCLD controls:

a) Selection of Enhanced Video Modes

b) Ink selection for 64-Column Mode

c) Enable/Inhibit the 17 ms interruption to the Z80

d) Selection of Extension ROM or Cartridge (see Section 2.1.8.1)

<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>64-Column Mode</td>
<td>Video Mode Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ink/Paper Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O00 - Black/White</td>
<td>000 - Normal (Primary Display File)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O01 - Blue/Yellow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O10 - Red/Cyan</td>
<td>001 - Second Display File</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O11 - Magenta/Green</td>
<td>010 - High Res. Graphics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 - Green/Magenta</td>
<td>110 - 64-Column Mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101 - Cyan/Red</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110 - Yellow/Blue</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111 - White/Black</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inhibit 17 ms Interruption (0 to Enable)

EXROM/Cartridge Select (See 2.1.8.1)
### TABLE 2.1.13-1

**I/O PORT MAP**

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>PORT ADDRESS</th>
<th>(HEX)</th>
<th>(DECIMAL)</th>
<th>(BINAR Y)</th>
<th>OPERATION</th>
<th>REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Enhancement Control</td>
<td>FF</td>
<td>255</td>
<td>11111111</td>
<td>R/W</td>
<td>2.1.10, 2.1.13.1, 3.2.2.3, 5.2</td>
<td></td>
</tr>
<tr>
<td>Keyboard/Tape I/O</td>
<td>FE</td>
<td>254</td>
<td>11111110</td>
<td>R/W</td>
<td>2.1.9, 2.1.13.2, 2.4.3, 4.1.1, 4.2</td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>FD</td>
<td>253</td>
<td>11111101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>FC</td>
<td>252</td>
<td>11111100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TS 2040 Printer</td>
<td>FB</td>
<td>251</td>
<td>11111011</td>
<td>R/W</td>
<td>2.1.13.3, 4.1.3</td>
<td></td>
</tr>
<tr>
<td>Sound Chip &amp; Joystick Data</td>
<td>F6</td>
<td>246</td>
<td>11110110</td>
<td>R/W</td>
<td>2.1.6, 2.1.7, 2.1.13.4, 2.4.4, 4.3, 4.5</td>
<td></td>
</tr>
<tr>
<td>Sound Chip Address</td>
<td>F5</td>
<td>245</td>
<td>11110101</td>
<td>W</td>
<td>Same</td>
<td></td>
</tr>
<tr>
<td>Horizontal Select Register</td>
<td>F4</td>
<td>244</td>
<td>11110100</td>
<td>R/W</td>
<td>2.1.8.1</td>
<td></td>
</tr>
</tbody>
</table>

2.1.13.2 Keyboard/Tape I/O (Port FEH)

Port FEH is used to input Keyboard and Tape data and to output Border color, Tape data, and Sound (BEEP) tones.

**READ (IN)**

```
<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>KEYBOARD INPUT DATA (See 2.1.9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Not Used (Set to 0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TAPE INPUT (See 4.2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
WRITE (OUT)

D7  D6  D5  D4  D3  D2  D1  D0

NOT USED

BORDER COLOR - 000 Black
001 Blue
010 Red
011 Magenta
100 Green
101 Cyan
110 Yellow
111 White

TAPE
OUTPUT
(See 4.2)

SOUND (BEEP)
OUT

2.1.13.3 TS 2040 Printer (Port 1XXXXOXXX)

The TS 2040 Printer peripheral is written to and status read from via OUT and IN instructions with Bit 7 = 1 and Bit 2 = 0 (other bits are not decoded by the printer).

READ (IN)

D7  D6  D5  D4  D3  D2  D1  D0

NOT USED

Printer Not Configured

Ready for Next Pixel

Start of Paper

WRITE (OUT)

D7  D6  D5  D4  D3  D2  D1  D0

NOT USED

NOT USED

Motor Speed
Select - 0 = Fast
1 = Slow

Motor ON/OFF
0 = ON
1 = OFF

Pixel to Print - 0 = None
1 = Black
2.1.13.4 Sound Chip & Joystick (Ports F5H and F6H)

Ports F5H and F6H are used to control and access the Sound Generator and the Joysticks. Details of the registers available via these ports is contained in Sections 2.1.6 and 2.1.7.

2.1.13.5 Horizontal Select Register (Port F4H)

The HSR addressed via Port F4H is used in the control of the Bank Switching logic as detailed in Section 2.1.8. Each bit, when set, enables the corresponding 8K memory "chunk" in either the Dock Bank (Port FF, Bit 7=0) or the Extension ROM Bank (Port FF, Bit 7=1). The HSR must be set to all zeroes in order to enable the entire Home Bank.

2.2 Schematic Diagram

Appendix D contains a detailed schematic diagram of the TS 2068.

2.3 Unit Absolute Ratings

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
<th>MIN</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS</td>
<td>Storage Temperature</td>
<td>-40°C</td>
<td>+65°C</td>
</tr>
<tr>
<td>VAC</td>
<td>AC Line Voltage</td>
<td>105V</td>
<td>130V</td>
</tr>
<tr>
<td>Ta</td>
<td>Operating Ambient Temp</td>
<td>0°C</td>
<td>40°C</td>
</tr>
<tr>
<td>Vin</td>
<td>Voltage on any Logic</td>
<td>-0.3V</td>
<td>+5.3V</td>
</tr>
<tr>
<td></td>
<td>Pin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vin (EAR)</td>
<td>EAR input Peak AC</td>
<td>-2.0V</td>
<td>+5.0V</td>
</tr>
<tr>
<td>Vdc (IN)</td>
<td>Input DC Voltage</td>
<td>14.75V</td>
<td>26V</td>
</tr>
</tbody>
</table>

2.4 Interfaces and Connectors

The TS2068 has a number of specialized interfaces that are accessible via the following connectors:

<table>
<thead>
<tr>
<th>CONNECTOR</th>
<th>TYPE</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Bus</td>
<td>2X32 Card Edge</td>
<td>Right Rear</td>
</tr>
<tr>
<td>Cartridge</td>
<td>2X18 Card Edge</td>
<td>Under TCC door</td>
</tr>
<tr>
<td>MIC</td>
<td>1/8&quot; Mini Phone</td>
<td>Rear</td>
</tr>
<tr>
<td>EAR</td>
<td>1/8&quot; Mini Phone</td>
<td>Rear</td>
</tr>
<tr>
<td>Player 1 Joystick</td>
<td>9-pin &quot;D&quot;</td>
<td>Left Side</td>
</tr>
<tr>
<td>Player 2 Joystick</td>
<td>9-pin &quot;D&quot;</td>
<td>Right Side</td>
</tr>
<tr>
<td>Monitor</td>
<td>RCA Phono</td>
<td>Rear</td>
</tr>
<tr>
<td>TV</td>
<td>RCA Phono</td>
<td>Rear</td>
</tr>
<tr>
<td>Keyboard</td>
<td>14-pin SIP</td>
<td>Inside-Left Rear</td>
</tr>
<tr>
<td>AC Adapter</td>
<td></td>
<td>Rear</td>
</tr>
</tbody>
</table>

52
2.4.1 System Bus Connector - P1

The TS2068 provides a 2 X 32 pin connector, which is designated as P1, at the right rear corner of the console. The mechanical, functional, and electrical requirements of the system bus connector are detailed in the following tables and figures:

<table>
<thead>
<tr>
<th>FIGURE/TABLE</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.4.1-1 P1</td>
<td>Mating Connector Mechanical Requirements</td>
</tr>
<tr>
<td>Figure 2.4.1-2 P1</td>
<td>Signal Layout</td>
</tr>
<tr>
<td>Table 2.4.1 - 1 P1</td>
<td>Signal Definition</td>
</tr>
<tr>
<td>Table 2.4.1 - 2 P1</td>
<td>Signal Electrical Characteristics</td>
</tr>
</tbody>
</table>

FIGURE 2.4.1-1

P1 MATING CONNECTOR MECHANICAL REQUIREMENTS

64 PIN CONNECTOR

SCALE: NONE
**SPECIFICATIONS:**

<table>
<thead>
<tr>
<th>LTR</th>
<th>DIMENSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>82.86 (.325)</td>
</tr>
<tr>
<td>W</td>
<td>8.928 ± 0.127 (.350 ± .005)</td>
</tr>
<tr>
<td>H</td>
<td>18.9750.354 (.860 ± .010)</td>
</tr>
<tr>
<td>A</td>
<td>3.54 (.100)</td>
</tr>
<tr>
<td>B</td>
<td>31 EQUAL SPACES AT 2.54 (.100) = 78.74 (.306)</td>
</tr>
<tr>
<td>C</td>
<td>3.54 (.10)</td>
</tr>
<tr>
<td>D</td>
<td>1.737 (.068) MAX</td>
</tr>
<tr>
<td>E</td>
<td>0.38210.506 (.330 ± .020)</td>
</tr>
<tr>
<td>F</td>
<td>FOR 1.875 (.073) BOARD</td>
</tr>
</tbody>
</table>

*All dimensions are in millimeters. Dimension shown [L2.X] are in inches.*

**NOTES:**

1. **INSULATOR MATERIAL:** Insulator body shall be 30% glass-filled polyester and shall meet UL94V0 requirements.
2. **CONTACT MATERIAL:** Contact material shall be phosphor bronze.
3. **CONTACT FINISH:** Contacts shall be selectively plated with gold, 0.00038 (.000015) thick over nickel on contact surfaces.
4. **INSERTION FORCE:** Insertion forces shall be 170.1-283.5 grams (6-10 oz) per contact pair using a 1.875 (.073) flat steel test blade.
5. **WITHDRAWAL FORCE:** Withdrawal forces shall be 226.8-340.2 grams (8-12 oz) per contact pair using a 1.875 (.073) flat test blade.
6. **NORMAL FORCE:** Normal force shall be 85.05 grams (3 oz) minimum when mated with a 1.37 (.054) thick test board.
7. **PURCHASE FROM:** San Diego Microtronics INC. San Diego, CA 92123.

**FIGURE 2.4.1-2**

**P1 CONNECTOR SIGNAL LAYOUT**

**COMPONENT SIDE**

**TS1000 COMPATIBLE**

**NON-COMPONENT SIDE**

(View from front of computer)
<table>
<thead>
<tr>
<th>PIN #</th>
<th>SIGNAL NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1A</td>
<td>GND</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>1B</td>
<td>GND</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>2A</td>
<td>EAR</td>
<td>EAR Input</td>
</tr>
<tr>
<td>2B</td>
<td>SPKR/TAPE OUT</td>
<td>Speaker/Tape Output</td>
</tr>
<tr>
<td>3A</td>
<td>A7/RA</td>
<td>Refresh Address Bit 7 Buffered</td>
</tr>
<tr>
<td>3B</td>
<td>+15V</td>
<td>+15 Volts DC</td>
</tr>
<tr>
<td>4A</td>
<td>D7</td>
<td>Data Bus Bit 7</td>
</tr>
<tr>
<td>4B</td>
<td>+5V</td>
<td>+5 Volts</td>
</tr>
<tr>
<td>5A</td>
<td>DZIN</td>
<td>Daisy In (Not Connected)</td>
</tr>
<tr>
<td>5B</td>
<td>Not Used</td>
<td>--</td>
</tr>
<tr>
<td>6A</td>
<td>Slot</td>
<td>--</td>
</tr>
<tr>
<td>6B</td>
<td>Slot</td>
<td>--</td>
</tr>
<tr>
<td>7A</td>
<td>DO</td>
<td>Data Bus Bit 0</td>
</tr>
<tr>
<td>7B</td>
<td>GND</td>
<td>Power Ground</td>
</tr>
<tr>
<td>8A</td>
<td>D1</td>
<td>Data Bus Bit 1</td>
</tr>
<tr>
<td>8B</td>
<td>GND</td>
<td>Power Ground</td>
</tr>
<tr>
<td>9A</td>
<td>D2</td>
<td>Data Bus Bit 2</td>
</tr>
<tr>
<td>9B</td>
<td>0</td>
<td>CPU Clock (Inverted)</td>
</tr>
<tr>
<td>10A</td>
<td>D6</td>
<td>Data Bus Bit 6</td>
</tr>
<tr>
<td>10B</td>
<td>A0</td>
<td>Address Bus Bit 0</td>
</tr>
<tr>
<td>11A</td>
<td>D5</td>
<td>Data Bus Bit 5</td>
</tr>
<tr>
<td>11B</td>
<td>A1</td>
<td>Address Bus Bit 1</td>
</tr>
<tr>
<td>12A</td>
<td>D3</td>
<td>Data Bus Bit 3</td>
</tr>
<tr>
<td>12B</td>
<td>A2</td>
<td>Address Bus Bit 2</td>
</tr>
<tr>
<td>13A</td>
<td>D4</td>
<td>Data Bus Bit 4</td>
</tr>
<tr>
<td>13B</td>
<td>A3</td>
<td>Address Bus Bit 3</td>
</tr>
<tr>
<td>14A</td>
<td>INT</td>
<td>Interrupt Request (Active Low)</td>
</tr>
<tr>
<td>14B</td>
<td>A15B</td>
<td>Address Bus Bit 15, Buffered</td>
</tr>
<tr>
<td>15A</td>
<td>NM1</td>
<td>Non-Maskable Int. (Active Low)</td>
</tr>
<tr>
<td>15B</td>
<td>A14B</td>
<td>Address Bus Bit 14, Buffered</td>
</tr>
<tr>
<td>16A</td>
<td>HALT</td>
<td>CPU HALT Indicator (Active Low)</td>
</tr>
<tr>
<td>16B</td>
<td>A13B</td>
<td>Address Bus Bit 13, Buffered</td>
</tr>
<tr>
<td>17A</td>
<td>MREQB</td>
<td>Memory Request (Active Low), Bfrd.</td>
</tr>
<tr>
<td>17B</td>
<td>A12</td>
<td>Address Bus 12</td>
</tr>
<tr>
<td>18A</td>
<td>TREQB</td>
<td>I/O Request (Active Low), Bfrd.</td>
</tr>
<tr>
<td>18B</td>
<td>A11</td>
<td>Address Bus 11</td>
</tr>
<tr>
<td>19A</td>
<td>RMB</td>
<td>Read (Active Low), Buffered</td>
</tr>
<tr>
<td>19B</td>
<td>A10</td>
<td>Address Bus Bit 10</td>
</tr>
<tr>
<td>20A</td>
<td>WBB</td>
<td>Write (Active Low), Buffered</td>
</tr>
<tr>
<td>20B</td>
<td>A9</td>
<td>Address Bus Bit 9</td>
</tr>
<tr>
<td>21A</td>
<td>BUSAK</td>
<td>Bus Acknowledge (Active Low)</td>
</tr>
<tr>
<td>21B</td>
<td>A8</td>
<td>Address Bus Bit 8</td>
</tr>
<tr>
<td>22A</td>
<td>WAIT</td>
<td>CPU WAIT (Active Low)</td>
</tr>
<tr>
<td>22B</td>
<td>A7</td>
<td>Address Bus Bit 7</td>
</tr>
<tr>
<td>23A</td>
<td>BUSREQ</td>
<td>Bus Request (Active Low)</td>
</tr>
<tr>
<td>23B</td>
<td>A6</td>
<td>Address Bus Bit 6</td>
</tr>
<tr>
<td>24A</td>
<td>RESET</td>
<td>CPU Reset (Active Low)</td>
</tr>
<tr>
<td>24B</td>
<td>A5</td>
<td>Address Bus Bit 5</td>
</tr>
<tr>
<td>25A</td>
<td>HI</td>
<td>CPU HI State (Active Low)</td>
</tr>
<tr>
<td>25B</td>
<td>A4</td>
<td>Address Bus Bit 4</td>
</tr>
<tr>
<td>26A</td>
<td>RFSHB</td>
<td>Refresh (Active Low), Buffered</td>
</tr>
<tr>
<td>26B</td>
<td>DZOUT</td>
<td>Daisy Out (Not Connected)</td>
</tr>
<tr>
<td>27A</td>
<td>EXRON</td>
<td>Extension ROM Enable (Active Low)</td>
</tr>
<tr>
<td>27B</td>
<td>R</td>
<td>Color Signal - Red</td>
</tr>
<tr>
<td>28A</td>
<td>ROCS5</td>
<td>ROM Chip Select (Active Low)</td>
</tr>
<tr>
<td>28B</td>
<td>C</td>
<td>(Dock Bank Enable)</td>
</tr>
<tr>
<td>29A</td>
<td>BEE</td>
<td>Color Signal - Green</td>
</tr>
<tr>
<td>29B</td>
<td>B</td>
<td>Color Signal - Blue</td>
</tr>
<tr>
<td>30A</td>
<td>IOAS</td>
<td>Analog Sound Signal Output (0-5V)</td>
</tr>
<tr>
<td>30B</td>
<td>BUSISO</td>
<td>Composite Video Signal Output</td>
</tr>
<tr>
<td>31A</td>
<td>SOUND</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>31B</td>
<td>VIDEO</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>32A</td>
<td>GND</td>
<td>Signal Ground</td>
</tr>
<tr>
<td>32B</td>
<td>GND</td>
<td>Signal Ground</td>
</tr>
</tbody>
</table>

**NOTE:** All A Pins are on component side of board  
All B Pins are on non-component (soldering) side of board
## TABLE -2.4.1-2

**P1 SIGNAL ELECTRICAL CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Capacitive Loading MAX (PF)</th>
<th>V(OL) MAX</th>
<th>I(LOAD) MAX</th>
<th>V(OL) MIN</th>
<th>V(IL) MAX</th>
<th>V(TH) MIN</th>
<th>I IN (MAX)</th>
<th>Capacitive Loading MAX (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15B</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>40</td>
</tr>
<tr>
<td>A14B</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>40</td>
</tr>
<tr>
<td>A13B</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>40</td>
</tr>
<tr>
<td>A12</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>74</td>
</tr>
<tr>
<td>A11</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>24</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>74</td>
</tr>
<tr>
<td>A10</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>24</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A9</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A8</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A7</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A6</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A5</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A4</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A9</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A2</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A1</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>72</td>
</tr>
<tr>
<td>A0</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>1800</td>
<td>98</td>
</tr>
<tr>
<td>A7RB</td>
<td>30</td>
<td>0.5</td>
<td>0.35</td>
<td>2.7</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>120</td>
</tr>
<tr>
<td>T0R0B</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>MRE</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Bit</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>BUSAK</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>12</td>
</tr>
<tr>
<td>DMA</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>10</td>
</tr>
<tr>
<td>PAL</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>10</td>
</tr>
<tr>
<td>B1</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>10</td>
</tr>
<tr>
<td>B0</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>10</td>
</tr>
</tbody>
</table>

--- OPEN COLLECTOR WITH PULL-UP ---

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Capacitive Loading MAX (PF)</th>
<th>V(OL) MAX</th>
<th>I(LOAD) MAX</th>
<th>V(OL) MIN</th>
<th>V(IL) MAX</th>
<th>V(TH) MIN</th>
<th>I IN (MAX)</th>
<th>Capacitive Loading MAX (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>50</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>G</td>
<td>50</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>B</td>
<td>50</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

**VIDEO**

--- TO 75 ohm COAX ---

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Capacitive Loading MAX (PF)</th>
<th>V(OL) MAX</th>
<th>I(LOAD) MAX</th>
<th>V(OL) MIN</th>
<th>V(IL) MAX</th>
<th>V(TH) MIN</th>
<th>I IN (MAX)</th>
<th>Capacitive Loading MAX (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D1</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D2</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D3</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D4</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D5</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D6</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>D7</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
<td>20</td>
<td>120</td>
</tr>
<tr>
<td>SPKP/TAPE OUT</td>
<td>0.5</td>
<td>0.04</td>
<td>0.3-0.5</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>EAR</td>
<td>15</td>
<td>0.5</td>
<td>1.6</td>
<td>2.4</td>
<td>+/-.1.3</td>
<td>+/-.5.0</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>SOUND</td>
<td>100</td>
<td>0.0</td>
<td>---</td>
<td>2.5</td>
<td>-0.3</td>
<td>+5.0</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>BUSRD</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>0.8</td>
<td>2.0</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

**RESET**

--- 1 µF WITH 220K PULL-UP ---

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Capacitive Loading MAX (PF)</th>
<th>V(OL) MAX</th>
<th>I(LOAD) MAX</th>
<th>V(OL) MIN</th>
<th>V(IL) MAX</th>
<th>V(TH) MIN</th>
<th>I IN (MAX)</th>
<th>Capacitive Loading MAX (PF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO5S</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
2.4.1.1 Attachment of an RGB Monitor

The TS 2068 provides via the P1 rear-edge connector the ability to attach an RGB monitor for excellent picture clarity and resolution. The TTL-level logic signals appear directly on the rear-edge connector of the TS 2068 -- the necessary synch signals can be derived from the simple synch stripper/separater circuit described here.

The Schematic of Figure 2.4.1-3 shows the required connections and electronics. Attachment is via the 64-pin keyed P1 connector. Shielding should not normally be required, but ferrite beads are recommended on each wire to minimize EMI, TVI, etc.

Circuit Operation - R1 and the base-emitter junction of Q1 operate as a DC restoration circuit with current flowing only when the composite video input signal from connector pin B31 is at the synch level. With the charge maintained on C1, Q1 conducts only during the synch pulse interval (not during the color burst time). During this conduction interval, the composite synch signal appears in inverted form on the collector of Q1. The Q2 stage simply re-inverts the signal, providing at its collector a composite synch signal for the connected monitor.

To provide a separated Vertical synch pulse, R5 and C3 filter the output of Q1 to partially eliminate the Horizontal synch pulses which are shorter than the Vertical synch pulses. The partially filtered inverted signal is re-inverted by Q3, then R6 and C4 complete the elimination of the Horizontal synch pulses so that a separate Vertical synch pulse is supplied for the attached monitor.

Signals R, G, and B from connector pins B27, B28, and B29 can be supplied directly to the attached monitor.

![Figure 2.4.1-3](image-url)
2.4.2 Cartridge Connector - J4

The TS2068 provides a 2 X 18 pin connector (designated J4 on the schematic) under the door at the front right of the console. The table and figures listed below detail the mechanical, functional, and electrical requirements and limits of the J4 Cartridge Connector.

<table>
<thead>
<tr>
<th>FIGURE/TABLE</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.4.2-1</td>
<td>J4 Mating PCB Mechanical Requirements</td>
</tr>
<tr>
<td>Figure 2.4.2-2</td>
<td>J4 Signal Layout</td>
</tr>
<tr>
<td>Table 2.4.2-1</td>
<td>J4 Signal Definition</td>
</tr>
<tr>
<td>Table 2.4.2-2</td>
<td>J4 Signal Electrical Characteristics</td>
</tr>
</tbody>
</table>
FIGURE 2.4.2-1
J4 MATING PCB MECHANICAL REQUIREMENTS

NOTES:

(1) Circuit Board Material:
   FLGPN C62
   C1/1A2A (94V-0)
   Copper 1 or 2 sides
(2) Contact Fingers: Min. 10
   millionth MIL-G - 45204 Gold over
   .00005 to .00010 inch low stress
   nickel.
(3) Contact Fingers 2 and 36
   should be longer than other
   fingers to latch-up when inserted
   with power on.

FIGURE 2.4.2-2
J4 SIGNAL LAYOUT
(View from Front)
<table>
<thead>
<tr>
<th>PIN #</th>
<th>SIGNAL NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A14B</td>
<td>Address Bus Bit 14, Buffered</td>
</tr>
<tr>
<td>2</td>
<td>+5V</td>
<td>+5 volts DC</td>
</tr>
<tr>
<td>3</td>
<td>A12</td>
<td>Address Bus Bit 12</td>
</tr>
<tr>
<td>4</td>
<td>A13B</td>
<td>Address Bus Bit 13, Buffered</td>
</tr>
<tr>
<td>5</td>
<td>D0</td>
<td>Data Bus Bit 0</td>
</tr>
<tr>
<td>6</td>
<td>D7</td>
<td>Data Bus Bit 7</td>
</tr>
<tr>
<td>7</td>
<td>D1</td>
<td>Data Bus Bit 1</td>
</tr>
<tr>
<td>8</td>
<td>A0</td>
<td>Address Bus Bit 0</td>
</tr>
<tr>
<td>9</td>
<td>D2</td>
<td>Data Bus Bit 2</td>
</tr>
<tr>
<td>10</td>
<td>A1</td>
<td>Address Bus Bit 1</td>
</tr>
<tr>
<td>11</td>
<td>D6</td>
<td>Data Bus Bit 6</td>
</tr>
<tr>
<td>12</td>
<td>A2</td>
<td>Address Bus Bit 2</td>
</tr>
<tr>
<td>13</td>
<td>D5</td>
<td>Data Bus Bit 5</td>
</tr>
<tr>
<td>14</td>
<td>A3</td>
<td>Address Bus Bit 3</td>
</tr>
<tr>
<td>15</td>
<td>D3</td>
<td>Data Bus Bit 3</td>
</tr>
<tr>
<td>16</td>
<td>A15B</td>
<td>Address Bus Bit 15, Buffered</td>
</tr>
<tr>
<td>17</td>
<td>D4</td>
<td>Data Bus Bit 4</td>
</tr>
<tr>
<td>18</td>
<td>MREQB</td>
<td>Memory Request (Active Low), Bfrd.</td>
</tr>
<tr>
<td>19</td>
<td>TORD</td>
<td>I/O Request (Active Low), Buffered</td>
</tr>
<tr>
<td>20</td>
<td>A7RB</td>
<td>Refresh Address Bit 7, Buffered</td>
</tr>
<tr>
<td>21</td>
<td>RDW</td>
<td>Read (Active Low), Buffered</td>
</tr>
<tr>
<td>22</td>
<td>MT</td>
<td>CPU M1 State (Active Low)</td>
</tr>
<tr>
<td>23</td>
<td>WRW</td>
<td>Write (Active Low), Buffered</td>
</tr>
<tr>
<td>24</td>
<td>A8</td>
<td>Address Bus Bit 8</td>
</tr>
<tr>
<td>25</td>
<td>A7</td>
<td>Address Bus Bit 7</td>
</tr>
<tr>
<td>26</td>
<td>A9</td>
<td>Address Bus Bit 9</td>
</tr>
<tr>
<td>27</td>
<td>A6</td>
<td>Address Bus Bit 6</td>
</tr>
<tr>
<td>28</td>
<td>A10</td>
<td>Address Bus Bit 10</td>
</tr>
<tr>
<td>29</td>
<td>A5</td>
<td>Address Bus Bit 5</td>
</tr>
<tr>
<td>30</td>
<td>A11</td>
<td>Address Bus Bit 11</td>
</tr>
<tr>
<td>31</td>
<td>A4</td>
<td>Address Bus Bit 4</td>
</tr>
<tr>
<td>32</td>
<td>RFSHB</td>
<td>Refresh (Active Low), Buffered</td>
</tr>
<tr>
<td>33</td>
<td>BE</td>
<td>Bank Enable (Active Low)</td>
</tr>
<tr>
<td>34</td>
<td>EXROM</td>
<td>Extension ROM Enable (Active Low)</td>
</tr>
<tr>
<td>35</td>
<td>ROSCS</td>
<td>ROS Chip Select (Active Low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Dock Bank Enable)</td>
</tr>
<tr>
<td>36</td>
<td>GND</td>
<td>Ground</td>
</tr>
</tbody>
</table>

60
### TABLE 2.4.2-2

**J4 SIGNAL ELECTRICAL CHARACTERISTICS**

<table>
<thead>
<tr>
<th>MNEMONIC</th>
<th>V(OL) MAX (VOLTS)</th>
<th>I(LOAD) MAX (MA)</th>
<th>V(OH) MIN (VOLTS)</th>
<th>I(LOAD)*V(IL) MAX (\mu A)</th>
<th>V(IH) MIN (VOLTS)</th>
<th>I IN (MAX) ( \mu A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A15B</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A14B</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A13B</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A12</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A11</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A10</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A8</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A7</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A0</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>A78B</td>
<td>30</td>
<td>0.5</td>
<td>0.35</td>
<td>2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROSCS</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>MREQB</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>RD8</td>
<td>30</td>
<td>0.5</td>
<td>1.8</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>I0RQB</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>WRB</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>NSFHB</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>EXROM</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>MT</td>
<td>30</td>
<td>0.5</td>
<td>12</td>
<td>2.4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>D0</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D1</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D2</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D3</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D4</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D5</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D6</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>D7</td>
<td>30</td>
<td>0.4</td>
<td>1.8</td>
<td>2.4</td>
<td>0.8</td>
<td>2.0</td>
</tr>
<tr>
<td>Vcc (+5V)</td>
<td>--</td>
<td>5.25</td>
<td>300</td>
<td>4.75</td>
<td>--</td>
<td>---</td>
</tr>
<tr>
<td>GND</td>
<td>--</td>
<td>--</td>
<td>---</td>
<td>---</td>
<td>--</td>
<td>---</td>
</tr>
</tbody>
</table>
2.4.3 Cassette I/O

The EAR and MIC connectors provided on the rear of the TS2068 are 1/8" mini-phone jacks requiring 1/8" mini-phone plugs as mating connectors.

The MIC output is filtered by a low-pass filter with a breakpoint of 2.5KHz and provides a signal output of 0.15 to 0.67 V p-p.

The EAR input is filtered by a low-pass filter with a breakpoint of 23 KHz. Input voltages should be between 4.0 and 10.0 V p-p.

2.4.4 Joystick

The joystick input connectors, one on each side of the TS2068 case, are standard 9-pin "D" type connectors for use with 5-switch type joysticks.

Connector layout and the function of each pin is given in Figure 2.4.4-1 and Table 2.4.4-1, respectively.

FIGURE 2.4.4-1

JOYSTICK CONNECTOR
# TABLE 2.4.4-1

JOYSTICK CONNECTOR SIGNAL ASSIGNMENT

<table>
<thead>
<tr>
<th>P/N</th>
<th>SIGNAL NAME</th>
<th>I/O PORT BIT</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DIRT</td>
<td>0</td>
<td>STICK UP</td>
</tr>
<tr>
<td>2</td>
<td>DIR2</td>
<td>1</td>
<td>STICK DOWN</td>
</tr>
<tr>
<td>3</td>
<td>DIR3</td>
<td>2</td>
<td>STICK LEFT</td>
</tr>
<tr>
<td>4</td>
<td>DIR4</td>
<td>3</td>
<td>STICK RIGHT</td>
</tr>
<tr>
<td>5</td>
<td>---</td>
<td>---</td>
<td>not used</td>
</tr>
<tr>
<td>6</td>
<td>BUTTON</td>
<td>7</td>
<td>PUSH BUTTON</td>
</tr>
<tr>
<td>8</td>
<td>5V</td>
<td>---</td>
<td>5 VOLT POWER</td>
</tr>
<tr>
<td>8</td>
<td>READ STROBE</td>
<td>---</td>
<td>ADDRESS BIT 8 OR 9*</td>
</tr>
<tr>
<td>9</td>
<td>GND</td>
<td>---</td>
<td>POWER GROUND</td>
</tr>
</tbody>
</table>

*When Address Bit 8 is high, the READ strobe to the left joystick is driven low. When address Bit 9 is high, the READ strobe to the right joystick is driven low.

## 2.4.5 AC Adapter Power Plug

The AC Adapter provided with the TS 2068 provides unregulated DC to the unit as described in Section 2.1.1 Mechanical details of the plug which mates to the TS 2068 are shown below:

![MINI-POWER PLUG DETAIL](image)

---

63
2.4.6 Composite Monitor Output

The MONITOR output on the rear of the TS2068 provides a 1 V p-p (+/- 20%) composite color video signal output to an RCA phono jack which is mated by a standard phono plug into a 75 ohm coax cable. See Section 2.1.11.1.

2.4.7 RF Output

The TV output on the rear of the TS2068 provides a modulated color video signal on VHF Channel 2 or Channel 3 as selected by the channel select switch on the bottom of the unit. Connection to the RCA phono jack output should be via a standard phono plug and 75 ohm coax cable. See Section 2.1.11.2.

Channel frequencies provided are

Channel 2  55,250 +/- 100 KHz
Channel 3  61,250 +/- 100 KHz

Output levels are less than 3 milliwatts as limited by the Federal Communications Commission.

2.4.8 Keyboard Interface - J9 Connector

Located on the PCB inside the TS 2068 is a 14-pin single-in-line flex cable connector (AMP TRIO-MATE P/N 1-520315-4 or equivalent). Signals are as listed below:

<table>
<thead>
<tr>
<th>PIN</th>
<th>SIGNAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>GND</td>
</tr>
<tr>
<td>1</td>
<td>KB0</td>
</tr>
<tr>
<td>2</td>
<td>KB1</td>
</tr>
<tr>
<td>3</td>
<td>KB2</td>
</tr>
<tr>
<td>4</td>
<td>KB3</td>
</tr>
<tr>
<td>5</td>
<td>KB4</td>
</tr>
<tr>
<td>6</td>
<td>CR6/A11</td>
</tr>
<tr>
<td>7</td>
<td>CR7/A10</td>
</tr>
<tr>
<td>8</td>
<td>CR8/A9</td>
</tr>
<tr>
<td>9</td>
<td>CR9/A12</td>
</tr>
<tr>
<td>10</td>
<td>CR10/A13B</td>
</tr>
<tr>
<td>11</td>
<td>CR11/A8</td>
</tr>
<tr>
<td>12</td>
<td>CR12/A14B</td>
</tr>
<tr>
<td>13</td>
<td>CR13/A15B</td>
</tr>
</tbody>
</table>

Any modification to or replacement of the keyboard supplied must consider the following:

(1) Contact resistance less than 200 ohms.
(2) Bounce less than 10 ms.
(3) Capacitance per line less than 20 pF (0 or 1 key depressed); less than 40 pF (more than 1 key depressed).
3.0 SYSTEM SOFTWARE GUIDE

3.1 Identifier

Location 19 (13H) of the Home Bank ROM is used to identify the revision level of the System Software. The initial version is identified by this location having a value of 255 (FFH). Any subsequent versions will decrement this value by 1, e.g., the first revision would be identified by a value of 254 (FEH). This identifier should be used to conditionally apply patches or execute "work-arounds" identified as necessary with a particular version of the System Software.

3.2 ROM Organization and Services

3.2.1 Home ROM

3.2.1.1 Fixed Entry Points

Home ROM Location 0 is the entry to the system initialization code upon power-up (Ref. Figure 1.1-4). Locations 8 through 48 (8H through 30H) are the Z80 RESTART entry points for the following functions:

<table>
<thead>
<tr>
<th>RESTART</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>ERROR - Error exit from BASIC (Address on Stack points to Error Number)</td>
</tr>
<tr>
<td>15</td>
<td>WRCH - Write Character (Code in A) to Current Output Channel as established by SELECT (Address of output routine pointed to by System Variable CURCHL). (See Section 4.0).</td>
</tr>
<tr>
<td>24</td>
<td>IGN SP - Return in A the current significant character in the Program Line (Address in System Variable CH_ADD) skipping over spaces and control characters except End-of-Line (ODH=ENTER)</td>
</tr>
</tbody>
</table>
32 NXT_IS - Like IGN_SP but returns in _A the Next Significant Character.

40 CALCTR - Entry to Calculator Routines.

48 COPYUP - Make room for BC Bytes of temporary workspace just before address in System Variable STKBOT by copying up memory between there and the address in STKEND, adjusting affected pointers. Returns DE=1st Byte of Space; HL=Last.

Location 56 (38H) is the entry to service the hardware generated interruption which occurs approximately every 1/60 of a second (16.67 ms). 280 Int. Mode 1 is used. This interruption is used to scan the keyboard (call to routine UPD_K - see Section 4.1.1). It is also used to update the Frame Counter (3 bytes pointed to by the System Variable FRAMES) used by the RANDOMIZE instruction.

Location 102 (66H) is the entry point for the NMI interruption, but this interruption is not used in the TS2068 design. (See Section 2.1.3.8 NMI Interruption.)

3.2.1.2 BASIC AROS Support

BASIC Application Cartridges are supported by special code in the Home ROM. A program line is copied from the cartridge to a buffer in the Home RAM (ARSBUF) and is then executed from there by the BASIC Interpreter. When a READ command is executed, the line containing the appropriate DATA statement is also copied from the cartridge to the RAM. The cartridge memory is enabled only for search and copy operations for both program lines and DATA statements, and when executing a USR function, otherwise the entire Home Bank is enabled while executing in the BASIC Interpreter. There is no support for User-Defined Functions which insert the expanded definition parameters directly into the program and then require search of the program area to find these parameters whenever a function is invoked.

See Section 5.1, Cartridge Software/Hardware, for additional details on BASIC AROS.
3.2.1.3 General

The balance of the Home ROM contains the BASIC Interpreter and standard I/O routines with the exception of the cassette I/O which is in the Extension ROM. The bit map table for the standard character set is located at the end of the Home ROM from location 15616 to 16383 (3000H to 3FFFH). The address of this table minus 256 (100H) is contained in the System Variable CHARS (=3COOH).

The Home ROM routines accessible via the Function Dispatcher are described in Table 3.3.4-2. See Appendix A for the ROM Maps giving the ROM addresses of these routines.

3.2.2 Extension ROM

3.2.2.1 Fixed Entry Points

Extension ROM Location 0 contains code to pass control to the initialization code in the Home ROM. (Figure 1.1-4).

Extension ROM Location 56 (38H) is the interruption fielder. Control is passed to the System RAM code (See Section 3.3.3) to bank switch to the Home Bank and call the interruption service routines after which the state of the machine is restored and control returns to the interrupted process. Figure 3.2.2-1 shows the Extension ROM Interruption Fielder code.

3.2.2.2 General

The balance of the Extension ROM contains the following major components:

- Final Phase of System Initialization (See Figure 1.1-4)
- Cassette tape I/O (see Section 4.2)
- Change Video Mode Service
- OS RAM routines including the Function Dispatcher (copied to RAM at System Initialization) (see Section 3.3.3)
- Function Dispatcher Jump Table
FIGURE 3.2.2-1

Extension ROM Interruption Fielder

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>OBJECT CODE</th>
<th>SOURCE CODE</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0038</td>
<td>F5</td>
<td>PUSH AF</td>
<td>Save AF</td>
</tr>
<tr>
<td>0039</td>
<td>F3</td>
<td>DI</td>
<td>Disable Ints.</td>
</tr>
<tr>
<td>003A</td>
<td>3AC25C</td>
<td>LD A,(VIDMOD)</td>
<td>Test Vidmod</td>
</tr>
<tr>
<td>003D</td>
<td>A7</td>
<td>AND A</td>
<td></td>
</tr>
<tr>
<td>003E</td>
<td>00</td>
<td>NOP</td>
<td></td>
</tr>
<tr>
<td>003F</td>
<td>2804</td>
<td>JR Z,CHK3</td>
<td>Vidmod=0</td>
</tr>
<tr>
<td>0041</td>
<td>F1</td>
<td>POP AF</td>
<td>Restore AF</td>
</tr>
<tr>
<td>0042</td>
<td>C36EFA</td>
<td>JP INT7</td>
<td>Chunk 7 if Vidmod not 0</td>
</tr>
<tr>
<td>0045</td>
<td>F1</td>
<td>CHK3 POP AF</td>
<td>Restore AF</td>
</tr>
<tr>
<td>0046</td>
<td>C3AE62</td>
<td>JP INT3</td>
<td>Chunk 3 if Vidmod = 0</td>
</tr>
</tbody>
</table>

3.2.2.3 Video Mode Change Service

The routine CHNG VID takes as input a single byte in Register A which designates the desired video mode as shown in Table 3.2.2-1. All non-zero values involve access to the second display file located at 60000H-7AFFH. When the mode change requires remapping of the RAM (see Figure 1.1-3), the necessary relocation (BASIC program, machine stack, OS RAM code, UDG area, etc.) and modifications (system variables, RAM code internal addresses, stack pointer, etc.) are done by this service. The desired video mode is written to Port OFFH, Bits 0-5, and the System Variable VIDMOD (5CC2H) is updated. The second display file is cleared to zeros on initial access (for Dual Screen Mode and High Resolution Graphics Mode, this results in a black screen since 0 yields attributes of black ink on black paper). If there is not enough free memory to do the necessary remapping, Error 4, Out of Memory is given.

Access to this service via the Function Dispatcher cannot be made consistently for various reasons. An Interface Routine is given in Section 3.2.2.4, to be executed from the Home RAM, which provides access to the Video Mode Change Service as well as other Extension ROM routines.

See Sections 4.1.2 and 5.2 for discussion of video screen support software. See Section 6.4 for details on known problems and corrections related to the Video Mode Change Service.
TABLE 3.2.2-1

INPUT TO VIDEO MODE CHANGE SERVICE

<table>
<thead>
<tr>
<th>VALUE IN A</th>
<th>VIDEO MODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal</td>
<td>Primary Display File Only (Close 2nd Display File if Open)</td>
</tr>
<tr>
<td>128 (80H)</td>
<td>Dual Screen</td>
<td>Two Display Files Available. Primary Display File Active at Screen.</td>
</tr>
<tr>
<td>1</td>
<td>Dual Screen</td>
<td>Two Display Files Available. Second Display File Active at Screen</td>
</tr>
<tr>
<td>2</td>
<td>High Resolution Graphics</td>
<td>Primary Display File contains data for 256X192 pixels. Second Display File contains 6144 Attribute Bytes, each one controlling 8X1 pixels. NOTE 1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>64-Column Inks</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Black White</td>
</tr>
<tr>
<td>14 (0EH)</td>
<td>Blue Yellow</td>
</tr>
<tr>
<td>22 (16H)</td>
<td>Red Cyan</td>
</tr>
<tr>
<td>30 (1EH)</td>
<td>Magenta Green</td>
</tr>
<tr>
<td>38 (26H)</td>
<td>Green Magenta</td>
</tr>
<tr>
<td>46 (2EH)</td>
<td>Cyan Red</td>
</tr>
<tr>
<td>54 (36H)</td>
<td>Yellow Blue</td>
</tr>
<tr>
<td>62 (3EH)</td>
<td>White Black</td>
</tr>
</tbody>
</table>

NOTE 1: The areas of memory normally used for Attribute Bytes are not accessed by the video hardware in this mode.
3.2.2.4 Extension ROM Interface Routine

The Extension ROM routines W TAPE (Write from RAM to Tape), R-TAPE (Read from Tape to RAM) (see Section 4.2) and CHNG VID (see Section 3.2.2.2) may be of interest to the machine code programmer. Because of a conflict with the use of the IX Register, the tape routines cannot be successfully accessed via the Function Dispatcher. Because the Change Video Mode Service may involve relocating the OS RAM routines (including the Function Dispatcher), and for other reasons, it also cannot be consistently accessed using the Function Dispatcher. Figure 3.2.2-2 gives a sample routine, to be executed from the Home RAM, which can be used to bank switch to the Extension ROM and call directly to the desired service. Appendix A contains an Extension ROM Map giving the addresses of these and other routines.
FIGURE 3.2.2-2
EXTENSION ROM INTERFACE ROUTINE

```
0000" 21 00FC
0003" CC 002D"
0006" 18 17

0008" 21 0068
000B" CD 002D"
000E" 18 0F

0010" 21 0E8E
0013" F5
0014" CD 002D"

0017" F1
0018" FE 80
001A" 20 03
001C" 32 5CC2
001F" 3A 002C"
0022" D3 F4
0024" DB FF
0026" CB BF
0028" FB FF
002A" FB C9

002C" 00
002D" F3
002E" F5
002F" DB FF
0031" CB FF
0033" D3 FF
0035" D8 FA
0037" 32 002C"
003A" 3E 01
003C" D3 FA
003E" F1
003F" E9

1: R_TAPE EQU 00FCM
2: W_TAPE EQU 0068M
4: CHNG_VID EQU 00EBM
5: VIDMOD EQU 5CC2M

11: READTP LD ML,R_TAPE
12: CALL IFRTN
13: JR EXIT

18: WRITTP LD ML,W_TAPE
19: CALL IFRTN
20: JR EXIT

25: CHNGVID LD ML,CHNG_VID
26: PUSH AF
27: CALL IFRTN

28: CP AF
29: CP 80H
30: JR NZ,EXIT
31: LD (VIDMOD),A
32: EXIT
33: LD A,(CHNGVID)
34: ADD A,(OFFHM)
35: CALL IFRTN

36: POP AF
37: CP 80H
38: JR NZ,EXIT
39: LD (VIDMOD),A
40: EXIT
41: LD A,(MSSAVE)
42: ADD A,(OFFHM)
43: RES 7,A
44: OUT (OFFHM),A
45: EI
46: RET

47: MSSAVE DEPB 0
48: IFRTN DI
49: PUSH AF
50: IN A,(OFFHM)
51: SET T,A
52: OUT (OFFHM),A
53: IN A,(OFFHM)
54: LD (MSSAVE),A
55: LD A,1
56: OUT (OFFHM),A
57: POP AF
58: JP (ML)
59: END

71
3.3 RAM Organization and Services

3.3.1 System Variables

RAM beginning at 23552 (5C00H) is dedicated to the BASIC System Variables as defined in Appendix D of the TS 2068 User Manual and in Appendix B of this document. The area from the end of the defined variables (STRTNM - 23755 (5CBB)) to 24297 (5EE9H) is reserved for expansion of the System Variables, but is not used by the Operating System in the current TS 2068.

3.3.2 System Configuration Table

The area from 24298 (5EEAH) to 24575 (5FFFH) is reserved for the System Configuration Table (SYSCON). This table is built at system initialization time and is comprised of an 8 byte entry for AROS, a 4 byte entry for LROS, followed by eleven 24-byte entries for proposed expansion banks and an End-of-Table marker. In the original TS 2068 the actual usage of this table is limited to the 12 bytes for software cartridge identification (see Section 5.1 for details of the LROS and AROS Overhead Bytes).

3.3.3 Machine Stack

The TS 2068 reserves 512 (200H) bytes of RAM for the Machine Stack. The Machine Stack pointer is initialized to a value of 6200H (value also in System Variable MSTBOT); the pointer is decremented as items are pushed onto the stack (the pointer may also be modified directly by software). While the area reserved for the stack extends to 6000H, there is no actual check made to enforce this limit.

Note that the Machine Stack is located in the same memory area as the second display file. The CHNG_VID routine relocates the stack to the memory area from OF7COH to OF8BFH, and modifies the Stack Pointer and MSTBOT (OF8COH), as well as other affected system variables, when initializing the second display file. (See Section 3.2.2.3.)

3.3.4 OS RAM Routines

The code for the following Operating System functions is copied from the Extension ROM to Chunk 3 of the RAM at System initialization time. Since this is in the same memory area as the second display file, this code must be relocated, along with the machine stack, if the second display file is to be used. The CHNG_VID routine does the necessary relocation and modifications. (Section 3.2.2.3.)
Because this code is not in a fixed location, access to the OS RAM routines is conditional on the current video mode. The standard technique employed is to test the value in the System Variable VIDMOD at location 23746 (5CC2H). A zero indicates that the second display file is not in use and that the OS RAM routines are therefore in Chunk 3; any non-zero value indicates that the routines are in Chunk 7.

NOTE: This design implies that Chunks 2, 3 and 7 are always enabled in the Home Bank RAM whenever the System ROM and/or RAM routines are being used.

The OS RAM routines are contained in Module "Dispatch" which is included in Appendix A.

3.3.4.1 RAM Interruption Handler

Chunk 3 Entry: 62AEH

Chunk 7 Entry: FA6EH

The user must enter with bank status and Z80 registers intact, with address from point of interruption on the stack.

The RAM interruption handler saves state, including memory selection, enables the Home Bank, updates the Frame Counter, calls the keyboard scan routine in the Home ROM, restores state, and returns to the interrupted process.

The RAM Interruption handler is used whenever the interruption occurs while the Extension ROM is enabled. See Figure 3.2.2-1, Extension ROM Interruption Fielder. This same technique can be used for interruption processing in another bank, e.g. if an LROS wanted to use the standard system ROM keyboard scanning routines.

3.3.4.2 RAM Service Routines

Table 3.3.4-1 lists the RAM service routines which are designed to facilitate communication between memory banks. Those with Service Codes are accessible via the Function Dispatcher.
### TABLE 3.3.4-1

**OS RAM SERVICE ROUTINES**

<table>
<thead>
<tr>
<th>Label</th>
<th>Service Code</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET _WORD</td>
<td>6316</td>
<td>FAD6</td>
<td>Returns in HL the word from the address in HL in the bank specified in B.</td>
</tr>
<tr>
<td>PUT _WORD</td>
<td>633B</td>
<td>FAFB</td>
<td>Writes the word in DE to the address in HL in the bank specified in B.</td>
</tr>
<tr>
<td>GET _STATUS</td>
<td>14</td>
<td>6405</td>
<td>FBC5 Returns current memory selection (Horizontal Select byte - Low active) in C for the bank specified in B. Preserves Bank # in B for Home, Ext. or Dock.</td>
</tr>
<tr>
<td>GET _CHUNK</td>
<td>-</td>
<td>644D</td>
<td>FCOD Returns a single byte mask in A with all bits 0 except the one corresponding to the chunk for the address in HL.</td>
</tr>
<tr>
<td>GET _NUMBER</td>
<td>15</td>
<td>645E</td>
<td>FC1E Returns in Reg. A the bank number currently controlling the address in HL.</td>
</tr>
<tr>
<td>BANK _ENABLE</td>
<td>-</td>
<td>6499</td>
<td>FC59 Enables the memory selected (Horizontal Select byte - Low active) in the specified bank. (Bank # in B; Mem.Sel.in C)</td>
</tr>
<tr>
<td>GOTO _BANK</td>
<td>-</td>
<td>6572</td>
<td>FD32 Transfers control to the specified address after enabling the memory selected in the specified bank. Parameters passed on stack by pushing target address, then Bank #/Mem.Select prior to calling GOTO _BANK. (Return address is discarded.).</td>
</tr>
<tr>
<td>CALL _BANK</td>
<td>-</td>
<td>6500</td>
<td>FD90 Like GOTO _BANK except saves current bank status, calls target address, and restores status prior to returning to user. Two additional parameters are passed on stack prior to doing call to CALL _BANK. These are PRM _OUT (16-5Tt) following by PRM _IN (16 bits) as described for the Function Dispatcher.</td>
</tr>
<tr>
<td>LABEL</td>
<td>SERVICE CODE</td>
<td>LOCATION</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>XFER_BYTES</td>
<td>-</td>
<td>6722</td>
<td>FEE2</td>
</tr>
</tbody>
</table>

Copies n byte(s) from specified source to specified destination in either ascending or descending order. Source and destination can be in the same or different banks and can be in shadowing chunks, but neither source nor destination can pass a "chunk" (8K) boundary since only the chunks containing the starting source and destination addresses are explicitly enabled.

Parameters passed on stack by pushing:

- Source Bank/Dest.Bank
- Source Address
- Dest. Address
- Length
- O/Direction:
  - 0=Ascending
  - -1=Descending

NOTE: See Appendix A for listing of these routines. See Section 6.0 for known corrections to the routines.
3.3.4.3 Function Dispatcher

Chunk 3 Entry: 6200H

Chunk 7 Entry: F9COH

The Function Dispatcher provides a common interface to a number of system routines via a Service Code and Jump Flag parameter passed on the machine stack. Table 3.3.4-2 lists the routines in Service Code order. Codes for routines that are known to not be successfully accessible via the Function Dispatcher have been deleted (marked Reserved). However, there is no guarantee that those on the list can be accessed without problems. Some ROM routines require data in a particular format, e.g. BASIC floating point number(s), both standard and special integer format, on the Calculator Stack which is located between (STKBOT) and (STKEND) (see Appendix C of the TS 2068 User Manual). An effort has been made to include information on register usage and functionality, but some of the ROM routines are so tightly tied to the BASIC Interpreter that they would require analysis which is beyond the scope of this document. These have been flagged with an Asterisk, but included in the list for documentation purposes only. Most of the routines which are directly implementing a BASIC command or function have two different action sequences based on the INTPT Flag (Bit 7 of FLAGS) which distinguishes syntax checking (Flag=0) from actual execution (Flag=1).

In order to use the Function Dispatcher, first set up any memory and stack (both machine and/or calculator) locations as if invoking the desired service directly. Then push the parameter(s) for the Dispatcher on the machine stack in the order outlined below. Finally, set up the registers as if invoking the desired service directly and call the Dispatcher based on its current location (Chunk 3 if VIDMOD=0 or Chunk 7 if VIDMOD has a non-zero value).

1. **PRM_OUT** 16 bits - Number of bytes of parameter data being passed on the stack to the specified Service (number of stack "pushes" * 2). Zero if no parameters being passed. E.g., to pass 4 bytes:
LD HL,4
PUSH HL

This parameter is passed to the Dispatcher only if the Jump Flag (SVC_CODE) Bit 15) is not set. NOTE: This parameter refers to machine stack entries only, not to the Calculator Stack.

2. PRM_IN
   16 bits - Number of bytes of parameter data to be passed back from the specified Service (number of stack "pushes" * 2). Zero if no parameters to be passed back.

This parameter is passed to the Dispatcher only if the Jump Flag (SVC_CODE Bit 15) is not set. NOTE: This parameter refers to machine stack entries only, not to the Calculator Stack.

3. SVC_CODE
   16 bits - Bits 0-14 identify the Service to be invoked. Bit 15 (Jump Flag) is set if no return is desired (jump to Service rather than call). Bit 15 is zero if return is desired. E.g., to call K_SCAN using Service Code 136:

   LD HL,136 or LD HL,88H
   PUSH HL

Addendum To TS 2068 Function Dispatcher Services:
On page 84, COLOR and HIFLASH (service codes 85 and 86) cannot always be accessed through the Function Dispatcher, due to resetting of the carry flag by the FD. COLOR may be accessed by setting the registers as described in the manual, and then coding CALL #23DE. HIFLASH can be accessed similarly by coding CALL #2410.
<table>
<thead>
<tr>
<th>SERVICE</th>
<th>SERVICE CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_STATUS</td>
<td>14 (0EH)</td>
<td>Returns Memory Selection (Low Active) in C for Bank # in B</td>
</tr>
<tr>
<td>GET_NUMBER</td>
<td>15 (0FH)</td>
<td>Returns Bank # in A for Address in HL</td>
</tr>
<tr>
<td>16-24</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>UPD_K</td>
<td>25 (19H)</td>
<td>Process Keyboard Input (See Section 4.1.1)</td>
</tr>
<tr>
<td>PARP</td>
<td>26 (1AH)</td>
<td>Generates DE+1 Cycles of a Tone having the Period 8N+236 to 8N+246 T-States. HL=N. (See 4.4)</td>
</tr>
<tr>
<td>BEEP</td>
<td>27 (1BH)</td>
<td>BEEP Command - processes parameters on Calculator Stack. Exits via PARP. (See 4.4)</td>
</tr>
<tr>
<td>K_DUMP</td>
<td>28 (1CH)</td>
<td>COPY Command. Dumps Primary Display File to Printer. (See 4.1.3)</td>
</tr>
<tr>
<td>SENDTV</td>
<td>29 (1DH)</td>
<td>Char.Output to Screen/Printer. Character Code in A. (See 4.1.2)</td>
</tr>
<tr>
<td>SETAT</td>
<td>30 (1EH)</td>
<td>Set Print Position to value in BC. B=Line No. (0-23); C=Column No. (0-31)</td>
</tr>
<tr>
<td>ATTBYT</td>
<td>31 (1FH)</td>
<td>Set Attribute Byte for Display File Adrs. in HL using ATTR_T, MASK_T and P_FLAG.</td>
</tr>
<tr>
<td>R_ATTS</td>
<td>32 (20H)</td>
<td>Permanent Attribute Info. to Temporary Attribute Variables</td>
</tr>
<tr>
<td>CLLHS</td>
<td>33 (21H)</td>
<td>Clear Lower Screen (Primary Display File)</td>
</tr>
<tr>
<td>CLS</td>
<td>34 (22H)</td>
<td>Clear Entire Screen (Primary Display File)</td>
</tr>
<tr>
<td>DUMPPR</td>
<td>35 (23H)</td>
<td>Print/Clear Print Buffer. (See 4.1.3)</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PRSCAN</td>
<td>36 (24H)</td>
<td>Send Scan (32 bytes) to Printer. Pixel Data Address in HL. Number of Scans remaining in B (=1-8). (See 4.1.3)</td>
</tr>
<tr>
<td>DESLUG</td>
<td>37 (25H)</td>
<td>Remove Number Slugs from Edit Line Buffer (Address in HL)</td>
</tr>
<tr>
<td>K_NEW</td>
<td>38 (26H)</td>
<td>NEW command. See Fig. 1.1-4</td>
</tr>
<tr>
<td>INIT</td>
<td>39 (27H)</td>
<td>Initialize: DE=Maximum RAM Address. A=0 for Power-On; = -1 (FFH) for NEW. (See Fig.1.1-4)</td>
</tr>
<tr>
<td>INCH</td>
<td>40 (28H)</td>
<td>Input Character to A from currently Selected Channel. Returns NC if no input.</td>
</tr>
<tr>
<td>SELECT</td>
<td>41 (29H)</td>
<td>Select Channel (Stream) - # in A. (See 4.1)</td>
</tr>
<tr>
<td>INSERT</td>
<td>42 (2AH)</td>
<td>Insert BC Bytes before byte whose address is in HL. Copies up all from HL to (STKEND) and updates affected system variables. Returns BC=0; DE=adr.of last byte of inserted space; HL=adr.of byte before first.</td>
</tr>
<tr>
<td>RESET</td>
<td>43 (2BH)</td>
<td>Reset Calculator Stack. Sets (STKEND) = (STKBOT) and (NEM)=MEMBOT (5C92H).</td>
</tr>
<tr>
<td>CLOSE</td>
<td>44 (2CH)</td>
<td>CLOSE # Command. Channel # on Calculator Stack.</td>
</tr>
<tr>
<td>CLCHAN</td>
<td>45 (2DH)</td>
<td>Close Channel. BC=Value from STRMS (Index into CHANS).</td>
</tr>
<tr>
<td>OPEN</td>
<td>46 (2EH)</td>
<td>OPEN # Command. Channel # and Device Spec. on Calculator Stack</td>
</tr>
<tr>
<td>OPCHAN</td>
<td>47 (2FH)</td>
<td>Open Channel. Device Spec. on Calculator Stack. #=pointer into STRMS based on Ch.#.</td>
</tr>
</tbody>
</table>

(See 4.1 for more info. on OPEN and CLOSE)
<table>
<thead>
<tr>
<th>SERVICE</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT</td>
<td>48 (30H)</td>
<td>CAT Command (Not Applicable)</td>
</tr>
<tr>
<td>ERASE</td>
<td>49 (31H)</td>
<td>ERASE Command (Not Applicable)</td>
</tr>
<tr>
<td>FORMAT</td>
<td>50 (32H)</td>
<td>FORMAT Command (Not Applicable)</td>
</tr>
<tr>
<td>MOVE</td>
<td>51 (33H)</td>
<td>MOVE Command (Not Applicable)</td>
</tr>
<tr>
<td>FLASHA</td>
<td>52 (34H)</td>
<td>Flash Char.in A to Screen. (Calls SENDTV; assumes Lower Screen selected. Used to Flash Cursor.)</td>
</tr>
<tr>
<td>FIND_L</td>
<td>53 (35H)</td>
<td>Find BASIC Program Line with the number in HL. If Line found, returns Z and Address of Line in HL, else returns NZ and HL contains either address of line with next larger line number or points to the Variables area if there is no larger line number. Requested Line No. returned in BC and Address of Preceding Line in DE (DE=HL if no preceding line).</td>
</tr>
<tr>
<td>SUBLIN</td>
<td>54 (36H)</td>
<td>Finds either the D'th statement (D=Statement #; E=0) or 1st statement whose keyword token matches E (D=0), in a line pointed to by HL. If the D'th statement is found, returns Z and HL and (CH ADD) both point to 1 byte before statement. (If line contains exactly D-1 statements, then the next line counts as the D'th.). If match on E is found, then returns NZ,NC and both HL and (CH ADD) point to keyword. D is decremented by the number of statements looked at (e.g. D= -2 if two statements). If no match on E then returns NZ,C with both HL and (CH ADD) pointing to End-of-Line byte (ODH).</td>
</tr>
</tbody>
</table>
### TABLE 3.3.4-2

**TS 2068 FUNCTION DISPATCHER SERVICES**

(continued)

<table>
<thead>
<tr>
<th>SERVICE</th>
<th>SERVICE CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECLEN</td>
<td>55 (37H)</td>
<td>Returns in BC the length of the record pointed to by HL. Sets DE to HL+BC. The record can be a program line, or a string or numeric variable or array.</td>
</tr>
<tr>
<td>DELREC</td>
<td>56 (38H)</td>
<td>Delete record pointed to by HL having length BC from Program or Variables memory. Updates affected system variables.</td>
</tr>
<tr>
<td>PUT_BC</td>
<td>57 (39H)</td>
<td>Converts number in BC from binary to ASCII and outputs to currently selected channel. If BC less than 0, outputs a 0.</td>
</tr>
<tr>
<td>SYNTAX</td>
<td>58 (3AH)</td>
<td>Check syntax of command or program line in Edit Line Buffer (E_LINE). ERR_NR= -1 if no errors, otherwise contains Error Number-1.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>59 (3BH)</td>
<td>Execute command(s) from Edit Line buffer.</td>
</tr>
<tr>
<td>FOR</td>
<td>60 (3CH)</td>
<td>FOR command. *</td>
</tr>
<tr>
<td>STOP</td>
<td>61 (3DH)</td>
<td>STOP command. Does RESTART 8 with Error No. 9.</td>
</tr>
<tr>
<td>NEXT</td>
<td>62 (3EH)</td>
<td>NEXT command. *</td>
</tr>
<tr>
<td>READ</td>
<td>63 (3FH)</td>
<td>READ command. *</td>
</tr>
<tr>
<td>DATA</td>
<td>64 (40H)</td>
<td>DATA statement. *</td>
</tr>
<tr>
<td>RESTBC</td>
<td>65 (41H)</td>
<td>RESTORE command - Line No. in BC</td>
</tr>
<tr>
<td>RAND</td>
<td>66 (42H)</td>
<td>RANDomize command. Sets seed for Random Number Generator based on Parameter on Calculator Stack. If parameter is non-zero, value is loaded to SEED; if zero, value in FRAMES is loaded to SEED.</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CON'T</td>
<td>67 (43H)</td>
<td>CONT command. Loads values from OLDPPC and OSPPC to NEWPPC and NSPPC and returns. Inside the BASIC Interpreter, this results in executing from Line No. in NEWPPC, Statement No. in NSPPC.</td>
</tr>
<tr>
<td>JUMP</td>
<td>68 (44H)</td>
<td>Jump to Line - Loads Line Number from Calculator Stack to NEWPPC and sets NSPPC to 0 and returns.</td>
</tr>
<tr>
<td>FIX_U1</td>
<td>69 (45H)</td>
<td>Converts Floating Point number on Calculator Stack to a single byte unsigned binary value in A (uses FP2A). Does RESTART 8 for Error B if number out of range.</td>
</tr>
<tr>
<td>FIX_U</td>
<td>70 (46H)</td>
<td>Converts Floating Point number on Calculator Stack to a 2-byte unsigned binary value in BC (uses FP2BC). Error B if number out of range.</td>
</tr>
<tr>
<td>CLEAR</td>
<td>71 (47H)</td>
<td>CLEAR command. Processes parameter on Calculator Stack to value in BC for CLR_BC.</td>
</tr>
<tr>
<td>CLR_BC</td>
<td>72 (48H)</td>
<td>Value in BC is new RAMTOP. Deletes Variables, clears screen, and Calculator Stack, etc.</td>
</tr>
<tr>
<td>GO_SUB</td>
<td>73 (49H)</td>
<td>GO_SUB command. Inserts a 3-byte GO_SUB Block into the machine stack above the 2 most recent entries. The Block consists of current Line No. (2 bytes) and Statement No. (1 byte) to be used when RETURN is executed. Then calls JUMP to process GO_SUB parameter and returns. At return to caller, machine stack consists of top of stack at point GO SUB was called, followed by 3-byte entry (Line No. MSB/Line No. LSB/Statement No.).</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CHK_SZ</td>
<td>74 (4AH)</td>
<td>Checks if room for BC + 80 (50H) bytes between (STKEND) and (RAMTOP). Addition of 80 bytes is &quot;left-over&quot; from Spectrum to guarantee minimum machine stack where the stack was at the top of RAM. Error 4 if not enough room.</td>
</tr>
<tr>
<td>RETURN</td>
<td>75 (4BH)</td>
<td>RETURN command. Retrieves most recent GO SUB Block from Machine Stack (SP+4), loads data to NEWPPC and NSPPC and returns. Error 7 if MSB Line No.=3EH (End of Stack Marker).</td>
</tr>
<tr>
<td>PAUSE</td>
<td>76 (4CH)</td>
<td>PAUSE command. Processes parameter on Calculator Stack to BC then waits BC frames or until key is depressed. (Uses HALT instruction, so interruptions must be enabled.)</td>
</tr>
<tr>
<td>BREAK?</td>
<td>77 (4DH)</td>
<td>Reads BREAK key. Returns NC if it is pressed and ON ERROR is not active.</td>
</tr>
<tr>
<td>DEF</td>
<td>78 (4EH)</td>
<td>Define Function.*</td>
</tr>
<tr>
<td>K_LPR</td>
<td>79 (4FH)</td>
<td>LPRINT - Selects Channel 3 and processes items in LPRINT statement for output via WRCH.</td>
</tr>
<tr>
<td>K_PRIN</td>
<td>80 (50H)</td>
<td>PRINT - Selects Channel 2 and processes items in PRINT statement for output via WRCH (same code used for K_LPR).</td>
</tr>
<tr>
<td>P_SEQ</td>
<td>81 (51H)</td>
<td>Code used by K_LPR and K_PRIN to process output data and controls in BASIC statement (address in CH ADD).</td>
</tr>
<tr>
<td>INPUT</td>
<td>82 (52H)</td>
<td>INPUT command. Selects Channel 1 and processes I/O for Keyboard/Lower Screen using a buffer at (WORKSP) for input. *</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>I_SEQ</td>
<td>83 (53H)</td>
<td>Code used by INPUT to process input items and controls in BASIC statement (address in CH_ADD).</td>
</tr>
<tr>
<td>NOTKB?</td>
<td>84 (54H)</td>
<td>Returns Z if current channel is Keyboard/Lower Screen (device specification=&quot;K&quot;).</td>
</tr>
<tr>
<td>COLOR</td>
<td>85 (55H)</td>
<td>Adjusts system variables ATTR_T, MASK_T and P_FLAG for color code in D (0-9). Enter with C set to set Ink or NC set to set Paper. Error K if D is invalid.</td>
</tr>
<tr>
<td>HIFLUSH</td>
<td>86 (56H)</td>
<td>Adjusts system variables (ATTR_T and MASK_T) for Flash/Bright code in D (0, 1 or 8) else Error K. Enter with C for Flash or NC for Bright.</td>
</tr>
<tr>
<td>SCRMBL</td>
<td>87 (57H)</td>
<td>Returns in HL the primary display file address for the pixel with coordinates in BC (B=Y; C=X). Returns in A the bit no (0-7) where 0=lethand or most significant bit. Error B if Y is greater than 175.</td>
</tr>
<tr>
<td>PLOT</td>
<td>88 (58H)</td>
<td>PLOT command. Processes X/Y parameters on the Calculator Stack to BC for plotting of pixel via PLOTBC.</td>
</tr>
<tr>
<td>PLOTBC</td>
<td>89 (59H)</td>
<td>Deals with pixel for coordinates in BC (B=Y; C=X). Processes using P_FLAG for Inverse and Over attributes. Updates Attribute File and sets COORDS=BC.</td>
</tr>
<tr>
<td>GET_XY</td>
<td>90 (5AH)</td>
<td>Converts a pair of numbers from the Calculator Stack to 2 single byte numbers. Top number goes to B and second to C. D=sign of B and E=sign of C (+1 or -1). Used by PLOT and other routines.</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CIRCLE</td>
<td>91 (58H)</td>
<td>CIRCLE command. Calculates successive plot positions from the parameters in the BASIC statement. *</td>
</tr>
<tr>
<td>DRAW</td>
<td>92 (5CH)</td>
<td>DRAW command. Calculates successive plot positions from the parameters in the BASIC statement. *</td>
</tr>
<tr>
<td>DRAW_L</td>
<td>93 (5DH)</td>
<td>Plots a straight line from current position (COORDS) based on parameters from Calculator Stack (X,Y). *</td>
</tr>
<tr>
<td>EXPRN</td>
<td>94 (5EH)</td>
<td>Evaluates expression in BASIC program line (CH_ADD), putting value on Calculator Stack. *</td>
</tr>
<tr>
<td>F_SCRN</td>
<td>95 (5FH)</td>
<td>SCREEN$ function. Matches screen line/col. position (parameters on Calculator Stack) against standard ASCII character set. Returns BC=0 if no find. BC=1 and DE points to Char. Code byte if match found.</td>
</tr>
<tr>
<td>F_ATTRR</td>
<td>96 (60H)</td>
<td>ATTR function. Returns attribute byte value controlling screen pixel position based on parameters on Calculator Stack (X,Y).</td>
</tr>
<tr>
<td>RND</td>
<td>97 (61H)</td>
<td>RND function. Uses value in SEED to generate a pseudo-random number which is placed on the Calculator Stack (Floating Point number).</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>INKEY</td>
<td>99 (63H)</td>
<td>INKEY$ function. Scans keyboard and puts character code byte in (WORKSP) if key detected. In any case, pushes Regs. AEDCB onto Calculator Stack - BC=0 if no input; =1 if char. code stored; DE=address of char. code byte.</td>
</tr>
<tr>
<td>FIND_N</td>
<td>100 (64H)</td>
<td>Find Variable. Searches Variables area for match against identifier pointed to by CH ADD. Adjusts bit NO of FLAGS (Bit 6) for type (1=numeric; 0=string). Also used to find formal parameters for User Defined Functions. *</td>
</tr>
<tr>
<td>PSHSTR</td>
<td>101 (65H)</td>
<td>Push String - Clears bit NO of FLAGS and pushes Regs. AEDCB onto Calculator Stack adjusting (STKNXT) upwards. DE contains address of string; BC contains length.</td>
</tr>
<tr>
<td>PAEDCB</td>
<td>102 (66H)</td>
<td>Same code as for PSHSTR but preserves state of bit NO of FLAGS (Bit 6).</td>
</tr>
<tr>
<td>LET</td>
<td>103 (67H)</td>
<td>LET command. Processes existing or creates new variables. *</td>
</tr>
<tr>
<td>POPSTR</td>
<td>104 (68H)</td>
<td>Pop String - Pops end of Calculator Stack ( (STKNXT)-1 through (STKNXT)-5 ) to Regs. BCDEA, adjusting (STKNXT) downwards.</td>
</tr>
<tr>
<td>DIM</td>
<td>105 (69H)</td>
<td>DIM statement. Creates or initializes numeric or string arrays. *</td>
</tr>
<tr>
<td>STKUSN</td>
<td>106 (6AH)</td>
<td>Stack Unsigned Number - inputs a floating point number onto the Calculator Stack from a series of ASCII characters addressed by (CH ADD). The first character is already in Reg. A (either decimal point, binary token or digit).</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>STK_A</td>
<td>107 (6BH)</td>
<td>1-byte unsigned integer in A to top of Calculator Stack (binary to floating point). Loads O to B and A to C, then executes STK_BC.</td>
</tr>
<tr>
<td>STK_BC</td>
<td>108 (6CH)</td>
<td>2-byte unsigned integer in BC to top of Calculator Stack (binary to floating point).</td>
</tr>
<tr>
<td>ININT</td>
<td>109 (6DH)</td>
<td>Converts a series of ASCII digits pointed to by (CH_ADD) into an unsigned floating point integer on the Calculator Stack. First character is in A on entry. Terminates when non-digit found.</td>
</tr>
<tr>
<td>FP2BC</td>
<td>110 (6EH)</td>
<td>Pops top of Calculator Stack (floating point number) and puts in BC, rounded to nearest integer. Returns NZ if value is negative. Returns C if number exceeded maximum 2-byte value (65535). Range: -65535 to +65535.</td>
</tr>
<tr>
<td>FP2A</td>
<td>111 (6FH)</td>
<td>Pops top of Calculator Stack (floating point number) and puts in A, rounded to nearest integer. Returns NZ if value is negative. Returns C if number exceeded maximum 1-byte value (255). Range: -255 to +255.</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>112 (70H)</td>
<td>Outputs number on top of Calculator Stack to currently selected channel via WRCH. (Converts from floating point to ASCII.)</td>
</tr>
</tbody>
</table>

Full explanation of the following Calculator Routines is beyond the scope of this document.

<p>| SUB     | 113 (71H)    | Subtract floating point format numbers (HL) minus (DE). (DE) assumed to be (HL) + 5. |</p>
<table>
<thead>
<tr>
<th>SERVICE</th>
<th>SERVICE CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>114 (72H)</td>
<td>Add (HL) + (DE). See SUB.</td>
</tr>
<tr>
<td>MULT</td>
<td>115 (73H)</td>
<td>Integer multiply HL * DE. Returns C if overflow.</td>
</tr>
<tr>
<td>TIMES</td>
<td>116 (74H)</td>
<td>Floating Point Multiply (HL) * (DE).</td>
</tr>
<tr>
<td>DIVIDE</td>
<td>117 (75H)</td>
<td>Floating Point Divide (HL)/(DE).</td>
</tr>
<tr>
<td>TRUNC</td>
<td>118 (76H)</td>
<td>Truncates a floating point number (HL) towards zero to an integer. Assumes (DE) = (HL) + 5.</td>
</tr>
<tr>
<td>FLOAT</td>
<td>119 (77H)</td>
<td>Converts number (HL) to floating point format. Assumes HL points to an integer in 5-byte format.</td>
</tr>
<tr>
<td>INTDIV</td>
<td>120 (78H)</td>
<td>Replaces top two numbers on Calculator Stack (X and Y) by X Mod Y and the integer quotient INT (X/Y). Returns with DE and HL = Calc.Stack Pointers.</td>
</tr>
<tr>
<td>INT</td>
<td>121 (79H)</td>
<td>Replaces the top of the Calculator Stack by its integer part. Returns with HL = top of Calc. Stack and DE = next free space.</td>
</tr>
<tr>
<td>EXP</td>
<td>122 (7AH)</td>
<td>Replaces the top of the Calculator Stack, X, by EXP(X). Returns with DE and HL = Calc.Stack Pointers.</td>
</tr>
<tr>
<td>LN</td>
<td>123 (78H)</td>
<td>Replaces the top of the Calculator Stack by its natural logarithm. Returns DE and HL = Calc.Stack Pointers.</td>
</tr>
<tr>
<td>ANGLE</td>
<td>124 (7CH)</td>
<td>Replaces the top of the Calculator Stack (X) by Y where Y is greater than or equal to -1 and less than or equal to +1 and the SIN X = SIN (PI/2 * Y).</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>COS</td>
<td>125 (7DH)</td>
<td>Replaces the top of the Calculator Stack by its COSINE.</td>
</tr>
<tr>
<td>SIN</td>
<td>126 (7EH)</td>
<td>Replaces the top of the Calculator Stack by its SINE.</td>
</tr>
<tr>
<td>TAN</td>
<td>127 (7FH)</td>
<td>Replaces the top of the Calculator Stack by its TANGENT.</td>
</tr>
<tr>
<td>ATN</td>
<td>128 (80H)</td>
<td>Replaces the top of the Calculator Stack by its inverse TANGENT.</td>
</tr>
<tr>
<td>ASN</td>
<td>129 (81H)</td>
<td>Replaces the top of the Calculator Stack by its inverse SINE.</td>
</tr>
<tr>
<td>ACS</td>
<td>130 (82H)</td>
<td>Replaces the top of the Calculator Stack by its inverse COSINE.</td>
</tr>
<tr>
<td>ROOT</td>
<td>131 (83H)</td>
<td>Replaces the top of the Calculator Stack by its Square Root.</td>
</tr>
<tr>
<td>TO_THE</td>
<td>132 (84H)</td>
<td>Replaces the top two numbers on the Calculator Stack (X, Y) by X**Y.</td>
</tr>
<tr>
<td>RDCH</td>
<td>133 (85H)</td>
<td>Wait for character from currently selected channel (calls INCH). Returns character code in A. See 4.1.1.</td>
</tr>
<tr>
<td>SENDCH</td>
<td>134 (86H)</td>
<td>Write character whose code is in A to currently selected output channel. See 4.1.2.</td>
</tr>
<tr>
<td>WRCH</td>
<td>135 (87H)</td>
<td>See 3.2.1.1, RESTART 16.</td>
</tr>
<tr>
<td>K_SCAN</td>
<td>136 (88H)</td>
<td>Keyboard Scan. See 4.1.1</td>
</tr>
<tr>
<td>SERVICE</td>
<td>SERVICE CODE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>P_LFT</td>
<td>137 (89H)</td>
<td>Backspace. Sets current column position back 1 for selected device. (System Variable updated is S_POSN, SPOSNL, or P_POSN for Screen, Lower Screen or Printer respectively.)</td>
</tr>
<tr>
<td>P_RT</td>
<td>138 (8AH)</td>
<td>Outputs a space to currently selected device.</td>
</tr>
<tr>
<td>P_NL</td>
<td>139 (8BH)</td>
<td>End-of-Line. Sets current position to start of next line if screen, or outputs printer buffer if printer.</td>
</tr>
<tr>
<td>PUTMES</td>
<td>140 (8CH)</td>
<td>Output message to currently selected device. DE points to base of message table which contains variable length ASCII coded messages. The first byte of the table and the last byte of each message must have the most significant bit set. Register A contains the message number, numbered from 0 upwards.</td>
</tr>
<tr>
<td>K_CLS</td>
<td>141 (8DH)</td>
<td>CLS command. Executes both CLS and CLLHS.</td>
</tr>
<tr>
<td>SCRL</td>
<td>142 (8EH)</td>
<td>Scrolls entire screen (primary display file) up 1 line.</td>
</tr>
<tr>
<td>F_PNT</td>
<td>143 (8FH)</td>
<td>POINT function. Processes X,Y parameters from Calculator Stack to BC. Returns unsigned integer value = 0 or 1 on Calculator Stack reflecting state of pixel at coordinates X/Y.</td>
</tr>
<tr>
<td>DRAWLN</td>
<td>144 (90H)</td>
<td>Same as DRAW L but enter with BC register containing coordinates, B=Y and C=X.</td>
</tr>
<tr>
<td>PUTLN</td>
<td>145 (91H)</td>
<td>Output Line Number as 4 digits, right aligned and space filled to currently selected output channel. HL points to MSB of 2-byte Line Number.</td>
</tr>
</tbody>
</table>

90
4.0 SYSTEM I/O GUIDE

4.1 I/O Channels

The TS 2068 software architecture supports up to 19 I/O Channels or "Streams", numbered from -3 through 15. Those numbered less than 0 are "hidden" or reserved for system use; Channels 0 through 15 are available for assignment via the OPEN # command which has the following format:

    OPEN # n,s

where n is the Channel number (0-15) and s is the Device Specification, e.g. "K" (keyboard), "S" (screen) or "P" (printer).

Channels 0 through 3 are initialized at power-on or execution of a NEW command to support the standard system devices and character I/O functions as shown in Figure 4.1-1. Channels 4-15 are considered "Closed". You can re-assign the standard I/O, e.g. OPEN # 2,"P" will direct all PRINT and LIST commands to the 2040 Printer instead of the screen. You can also assign Channels 4-15 and then direct I/O by including the Channel number (or a variable equated to the channel number) in the I/O statement, e.g. PRINT # n. Support for other than the standard system devices described above is not implemented in the original version of the TS 2068 and attempts to OPEN Channels or "Streams" using other than the standard device specifications ("K", "S" or "P") will result in an error message. One possibility for adding BASIC support for new devices is to intercept the I/O error on OPEN and other commands such as CAT and FORMAT via ON ERR and interpret the BASIC program line using your own machine code routines.

<table>
<thead>
<tr>
<th>Channel/Stream #</th>
<th>Device Specification</th>
<th>Command/Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>&quot;K&quot;</td>
<td>Keyboard/Lower Screen</td>
</tr>
<tr>
<td>-2</td>
<td>&quot;S&quot;</td>
<td>Main Screen</td>
</tr>
<tr>
<td>-1</td>
<td>&quot;R&quot;</td>
<td>RAM Write (not used)</td>
</tr>
<tr>
<td>0</td>
<td>&quot;K&quot;</td>
<td>Output to Lower Screen</td>
</tr>
<tr>
<td>1</td>
<td>&quot;K&quot;</td>
<td>INPUT command</td>
</tr>
<tr>
<td>2</td>
<td>&quot;S&quot;</td>
<td>PRINT/LIST commands</td>
</tr>
<tr>
<td>3</td>
<td>&quot;P&quot;</td>
<td>LPRINT/LLIST commands</td>
</tr>
</tbody>
</table>

FIGURE 4.1-1
The Channel architecture is implemented by a number of tables located in both ROM and RAM.

A. STRMS

STRMS is a 38 byte table (2 bytes for each of the 19 channels) located in the System Variables area beginning at 23568 (5C10H). It is initialized at power-on or NEW to the following values:

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5C10</td>
<td>0100 (Channel -3)</td>
</tr>
<tr>
<td>5C12</td>
<td>0600 (Channel -2)</td>
</tr>
<tr>
<td>5C14</td>
<td>0800 (Channel -1)</td>
</tr>
<tr>
<td>5C16</td>
<td>0100 (Channel 0)</td>
</tr>
<tr>
<td>5C18</td>
<td>0100 (Channel 1)</td>
</tr>
<tr>
<td>5C1A</td>
<td>0600 (Channel 2)</td>
</tr>
<tr>
<td>5C1C</td>
<td>1000 (Channel 3)</td>
</tr>
<tr>
<td>5C1E</td>
<td>0000 (Channel 4)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5C34</td>
<td>0000 (Channel 15)</td>
</tr>
</tbody>
</table>

This table is accessed using ((Ch.# * 2) + 16H) as an index added to 5C00H. The 2-byte value in the table is an index into the CHANS area of memory which contains the addresses of the I/O routines for the selected channel. If the 2-byte value is zero, the Channel is closed. The STRMS table is modified via the OPEN # and CLOSE # commands. When a Channel is OPENed, the device specification is used to obtain the 2-byte value to be inserted. This value is taken from the table STRMINIT in module EDIT of the Home ROM. When Channels 0 through 3 are CLOSEed, the values are restored to those used at power-on time. All others are cleared to zero.

B. CHANS

The CHANS System Variable at 23631 (5C4FH) contains the address of a 21-byte table initialized at power-on or execution of a NEW command to support "stream" I/O to the four standard system devices ("K", "S", "R" and "P"). Each table entry is 5 bytes long and is indexed by the value obtained from the STRMS table added to (CHANS)-1. Each entry has the following format:

- Output Routine Address 2 Bytes
- Input Routine Address 2 Bytes
- Device Specification 1 Byte

This table is copied from CHINIT in module EDIT of the Home ROM. The last byte of the table contains an 80H which will immediately precede the first line of the BASIC Program (PROG).

Whenever an I/O operation is performed, the appropriate Channel is "selected", i.e. its number is used as an index into STRMS to obtain the offset into the CHANS table. This offset is added to
(CHANS)-1 and the resultant pointer is loaded into the System Variable CURCHL for use by the next character I/O operation (WRCH/RDCH). The device specification from CHANS is used to find and execute the initialization routine in SELTAB.

C. SELTAB The Select Table is located in the EDIT module of the Home ROM and contains offsets to device dependent initialization routines for the standard devices "K", "S" and "P".

D. SPEC_T The Specification Table is located in the CHANS module of the Home ROM and contains offsets to device dependent OPEN routines for the standard devices "K", "S" and "P". It is accessed whenever an OPEN # is executed.

E. CL_TAB The Close Table is located in the CHANS module of the Home ROM and contains offsets to device dependent CLOSE routines for the standard system devices "K", "S" and "P". It is accessed whenever a CLOSE # is executed.

The following sections describe the standard system I/O devices supported via Channel I/O.

4.1.1 Keyboard

The low-level routines supporting keyboard input are executed every 1/60 of a second out of the Interruption Handler (Location 56 (38H)). The controlling routine is labelled UPD_K. This routine calls K_SCAN to determine if any key(s) are currently being depressed, controls the debouncing and repeat algorithms, calls K_BASE to determine the Base Code, calls CHCODE to translate the Base Code based on Mode (e.g. "K", "G" or "E" Mode), and finally, stores the resultant keystroke code in LAST_K and sets the flag KEYHIT. Figure 4.1.1-1 illustrates the mode control variable and associated flags and Figure 4.1.1-2 contains flowcharts of the keyboard support routines.

The character input routine associated with Device Spec. "K" is labeled IN_K. The entry address is obtained using the pointer in CURCHL when Channel 1 has been Selected and the Character I/O Input routines RDCH/INCH are executed. The IN_K routine tests the KEYHIT flag to detect the presence of input from the keyboard. When the KEYHIT flag=1, the contents of LAST_K are returned to the requestor.
FIGURE 4.1.1-1

TS 2068 MODE CONTROLS

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>23617(5C41H)</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = &quot;K&quot; or &quot;L&quot; Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = &quot;E&quot; Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 = &quot;G&quot; Mode</td>
</tr>
<tr>
<td>FLAGS</td>
<td>23611(5C3BH)</td>
<td>If MODE = 0 then:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 3 = 0 for &quot;K&quot; Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1 for &quot;L&quot; Mode</td>
</tr>
<tr>
<td>FLAGS2</td>
<td>23658(5C6AH)</td>
<td>If in &quot;L&quot; Mode then:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 3 = 0 CAPS Lock Off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>= 1 CAPS Lock On</td>
</tr>
</tbody>
</table>

FIGURE 4.1.1-2

FLOWCHARTS

TS 2068 KEYBOARD Routines
4.1.2 Video Screen

The TS 2068 system software supports I/O in the primary display file only. See Section 2.1.10 for the display file organization. The screen, which is 32 columns X 24 lines, is partitioned into two parts, the main or upper screen (22 lines) and the lower screen (2 lines). The lower portion of the screen is used for output of system messages and to echo input from the keyboard of BASIC commands, BASIC program lines, or data. The lower screen expands as needed for multi-line input, scrolling the entire screen upwards. The variable DF_SZ reflects the number of lines in the lower screen (default=2).

Character output to the screen is done using the Channel I/O described in Section 4.1 using device specification "K" for the lower screen and "S" for the upper screen. Each character is defined by an 8 X 8 group of pixels. The 8 bytes needed for each of the 133 characters supported by the TS 2068 are located as shown in Figure 4.1.2-1. Note that by constructing your own pixel data and placing (base address-100H) into CHARS, you can define your own character set.

Associated with each character position is an Attribute Byte controlling the background (PAPER) color, the foreground (INK) color, the intensity (BRIGHT), and whether the position is constant or alternates between true and inverse video (FLASH). Two other "attributes", OVER and INVERSE, are implemented by software at the time the character(s) are placed into the display file.

**FIGURE 4.1.2-1**

TS 2068 STANDARD CHARACTER TABLES

<table>
<thead>
<tr>
<th>Character Set</th>
<th>No.of Chars.</th>
<th>Char.Codes</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>96</td>
<td>32-127 (20-7FH)</td>
<td>Home ROM (3D00-3FFFH)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(Address-100H in CHARS)</td>
</tr>
<tr>
<td>Std.Graphics</td>
<td>16</td>
<td>128-143 (80-8FH)</td>
<td>Dynamically Generated by Software</td>
</tr>
<tr>
<td>User Defined Graphics</td>
<td>21</td>
<td>144-164 (90-A4H)</td>
<td>Home RAM (Address in UDG)</td>
</tr>
</tbody>
</table>
The screen output routine, SENDTV, is in Module IO 1 of the Home ROM. This routine is used for output to both the screen (upper and lower) and the dot matrix printer. The following sequence illustrates the major operations involved in executing a PRINT "A" statement:

1. Channel 2 is Selected (normal assignment assumed)
   - loads CURCHL with pointer into CHANS area for Channel 2 (first 2 bytes are address of Output Routine - SENDTV).
   - clears printer and lower screen flags
   - sets ATTR_T to values based on ATTR_P (current "permanent" attribute values are transferred to the system variable used by the screen output routine). If the PRINT statement contained temporary attribute controls, they would override the settings established via Select.

2. The character code for "A" (65/41H) is placed in Register A and a RESTART 16 (10H) is executed (WRCH). This jumps to SENDCH in module EDIT of the Home ROM which passes control to the SENDTV routine based on (CURCHL).

3. The registers are loaded from the System Variables with the current Row/Column position (S_POSN) and Display File address (DF_CC) for the main screen.

4. The character code is determined to be from the standard character set so the registers are loaded with the address from CHARS and the offset to the pixel pattern for "A" is calculated using the character code X 8 (shift left 3 places).

5. The first pixel row (8X1) from the character table is copied to the display file. The character table address is incremented by 1 and the display file address is incremented by 256 (100H). The next pixel row (8X1) is copied to the display file. This process is repeated until the 8 pixel rows have been copied. Masking of the data going into the display file is done based on the flags from P FLAG thus controlling the OVER and INVERSE attributes.

6. The attribute byte controlling the character position just written is updated based on the value in ATTR_T and other flags.
7. The variables S POSN and DF CC are updated to reflect the next screen position and return is made from the WRCH operation.

In the above sequence, if the print position for the "A" had started a new line following the 22 lines of the main screen, the SCROLL? prompt would have been outputted to the lower screen and, assuming a positive response, the upper screen would be scrolled up 1 line, a blank line inserted at the bottom of the upper screen, and the "A" printed at the start of the new line.

Graphics I/O using pixel coordinates is supported in the primary display file by the PLOT, DRAW and CIRCLE commands. The Home ROM module GRAPHS contains the major routines which implement these commands. They are limited to the 22 lines of the upper screen (256 X 176 pixels).

Figure 4.1.2-2 shows the internal representation used to designate row (line) and column positions. See Section 2.1.10 for details on the organization of the Display Pixel and Attribute Files. See Section 5.2 for details on software support necessary for the advanced video modes.

---

**FIGURE 4.1.2-2**

**DISPLAY FILE ROW/COLUMN NOTATION**

<table>
<thead>
<tr>
<th>BASIC Parameters</th>
<th>Internal Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line/Row 0</td>
<td>24 (18H)</td>
</tr>
<tr>
<td>1</td>
<td>23 (17H)</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>21</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
</tr>
</tbody>
</table>

| Column 0         | 33 (21H)                |
| 1                | 32 (20H)                |
| .                | .                       |
| 31               | 2                       |
4.1.3 2040 Dot Matrix Printer

Character output to the 2040 Printer is handled by the same routine used for the screen, SENDTV. When the Printer Flag=1, set by initialization for device "P", the pixel data is written into the Print Buffer instead of into the Display File. There is no Attribute Byte. The "attributes" OVER and INVERSE which are software controlled can be active. Since the Print Buffer is always precleared to zeros, OVER has no effect. INVERSE works exactly as it does for the screen, i.e. INK pixels are zero and PAPER pixels are 1.

The Print Buffer is located at 23296 (5800H) and is 256 (100H) bytes long, the data needed to print one line of 32 characters, each character comprised of 8 bytes (8 X 8 pixels/character). The buffer is cleared to zeros and the flag PRLEFT set to zero at power-on time (or execution of a NEW command). The PRLEFT flag is set to 1 whenever pixel data is written to the buffer. This flag is used when exit is made from a program to print any unprinted data prior to program termination. As the pixel data for a particular character is entered into the buffer, the buffer address is incremented by 32 (20H); the sequential data in the buffer therefore represents 8 complete scan lines of 32 characters. When the Print Buffer is full, or upon processing an End-of-Line (ODRE), or at program termination, the contents of the buffer are written to the Printer, the buffer is cleared and the PRLEFT Flag is set to zero.

Printer I/O is done via Port OFBH, but the Printer responds to any I/O Read/Write with Address Bit 7=1 and Address Bit 2=0. Therefore, any Port providing this combination, e.g. Ports OFA through OF8 and Ports OF3 through OF0 as well as others, will interface to the Printer. See Section 2.1.13.3 for the bit definitions for Printer I/O. The pixel data is written to the device by the routine PRSCAN in module IO_2 of the Home ROM which outputs 1 scan line (32 bytes) per bit at a time on each call to the routine.

There are two controlling routines for output to the printer. DUMPPR is called from SENDTV based on buffer full or End-of-Line control. This routine will call PRSCAN 8 times to output the 256 bytes of the Print Buffer (8 scan lines). The other routine is K DUMP which implements the COPY command. This routine calls PRSCAN 176 times to write the contents of the primary display file for the main screen to the printer (8 X 22). All of the low level print routines are in module IO_2 of the Home ROM.
4.2 Cassette Tape

Tape I/O is done via Port OFEH. An I/O read of Port OFEH pulls in the cassette input on Bit 6. An I/O write of Port OFEH Bit 3 controls the tape output with Bit 3 = 1 generating a high output and Bit 3 = 0 generating a low output.

Data is written to the tape under software control creating the following frequencies and format:

- Sync Pattern of 4032 cycles at 806.5 Hz. (5 sec.)
- Header: 17 bytes of data identifying the following data block as either Program, Number Array, Character Array, or Binary Code and containing other control information.

  The header is written as Data, i.e. the Most Significant Bit first in each byte, 1 cycle at 2040 Hz. for a Zero and 1 cycle at 1020 Hz. for a One. The first byte is zero identifying the header. The final byte is a Checksum calculated by XOR of all preceding data bytes.

- Software delay of approximately 835 milliseconds.
- Sync Pattern of 1612 cycles at 806.5 Hz. (2 secs.)
- Transition Pattern of 1 cycle at 2400 Hz.
- Data Block: Written as Data (see above) with first byte = -1 (FFH) and a final Checksum byte.

Figure 4.2-1 shows the header formats for the various types of data.

The routines used to actually write and read the tape (W_TAPE and R_TAPE) are in the TAPE Module of the Extension ROM (see map in Appendix A). They are accessible via the Extension ROM Interface Routine listed in Figure 3.2.2-2. The general flow required to write a header and data block is:

1. Call W_TAPE with A=0. IX contains the address of the header and DE contains the length.
2. Delay loop approximately 1 second.
3. Call W_TAPE with A=FFH. IX contains the address of the data block and DE contains the length.
The R TAPE routine performs either a LOAD (transfers data from tape to memory) or VERIFY (compare data from tape against data in memory) operation, based on the status at entry: Carry Set for Load and No Carry if Verify. As for the Write, A=Block Type (0 for Header and -1 (FFH) for Data Block). IX contains the memory address.

The tape routines return Carry=1 for successful completion and No Carry for error or Break Key detected. Both W TAPE and R TAPE exit via the routine W BORD which restores the Border color based on bits 3-5 of the system variable BORDCR. If the Break Key is detected during this exit routine, a RESTART 8 (ERROR) is executed.

NOTE: The write to Port OFEH in the exit routine restoring the Border Color has bit 3 = 0. This creates a final transition on the tape following a write operation. This transition is necessary in order to successfully read back the final data bit from some tape recording devices. If you are calling the W TAPE routine so as to bypass the normal exit path, you must perform this final write to Port OFEH with Bit 3 = 0 within a similar timeframe.

Addendum to R_TAPE routine: Register DE must contain the length of the block to be read (DE=17 for the Header, and DE=HDLEN for Data). See Fig. 4.2-1 for a definition of HDLEN.
### FIGURE 4.2-1

**TAPE HEADER FORMATS**

<table>
<thead>
<tr>
<th>HDTYPE</th>
<th>HDNAME</th>
<th>HDLEN</th>
<th>HDADD</th>
<th>HDVARS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(LSB/MSB)</td>
<td>(LSB/MSB)</td>
<td>(LSB/MSB)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>PROGRAM</td>
<td>Up to 10 ASCII Chars.</td>
<td>Starting Line No.</td>
<td>Length of Program = Offset or 8000H to Variables</td>
</tr>
<tr>
<td></td>
<td>E LINE (PROG)</td>
<td>E.G.: 0500=Line 5 or 0080H if no Line No.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NO.ARRAY</td>
<td>&quot;</td>
<td>Length LSB=00</td>
<td>N/A (=0)</td>
</tr>
<tr>
<td></td>
<td>Field from MSB=Array ID</td>
<td>Data 7.............0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Structure  TOO ... .... (ASCII - 60H)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CHAR.ARRAY</td>
<td>&quot;</td>
<td>Length LSB=00</td>
<td>N/A (=0)</td>
</tr>
<tr>
<td></td>
<td>Field from MSB=Array ID</td>
<td>Data 7.............0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Structure  T10 ... .... (ASCII - 60H)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CODE (BINARY)</td>
<td>&quot;</td>
<td>Length Address Specified in SAVE</td>
<td>N/A (=0)</td>
</tr>
<tr>
<td></td>
<td>Specified in SAVE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4.3 Joysticks

The two joysticks are controlled via Register 14 (I/O Port A) of the Programmable Sound Generator Chip (see Sections 2.1.6 and 2.1.7). Address and data are passed via Ports OF5H and OF6H respectively. The joysticks are read by first addressing Register 14 in the PSG by writing a 14 (OEH) to Port OF5H. The data is then read by executing an IN from Port OF6H, having the port address in Z80 Register C and the joystick (player) number in Register B (number = 1 or 2). Note that PSG Register 7, Bit 5 is assumed to be zero, enabling I/O Port A for input. If you ever use I/O Port A for output (R7,B6=1), you will want to clear Bit 6 prior to any input operation.
Sample routine:

```
GETJOY  LD  A,OEH
OUT  A,(OF5H)
LD  B,playerno
LD  C,OF6H
IN  A,(C)
CPL
AND 8FH
```

Load A = 14
Address the joystick port
Data Port address to C
Joystick data to A
Complement to High Active
Get significant bits

The data read is LOW ACTIVE, i.e. all bits = 1 (byte=FFH) when
the stick is at center and the button is not depressed. Figure
4.3-1 shows the interpretation of the data byte.

**FIGURE 4.3-1**

**JOYSTICK DATA**

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STICK UP</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STICK DOWN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>STICK LEFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>STICK RIGHT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT USED (Always '1')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**4.4 S/W Generated Sound (BEEP)**

The BEEP command produces sound using the speaker by toggling Bit
4 of I/O Port OFEH to generate a signal of a calculated frequency
and duration based on the command parameters. It uses the
routine PARP which takes as input two parameters, one defining
the period of the signal (HL) and the other defining the number
of cycles to be generated (DE) and outputs DE+1 cycles of a tone
having the period \(8N+236\) to \(8N+246\) T-States where \((HL) = N\). Both
the BEEP and PARP routines are in the K_SCAN module of the Home
ROM. The PARP routine is also used to generate the keyboard
"click" and the "raspberry" which can be varied by modifying the
values in the system variables PIP (23609/5C39H) and RASP (23608
5C38H).

**4.5 Sound Chip (SOUNDO)**

The SOUNDO command writes the first parameter (register number) to
Port OF5H (address to Programmable Sound Generator) and the
second parameter (load data) to Port OF6H (data to PSG). The
program line is scanned for multiple parameter pairs and
continues writing address/data pairs to the PSG until the end of
the statement is reached. See Section 2.1.6 for details on the
hardware of the PSG.
5.0 Advanced Concepts

5.1 Cartridge Software/Hardware

5.1.1 LROS

An LROS is identified by the following overhead bytes:

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Not Used</td>
</tr>
</tbody>
</table>
| 0001     | Cartridge Type  
|          | 01=LROS    |
| 0002/0003| Starting Address (LSB/MSB)  
|          | Address to be jumped to after  
|          | Operating System initialization is  
|          | complete. Order of bytes is as for a  
|          | JP instruction. |
| 0004     | Memory Chunk Specification.  
|          | Bits 0-7 represent Chunks 0-7  
|          | respectively in the Dock Bank in low  
|          | active format:  
|          | 0 if in use  
|          | 1 if not in use |

**NOTE:** When writing to the Horizontal Select Register (Port F4H), the Chunk Specification is High Active

The Memory Chunk Specification is used to enable the specified chunks in the Dock Bank prior to jumping to the address specified in Location 2 and 3. Control is transferred from the Initialization code in the Extension ROM via the GOTO BANK routine in Home Bank RAM Chunk 3, therefore Bit 3 of the Memory Chunk Specification must be set to 1 in order for the transfer to be accomplished as designed (Chunk 3 also contains the Machine Stack).

**CAUTION:** If Chunk 3 is marked for use in the Dock Bank, then when the Memory Chunk Spec. is written to Port F4H by the Bank Enable code, execution will continue from that point in Chunk 3 in the Dock Bank with the Stack Pointer addressing ROM.

An LROS is Z80 machine code and is in complete control of the TS 2068 hardware after transfer to the starting address has been made. It can directly implement an application, or it
can support multiple applications by implementing a language other than BASIC. An AROS dependent on such an LROS would have to be part of the same cartridge since there is only one cartridge connector.

 Interruption Mode 1 has been set by the TS 2068 and interruptions are enabled prior to passing control to the LROS starting address, therefore the LROS must contain appropriate code at location 56 (38H) to cover the case where the interruption occurs after Chunk 0 in the Dock Bank has been enabled, but before any action by the software cartridge to disable the interruption has been taken. Once control is transferred, the LROS may then disable the standard TS 2068 interruption by setting bit 6 of Port FFH, mask the interruption by executing a DI instruction, or set a different Interruption Mode. It may change the location of the Machine Stack. It may also change the memory selection by writing to Port OF4H with each bit set to 1 for the corresponding chunk to be enabled in the Dock Bank (high active format) or 0 to be enabled in the Home Bank. Thus, an LROS may contain code in Chunk 3, but it should be enabled after the OS RAM code has finished execution.

 Now that your LROS is in the driver's seat, you are on your own! Some important points to remember when mapping your Dock Bank memory and doing bank switching are:

1. The Display RAM is in Home Bank Chunk 2 for the primary display file and Chunk 3 for the second display file. This memory is accessed independently by the video hardware. The software only needs to enable it when actually reading or writing it.

2. The Dock Bank and Extension ROM Bank are mutually exclusive since they share the Horizontal Select Register in Port F4H. You will need a routine in the Home Bank RAM to do any switching between the two. You must also be careful to have the appropriate Home Bank Chunks enabled which are referenced by the Extension ROM code, e.g. the System Variables in Chunk 2 or possibly the bank switching code in Chunk 3.

3. Some interesting switching routines can be constructed by having parallel code in shadowing chunks of memory to take advantage of the "instant" switch in execution from one bank to another when the memory selection is made. E.g., a routine in the Dock Bank ROM in Chunk 6 could push a Home Bank address on the stack, write to Port F4H enabling Chunk 6 and any other desired chunks in the Home Bank (by deselecting them in the Dock), and have code at the next sequential instruction address in Home Bank RAM Chunk 6 to continue the path. A Return
instruction, for example, would pass control to the address on the stack. Code to switch memory back to the Dock Bank could be mapped in a similar way.

4. If you plan to use any of the System software routines, unless you know otherwise it is probably necessary to maintain the contents of Home Bank Chunks 2 and 3 intact (and Chunk 7 if the OS RAM routines have been relocated). The system routines rely heavily on the System Variables and assume that any pointers in them are pointing to the Home Bank. See Section 3.3.4.1 for details on using the RAM Interruption Handler and Section 6.0 for known corrections when using System S/W.

5. If you design an LROS implementing a higher-level language and want to support an AROS application, you must design your own initialization code to detect the presence of such an AROS. The TS 2068 will not look for the presence of an AROS if an LROS is present, therefore there will be no entry for the AROS in the System Configuration Table. Note that since there is only one cartridge connector, such an AROS would also have to be integrated with the supporting LROS in a single cartridge or cartridge board.
5.1.2 AROS

An AROS is identified by the following overhead bytes:

<table>
<thead>
<tr>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32768 (8000H)</td>
<td>Language Type</td>
</tr>
<tr>
<td></td>
<td>1 = BASIC [and machine code]</td>
</tr>
<tr>
<td></td>
<td>2 = Machine code only</td>
</tr>
<tr>
<td></td>
<td>(Any other value will result in Error 5, Missing LROS)</td>
</tr>
<tr>
<td>32769 (8001H)</td>
<td>Cartridge Type</td>
</tr>
<tr>
<td></td>
<td>2 = AROS</td>
</tr>
<tr>
<td>32770/32771 (8002/8003H)</td>
<td>Starting Address (LSB/MSB)</td>
</tr>
<tr>
<td></td>
<td>BASIC AROS = Addr. of First Program Line</td>
</tr>
<tr>
<td></td>
<td>Machine Code AROS = Addr. of First 780 Instruction</td>
</tr>
<tr>
<td>32772 (8004H)</td>
<td>Memory Chunk Specification</td>
</tr>
<tr>
<td></td>
<td>Bits 0-7 represent Chunks 0-7 respectively in the Dock Bank</td>
</tr>
<tr>
<td></td>
<td>in low active format as follows:</td>
</tr>
<tr>
<td></td>
<td>0 if in use</td>
</tr>
<tr>
<td></td>
<td>1 if not in use</td>
</tr>
<tr>
<td></td>
<td>NOTE: Bits 0-3 must be set to 1 for proper execution.</td>
</tr>
<tr>
<td>32773 (8005H)</td>
<td>Autostart Specification</td>
</tr>
<tr>
<td></td>
<td>0 = No Autostart</td>
</tr>
<tr>
<td></td>
<td>1 = Autostart</td>
</tr>
<tr>
<td>32774/32775 (8006/8007H)</td>
<td>Number of bytes of RAM to be</td>
</tr>
<tr>
<td></td>
<td>Reserved for Machine Code</td>
</tr>
<tr>
<td></td>
<td>Variables (LSB/MSB - 0100H=1 byte Reserved; 0002H=512 bytes Reserved.</td>
</tr>
</tbody>
</table>

5.1.2.1 BASIC AROS

A BASIC AROS is supported by special code in the System ROM (Section 3.2.1.2). The portion of the cartridge containing BASIC program lines is restricted to the upper half of the memory space beginning at location 32776 (8008H) in the Dock Bank. Support for User-Defined Functions, which requires searching for
the definition parameters within the program, is not implemented. Also, because the support code interfaces directly to the bank switching code in Home RAM Chunk 3 (does not allow for it to be relocated to Chunk 7), a BASIC AROS cannot utilize the advanced video modes and also execute BASIC program statements. If the cartridge contained machine code supporting advanced video modes, the TS 2068 would have to be returned to "Normal" video mode with the RAM mapped accordingly (see Figure 1.1-3) if control were to be returned to the BASIC Interpreter USR code.

Since execution of the cartridge BASIC program is done by copying program lines to a buffer in the Home Bank RAM (ARSBUF), the most efficient cartridge execution is obtained by making program lines as large as possible, i.e. making use of the multi-statement feature of the TS 2068. The reverse is true concerning execution of READ commands. An entire DATA statement is copied to the Home Bank RAM, but only the current item is accessed. It therefore will be more efficient to not make DATA statements excessively long. The BASIC program lines appear in the cartridge in exactly the same format used in the RAM, i.e. Line Number (2 bytes), Length (2 bytes), Command Token, etc. terminated by an Enter (ODH). Numerical constants appearing in a program line are followed by the CHR$ (OEH) byte and 5-byte floating point format described in the User Manual (see Appendix C of the TS 2068 User Manual). The Variables area is built in the RAM (address in VARS) exactly as though the program were in the RAM. All variables, including arrays, are built at the time of program execution - there is no provision for copying or accessing pre-defined variables from the cartridge, however, see Section 5.3.2. The last program line must be followed by a terminator byte having the Most Significant Bit set (e.g. 8OH), otherwise the Interpreter cannot detect the end of the program.

A BASIC AROS may contain machine code accessed via the USR function. If the machine code address is within the memory designated by the AROS Memory Select Specification as "in use", the Dock Bank will be enabled, otherwise the machine code address is assumed to be in the Home Bank. (See Section 6.0 for details on known problems in this area of the code.) Obviously, once control is transferred to the machine code in the AROS, the ball is now in your court. You could have additional machine code residing in the lower half of the Dock Bank memory space which you can now switch in. You only have to know what you're about. If and when you are ready to go back to
executing your BASIC program, you must enable Chunks 0-3 in the Home Bank and have the stack and other Home Bank RAM in the proper state for return to the USR function code in the BASIC Interpreter, i.e. what it was when the USR function passed control to you.

The Autostart feature begins execution out of the BASIC AROS immediately after system initialization. If the Autostart parameter is zero, control will go to the BASIC Interpreter as if there were no cartridge installed, although internal flags have been set noting that a BASIC AROS is present. The cartridge will be started when you execute a RUN or GOTO Line Number command.

The final parameter in the overhead bytes allows you to reserve RAM beginning in Chunk 3 at Location 26688 (6840H) for machine code and/or machine code variables. The designated number of bytes are reserved by the AROS support code prior to beginning program execution. The AROS buffer (ARSBUF) begins immediately following this reserved area (see Fig. 1.1-3). Note that this area is part of the RAM that gets relocated if the second display file is opened. Therefore access to your machine code and/or variables should be conditional on the video mode rather than direct if you are going to be using the advanced video modes. This reserved area begins at 31488 (7800H) when the second display file is open. Remember -- use of the second display file and execution of BASIC program from the cartridge are mutually exclusive.

The standard technique of reserving space for machine code by modifying RAMTOP could also be used to place machine code/variables at the top of the Home Bank RAM. If you place code above (RAMTOP) which is to be accessed via the BASIC USR function, the affected memory chunk(s) cannot be marked as "in use" in the cartridge in the AROS Memory Selection Specification.

5.1.2.2 Machine Code AROS

A machine code AROS is similar to an LROS with the exception that it is dependent on the System ROM for interruption handling if the interruption is enabled. This implies that Chunks 0-3 are enabled in the Home Bank.
The Autostart parameter should be set to 1 since if it is zero, control will be passed to the BASIC Interpreter as if the cartridge were not present. There is no BASIC command to directly start execution of a Machine Code AROS.

Because of a "bug" in the Initialization code handling a Machine Code AROS, the parameter specifying the number of bytes to be reserved for machine code variables must be adjusted by adding 21 (15H) to the actual number of bytes needed. This preserves the 21 byte CHANS area starting at 26688 (6840H). The reserved area then starts at 26709 (6855H) (or 31488 (7B15H) when the second display file is open). Access to the variables should be conditional based on the video mode rather than direct if you plan to use the advanced video modes. If you do not plan to utilize any of the system software, you can disregard the above and "do your own thing" with the RAM.

See Section 6.0 for known corrections when using System S/W.

5.1.3 EPROM Cartridge Board Application

Figure 5.1-1 provides the logic diagram for a pluggable EPROM cartridge board capable of configuring up to four 16K-byte (128K-bit) EPROM's of the 27128 type. The artwork for the PC board implementing that logic diagram is provided in Figures 5.1-2, 5.1-3 and 5.1-4 for the Component Side art, the Solder Side art, and the Solder Mask (one common mask for both sides), respectively.

See Section 2.4.2 for mechanical details of the connector portion of the PCB.
FIGURE 5.1-1
PLUGGABLE EPROM CARTRIDGE BOARD
LOGIC DIAGRAM
FIGURE 5.1-2
EPROM CARTRIDGE BOARD
COMPONENT SIDE ARTWORK

REDUCE TO 3.000 :.005

COMPONENT SIDE
SK2000-81

114
FIGURE 5.1-3
EPROM CARTRIDGE BOARD
SOLDER SIDE ARTWORK

TIMEX
SK2000-81

SOLDER SIDE
SK2000-81
FIGURE 5.1-4
EPROM CARTRIDGE BOARD
SOLDER MASK

SOLDER MASK
SK2000-81
5.2 Advanced Video Modes

The following sections describe the various video modes available on the TS 2068 and the major software support functions necessary. See Sections 3.2.2.3 and 3.2.2.4 for details on using the Video Mode Change Service. Appendix C contains descriptions and code listings for a number of software packages developed by Timex that support various screen modes and applications. Reference to these packages should aid in gaining an understanding of the software techniques needed to support the video mode hardware.

The TS 2068 video mode hardware works out of two areas of RAM, the primary display file at 4000H and the second display file at 6000H. Each area consists of 6912 (1800H) bytes used for pixel and/or attribute data based on the mode selected via bits 0-5 of Port FFH. The pixel data area divides into three blocks, each supporting 8 contiguous lines on the screen. See Section 2.1.10 for details on organization of the display RAM. Because the two display files occupy the same relative positions within their respective 8K Chunks, by setting/clearing Address Bit 13 a software routine can address the corresponding location in each file:

Address Bit 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

\[0\ 1\ 0\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\]

DF1

4000 - 5AFFH  (Bit 13 = 0)

Address Bit 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

\[0\ 1\ 1\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\ X\]

DF2

6000 - 7AFFH  (Bit 13 = 1)

In order to display a character on the screen, 8 bytes of pixel data must be entered into the display file, one for each scan row. For a particular character position, the scan rows are 100H bytes apart. E.g., the 8 bytes of pixel data for position Line 0/Column 0 are located at 4000H, 4100H, 4200H,......,4700H. Since this is the first character position on the screen, its Attribute byte, in Normal Mode, is the first byte in the Attribute File which starts at 5800H. The 768 (300H) Attribute Bytes are in sequential order starting at position 0/0 through 0/31,1/0 through 1/31, and so forth, ending with 23/0 through 23/31.

One method of determining the starting display file address for a particular line/column position is to build a table containing the starting address of each of the 24 lines (2 bytes per entry). Then construct an algorithm that takes the
line number and forms an index by multiplying it by 2 (shift left 1), add the index to the base address of the table, and read out the display file address. The column position is then simply an offset added to this address. By testing VDMMOD (23746 - 5CC2H) you can determine whether to set Bit 13 for the second display file, e.g. because you are in an odd column in 64-column mode, or simply because you are using the second display file in dual screen mode.

The following example illustrates this method. The table entries are in Hex:

<table>
<thead>
<tr>
<th>LINE #</th>
<th>INDEX</th>
<th>LSB/MSB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>00 40 4000H</td>
<td>Line 0 (Top of Screen)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>20 40</td>
<td>Line 1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>40 40</td>
<td>Line 2</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>(+20H)</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>(+20H)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>14(0EH)</td>
<td>E0 40</td>
<td>Line 7 (End of Upper Block)</td>
</tr>
<tr>
<td>8</td>
<td>15(10H)</td>
<td>00 48 4800H</td>
<td>Line 8 (Top of Middle Block)</td>
</tr>
<tr>
<td>9</td>
<td>13(12H)</td>
<td>20 48</td>
<td>Line 9</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>(+20H)</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>(+20H)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>30(1EH)</td>
<td>E0 48</td>
<td>Line 15(End of Middle Block)</td>
</tr>
<tr>
<td>16</td>
<td>32(20H)</td>
<td>00 50 5000H</td>
<td>Line 16(Top of Bottom Block)</td>
</tr>
<tr>
<td>17</td>
<td>34(22H)</td>
<td>20 50</td>
<td>Line 17</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>(+20H)</td>
<td></td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>(+20H)</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>45(2EH).....E0 50</td>
<td>Line 23(End of Bottom Block)</td>
<td></td>
</tr>
</tbody>
</table>

Line 17, Column 23 (11H/17H) would yield a display file address of 5020H + 17H = 5037H. If VDMMOD indicated the second display file was to be used, setting Bit 13 of the address would yield 7037H. If we were using 64-column mode, because the column is odd (Bit 0=1) we would set Bit 13 of the starting line address getting 7020H, then divide the column address by 2 (shift right 1) since there are only 32 columns in each display file. This would give us an offset of 11 (0BH) which added to the starting address results in a display file address of 7028H. Having the display file address, we now insert the 8 bytes of pixel data for the character desired, incrementing the display file address by 100H between each write (this is easily done by simply incrementing the upper register of the register pair containing the address). The following routine is a simplified version illustrating this process. It assumes that Reg. Pair DE contains the address of the desired character in the character table and that HL contains the address of the desired position in the display file.
Finally, we must update the Attribute Byte controlling the updated character position. The following sample algorithm will formulate the Attribute File address given the address of any of the scan rows of the character position. We will assume we have saved off the starting display file address and now have it in Register Pair HL.

```
GETATT
LD A,H MSB of DF Address
RRCA Shift right circular
RRCA to get Bits 3&4 (Block #)
RRCA to positions 0&1
AND 3 Clear other bits
OR 58H OR in Attr.File Base Adrs.
LD H,A Update MSB
```

NOTE: The LSB is the same as for the pixel data.

Using our first example, with a Display File address of 5037H, the Attribute File address would be 5A37H. The second example was using 64-Column Mode which does not require attribute file update (attributes determined by video mode setting).

See Section 5.2.2 for a sample algorithm to formulate the display file address for X,Y pixel coordinates. The above routine for calculating Attribute File address would be substituted for the method used in the example if not working in High Resolution Graphics mode.

In addition to data insertion, two major screen support functions are scrolling and clearing the screen. Scrolling is done in the System ROM by copying the entire display file data and attribute controls up one line position (Line 1 to Line 0, Line 2 to Line 1, etc.) and inserting a blank line at the bottom. Numerous more elaborate scrolling techniques can be implemented using various directions (up, down, left,
right) and smaller areas or "windows" of the screen. Similarly, clearing the screen, which consists of writing zeros to the data file and updating the attribute bytes to a uniform value, can be implemented on smaller sections of the screen. The software packages in Appendix C contain examples of such implementations.

5.2.1 Dual Screen Mode

In this mode the second display file is used to provide a second independent screen having the same data and attribute organization as the primary display file. By writing to Port FFH with Bits 0-5 = 1 (Bit 0 set), the second display file is activated at the video screen. Appendix C contains a software package supporting Dual Screen Mode. The software package uses the system variable VIMMOD to determine which display file is the target of the current operation. Special values for VIMMOD have been defined to permit building of one display file while the other is active at the screen so that a complete screen image is ready when the hardware mode is changed. Copy and Exchange routines have been provided to move data within and between the two display files. This enables the BASIC graphics commands like PLOT, CIRCLE and DRAW, which work only in the primary display file, to be used to create screens which are then moved into the second display file.

Because the System ROM works only in the primary display file, you can come up with some unusual situations when you have the second display file active at the screen and you are executing BASIC or using the System ROM routines. If an error occurs, for example, the error message will be placed into the primary display file and the ROM will be waiting for input from the keyboard to direct the next action, but all of this is invisible since you have the other display file active. The machine will appear to be "hung", but it is only doing its normal thing. Be prepared to enter a OUT 255,0 to an invisible command line in order to switch the display back to the standard file!!! Don't forget to also set VIMMOD (POKE 23746,128) to keep things consistent inside the dual screen support code.

5.2.2 High Resolution Graphics Mode

This mode is set by writing to Port OFFH with Bits 0-5=2 (Bit 1 set). In this mode, also called Extended Color Mode, the second display file is used to expand the number of Attribute bytes from one for each 8 X 8 pixel group to one for each 8 X 1 pixel group thus giving 32 X 192 positions within each of which two colors plus Bright and Flash can be defined. Each byte of pixel data entered into the primary display file has
its own Attribute byte in the corresponding location in the
second display file, e.g. the byte written to Location 4000H
has its Attribute byte at Location 6000H, the byte at 47FFH
(last byte of last scan row in Line 7) has its Attribute byte
at Location 67FFH, the byte at 57FFH (last byte of last scan
row in Line 23) has its Attribute byte at Location 77FFH.
The routine writing data to the screen would therefore enter
the pixel data to the desired location and then set Address
Bit 13 of the Primary Display File address and write the
desired attribute control byte to the resultant location. If
normal characters are being written to the screen in this
mode, eight Attribute bytes must also be written, one for
each of the bytes defining the character. The same technique
would be used for writing to both display files, i.e. for
each of the seven bytes entered after the first, the display
file address would be incremented by 256 (100H).

The System ROM graphics commands (PLOT, DRAW and CIRCLE)
place data into the Primary Display File and update the
Attribute File associated with the standard video mode
(5800H-5AFFH). In High Resolution Graphics Mode, the
hardware does not access this area for attribute control,
therefore its contents have no visible effect. If before or
immediately following execution of the BASIC graphics
operation, you update the attribute control information in
the second display file, you could possibly take advantage of
the System ROM graphics capability. Admittedly, this is not
a simple operation in the case of circles or drawing diagonal
lines and it will be more efficient to develop code
specifically to support this video mode.

The following sample routine takes as input two single byte
binary digits representing the X and Y coordinates of a pixel
position on the screen. It formulates the display file
address of the byte containing the pixel, creates a pattern
or mask byte for the specified bit position, sets the bit in
the display file, and updates the attribute byte (High
Resolution Graphics Mode assumed). This represents a
simplified version of the approach used in the System ROM
graphics support routines PLOTBC and SCRAMBL.

The two inputs are assumed to be as follows:

Reg. C = X Coordinate 0-255 (0-FFH) going left to right
across the screen.

Reg. B = Y Coordinate 0-191 (0-BFH) going from bottom to
top of the screen.

NOTE: This covers the full vertical range of 192 positions.
The Y Coordinate is checked for valid range and reversed directionally so that 0 represents the top of the screen and 191 represents the bottom. After this reversal, the two coordinates represent the following values:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCREEN</td>
<td>LIMIT</td>
<td>SCAN ROW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK</td>
<td>WITHIN</td>
<td>WITHIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0-2)</td>
<td>BLOCK</td>
<td>LINE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0-7)</td>
<td></td>
<td>(0-7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We first formulate the MSB of the display file address using the Block and Scan Line information in the Y Coordinate:

**PLOTXY**

- **PUSH (SAVECO),BC** - Save coordinates
- **LD A,191** - Test Y within range
- **SUB B**
- **JP C,ERROR** - Y coordinate beyond range
- **LD B,A** - Y Coordinate now 0=Top
- **AND 0COH** - Get Block No. (0-2)
- **RRA** - Shift Bits to Pos. 3&4
- **RRA**
- **LD H,A** - Save Block Bits
- **LD A,B** - Y Coordinate
- **AND 07** - Get Scan Row Bits
- **OR H** - Combine Block and Scan Row
- **OR 40H** - Base Address of DF (4000H)
- **LD H,A** - H = MSB of DF Address

Next we formulate the LSB of the display file address using the Line information from the Y Coordinate and the Column information from the X Coordinate:

- **LD A,C** - Get X Coordinate
- **RLCA** - Align to Pick Up Line
- **RLCA** - Bits from Y
- **RLCA** - A=2 LS Bits Column/XXX/3 MS
- **;** - Bits Column
- **AND 0C7H** - Clear Bits 3-5
- **LD L,A** - Save A in L
- **LD A,B** - Get Y Coordinate
- **AND 38H** - Get Line Bits
- **OR L** - Combine with Col.Bits
- **RLCA** - Shift to Final Position
- **RLCA** - A=Line #/Column
- **LD L,A** - L = LSB Display File Addr.
Next we get the pixel position within the byte by taking the last 3 bits of the X Coordinate and create a mask byte having all bits zero except the addressed pixel. This mask is then used to set the bit in the Display File. The address is set to Display File 2 to update the Attribute File (High Res. Graphics Mode is assumed to be active), and the routine is finished. The memory locations defined as ATTR and SAVECO are for illustration purposes only:

```
LD A,C                                   Get Pixel Position
AND 7                                    0=Leftmost (MSB); 7=
                                           Rightmost (LSB)
;                                           
LD B,A                                   Use as Control Count
INC B                                    B=1-8
LD A,0000001B                            Bit Mask
LOOP RRCA                                 Rotate Mask Bit
   DJNZ LOOP                              to Proper Position
   OR (HL)                                OR Bit into DF
   LD A,20H                               
   OR H                                   Set Bit 13 for DF2
   LD H,A                                 HL = Attribute File
   LD A,(ATTR)                            Get Attribute Byte
   LD (HL),A                              Update Attribute File
   POP BC                                 Original X/Y to BC Regs.
RET
```

Repetitive calls to this routine with the appropriate X/Y Coordinate values will "draw" on the screen. The System ROM routines for drawing lines and circles calculate the successive X/Y Coordinate values and use common low-level routines similar to the above to place each pixel in the display file.

5.2.3 54-Column Mode

In this mode, set by writing to Port OFFH with Bits 0-2=6 (Bits 1 and 2 set) and Bits 3-5 selecting ink color (0-7), the pixel data portions of the two display files are merged by the hardware on an alternating column basis to produce 64-columns across the screen. All even columns (0,2,4,...,62) are derived from the primary display file and all odd columns (1,3,5,...,63) are derived from the second display file. There are still 24 lines vertically from top to bottom. The attributes are controlled by bits 3-5 written to Port FFH selecting one of eight ink/paper combinations. The Bright and Flash attributes are fixed at 0 and the Border is fixed to match the paper color. The Attribute Files in RAM at 5800H-5AFFH (primary display file) and 7800H-7AFFH (second display file) are not utilized in this mode.
Software supporting this mode must set up the display file address for character insertion based on the column position (even=DF1; odd=DF2). When scrolling the screen (or a portion of it), any line of text on the screen requires the same operation to be done at the corresponding locations in each display file. This is also true to clear the screen (or a portion of it). To save a Screen on tape you must save two Code files, one for each display file. The SAVE filename SCREEN$ will work for the Primary Display File only. You will have to specifically SAVE the second display file via a SAVE filename CODE 24576,6144. Note also that because the Border color is fixed by the video mode, you will not see the usual "stripes" during a tape operation.

Code to support an 80-column mode screen was developed utilizing the 64-column hardware mode and redefining the character size to a 6 X 8 pixel group (there is really room for 84 characters if the full 256 pixel width is used). Since individual characters now can span the two display files (e.g. 2 pixels in DF1 and 4 in DF2) insertion of data into the display files involves masking the 6-bit character (or portion thereof) with the 8 bits of data read/written from/to the display file.

Appendix C contains descriptions and code listings of software packages supporting 64 and 80-Column modes.

5.2.4 Other

Appendix C also contains software packages supporting the following video screen features:

A. 40-Column Mode - utilizes the 6 X 8 character set defined for 80-Column Mode in "normal" mode. May be combined with the Dual Screen package.

8. Sprites - supports movement of software-defined objects and multi-directional screen scrolling services in the Primary Display File. You must create the actual bit map defining the shape of your sprite(s), but this package does the rest.
5.3 Other Advanced Concepts

5.3.1 Interruption Fielding

For a machine code program executing in the Home RAM, you can intercept the 17 ms. interruption for your own purposes by permanently enabling Chunk 0 in the Extension ROM Bank (write a 1 to Port 0F4H and always have Bit 7 of Port OFFH = 1) and inserting at Location 25262 (62AE Hex) a branch to your own interruption handler. (Or if VIDMOD is not zero, insert your branch instruction at Location 64110 (FA6EH).) By doing this you are forcing the interruption to branch to the RAM and then bypassing the OS RAM Interruption Handler - see Sections 3.2.2.1 and 3.3.3.1. Because the Video Mode Change Service automatically updates internal branch addresses in the OS RAM code when it is relocated between Chunk 3 and Chunk 7, you probably do not want to directly overlay the OS RAM Interruption Handler with your own code if you will be using the Video Mode service. Your branch instruction at 62AEH, however, will be copied unmodified to location FA6EH in Chunk 7 and vice versa.

Note that this technique cannot be used if you are using BASIC since then you must have Chunk 0 enabled in the Home Bank. It also cannot be used from a cartridge because the memory selection hardware (Port 0F4H) is common to the Dock and Extension ROM Banks and can only enable one of them at a given time as selected by Bit 7 of Port OFFH.

5.3.2 BASIC AROS Variables

In order to use pre-defined arrays and/or other BASIC variables, store them in the cartridge (possibly in the lower half of the addressable space which is not usable for BASIC program) and branch to a machine code routine via the USR function at the beginning of your BASIC AROS program. Use this routine to do the necessary memory selection and copy your data from the cartridge to the RAM (address in VARS). Adjust the System Variables E_LINE, WORKSP, STKBOT and STKEND to all point to the first free memory following your BASIC variables. Of course, all BASIC variables must conform to the format expected by the BASIC Interpreter. In addition to BASIC structures, you can also store screen images and machine code/variables in the cartridge for transfer to the RAM under your control. Consider using the XFER_BYTES service in the OS RAM.
6.0 Known "BUGS" and Corrections

This section describes the known problems in the TS 2068 System Software and gives corrections or work-arounds where these have been defined.

6.1 LROS and Autostart Machine Code AROS

6.1.1 If you will be using the System ROM Keyboard routines and accessing the input character code from system variable LAST_K (5C08H), you must initialize the TS 2068 to "L" mode by setting the system variable MODE at 23617(5C41H) to zero and setting Bit 3 of FLAGS (23611-5C3BH) to 1. (The TS 2068 is in "K" mode when control is passed from System Initialization to the Cartridge; Keyword Token codes will be placed in LAST_K instead of character codes.)

6.1.2 If you will be using the System ROM Calculator routines (RESTART 40 (28H) ) or any ROM routines that invoke them, you must initialize the System Variable MEM by doing the following:

```
LD HL,5C92H       Set HL=MEMBOT
LD (5C68H),HL    Initialize MEM
```

6.1.3 Chunk 3 must not be designated as "in use" by the Cartridge Memory Selection Specification byte. This will cause deselection of the bank switching code prior to completion of the transfer of control to the cartridge starting address. Once control has been transferred, the cartridge code may then enable Chunk 3 in the Dock Bank if desired. (See Section 5.1.)

6.1.4 No entry is made in the System Configuration Table for an AROS if an LROS is present. This means that an LROS designed to support either RAM based or cartridge based applications must include code for detection of an AROS.

6.2 Machine Code AROS

When setting the AROS Overhead parameter requesting RAM space for machine code variables, 21 + n bytes (15H + n) must be requested where n is the number of bytes needed. The machine language variables area then starts at 6855H immediately following the 21-byte CHANS area. (See Section 5.1.2.3.) NOTE: This does not apply to an AROS that contains both BASIC and machine code.
6.3 BASIC AROS

6.3.1 USR Function - When testing the USR address against the Cartridge Memory Selection byte to determine if the address is in the Home Bank or the Dock Bank, the wrong nibble is tested in the register thus a valid cartridge address could be erroneously processed as a Home Bank address. Since the ROM code cannot be corrected, the machine code in the cartridge would have to be moved to an address that does not cause a problem.

6.3.2 FOR/NEXT - If the limit of the FOR statement has already been passed on its initial execution, (e.g. FOR A=1 TO 10 and A has been set to 12), control is passed to the statement following the corresponding NEXT. In the AROS support code, the address of this statement is lost giving unpredictable results. Since the ROM code cannot be corrected, care must be taken not to use this technique in an AROS Cartridge. Normal usage of FOR/NEXT loops is not affected.

6.3.3 Advanced Video Modes - Because the BASIC AROS support code interfaces directly to the Bank Switching code in Chunk 3 (does not access based on its relocatability), the second display file cannot be open when executing BASIC program from an AROS.

6.4 Video Mode Change Service

6.4.1 Available Memory Test - When the size of memory needed is calculated by adding the size of the second display file (6912 bytes or 1800H) to the memory now in use (address in System Variable STKEND), the code fails to check for overflow. Thus if the address in STKEND is greater than 58623 (E4FFH), the fact that there is not enough free memory to open the second display file will not be detected and the system will "crash". If your BASIC program and/or variables area are large, you may want to make this test yourself prior to invoking the Video Mode Change Service in order to avoid this problem. The size of memory needed is subsequently tested against the contents of RAMTOP and if there is not sufficient space (value in RAMTOP is less than size needed), you will get Error 4, Out of Memory.
6.4.2 RAMTOP - When the machine stack and OS RAM code is moved to Chunk 7, the User Defined Graphics area is moved down in RAM by 2112 bytes (840H) to make room for the stack and OS RAM routines at the top of memory. The pointer in UDG is updated, however, the value in RAMTOP is not modified to insure that the relocated UDG area as well as the OS code and stack are protected from expansion of the BASIC program. You can avoid problems by setting RAMTOP via a CLEAR command specifying an address no greater than 63255 (F717H) prior to invoking the Video Mode Change Service. This reserves space between RAMTOP and the end of memory of 2280 bytes (8E8H) utilized as:

\[
\begin{align*}
168 \text{ bytes (A8H User Defined Graphics (21 X 8)} & \quad 2112 \text{ bytes (840H Machine Stack and OS Routines} \\
\hline
2280 & \quad (8E8H)
\end{align*}
\]

Example:

\[
\begin{align*}
\text{RAMTOP} = 63255 & \quad (F717H) \\
+ \text{Reserved Area} & \quad 2280 \quad (08E8H) \\
\hline
65535 & \quad (FFFFH)
\end{align*}
\]

The software packages in Appendix C are written assuming that RAMTOP is set to 57343 (DFFFH) or lower to protect the machine code which is loaded beginning at 57344 (E000H).

6.4.3 NEW Command - If you have used the Video Mode Change Service to open the second display file and now wish to execute the NEW command, you should first return the computer to "normal" mode by calling the video mode service with A=zero. This returns the User Defined Graphics and other RAM structures to their normal locations. If you don't do this, the UDG area will remain in the alternate location and, if you have not corrected RAMTOP as explained above, part or all of your UDG area could be cleared to zeros by the NEW command.

6.4.4 VIDMOD - When Mode 128 (80H) is designated for activating the Primary Display File in Dual Screen Mode the System Variable VIDMOD at 23746 (5CC2H) is set to zero instead of to 128. This creates a potential problem if the 17 ms. interruption occurs before VIDMOD can be corrected since the interruption fielder will branch to Chunk 3 instead of to Chunk 7 and Chunk 3 is now in use for the second display file. This problem is corrected by disabling the interruption prior to calling the Video Mode Change Service and setting VIDMOD to the correct value prior to re-enabling it. These corrections are included in the Extension ROM Interface Routine in Figure 3.2.2-2.
NOTE: On an initial access changing video mode from normal to Mode 128, the interruption is re-enabled within the Video Mode Change Service itself after copying the stack and other Chunk 3 data to Chunk 7. This cannot be corrected, but has not proven to present a problem in actual use. At the point where the interruption is first enabled, the Chunk 3 code is still intact allowing for correct processing of one interruption, and the path length from there to the point of correcting VIDMOD is apparently less than 17 ms. The interruption is also re-enabled within the Video Mode Change Service if you have applied the patches for the BANK ENABLE and RESTORE_STATUS routines (Section 6.5.4) which are executed in connection with inserting space into the RAM to open the second display file. Again, this has not proven to be a problem in actual use.

6.4.4 Interruption Inhibit - By setting Bit 6 of Port OFFH to a 1, the normal 17 ms. interruption generated from the SCLD to the Z80A CPU will be inhibited. When Port OFFH is written to by the Video Mode Change Service, Bit 6 is forced to zero. If you wish to inhibit the normal interruption via this mechanism, and also plan to use the Video Mode Change Service, it is recommended that you first invoke the service to remap the RAM and open the second display file, then set Bit 6 of Port OFFH to inhibit the normal interruption and write your own routine(s) for subsequent changing of the video mode setting that do not involve remapping the RAM. In this way you can maintain the value in Bit 6.

6.5 OS RAM Routines

In patching the OS RAM routines, care must be taken not to relocate CALL and JP instructions since this affects the modification of the code when it is moved between Chunks 3 and 7. All of the code containing actual addresses must be modified to reflect the relocation and this is done using a table in the Extension ROM. Since the table cannot be changed, none of these instructions can be moved. Also, any CALL or JP instructions added must be modified by you when the code is relocated.

6.5.1 Function Dispatcher - For a variety of reasons such as conflict with use of the IX Register, incorrect entries in the ROM Function Dispatcher Jump Table, etc. some Service Codes have been deleted from the Function Dispatcher table (Table 3.3.4-2). In addition, the following correction to the GET_STATUS routine is required in order to successfully utilize the Function Dispatcher from a cartridge.
6.5.2 GET_STATUS- Returns invalid memory selection status for the Home Bank, ROM Extension and Dock. This results in switching out of either the Home Bank or the Dock when status is "restored". This affects use of the Function Dispatcher and GET WORD routines, and any other code using GET_STATUS. Figure 6.5-1 shows the patches and additions necessary to correct this routine.

6.5.3 PUT WORD- Write data passed in Reg. Pair DE is overwritten prior to use. Figure 6.5-2 shows corrections.

6.5.4 BANK_ENABLE and RESTORE_STATUS-

If the 17 ms. interruption occurs during update of the memory selection hardware, it can cause the system to hang and RAM to be overwritten. This occurs when the interruption happens in an interval when Port FF Bit 7 is zero (thus selecting the Dock Bank) and Port F4 Bit 0 is one (thus enabling Chunk 0 in the Dock Bank) and there is no memory in Chunk 0 of the Dock Bank. This can be true when there is no cartridge installed, or if the cartridge installed is an AROS. This problem is corrected by disabling or masking the interruption while updating the memory selection hardware. Figure 6.5-3 shows one implementation of this correction.

6.5.5 SAVE STATUS and RESTORE STATUS - The value of Port FFH which includes video mode and interruption inhibit as well as Ext. ROM/Dock Select is saved and restored as a full 8-bits. Therefore any modification of this port by code accessed between execution of SAVE_STATUS and subsequent execution of RESTORE_STATUS (e.g. via CALL BANK or use of the Function Dispatcher) is "undone". This is one reason the Video Mode Change Service and some of the bank switching routines such as BANK_ENABLE cannot be meaningfully accessed via the Function Dispatcher.

6.5.6 CALL BANK- Does not correctly retrieve the stack entry designating the count of parameters being passed. Memory is overwritten in the case where this count is not zero. This is corrected by setting Location 6610H = 9 (POKE 26128,9). You only need to apply the correction once; it will be duplicated in Chunk 7 if the code is relocated.
**FIGURE 6.5-1**

**GET_STATUS CORRECTIONS**

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>OBJ_CODE</th>
<th>SOURCE STATEMENT</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(HEX)</td>
<td>(HEX)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input: Bank # in B

Output: Bank # in B (Bank Status if Exp.Bank) Memory Selection in C (Low Active Format)

| 6405 | F5 | GET_STATUS | PUSH AF | Save Regs. |
| 6406 | D5 |            | PUSH DE |           |
| 6407 | 78 |            | LD A,B  | Get Bank # |
| 6408 | FEFE | CP OFEH | Test if Ext.(254) |
| * 640A | 2824 | JR Z,GS_EXT |          |
| * 640C | 2837 | CP OFFH | Test if Home(255) |
| * 640E | A7 | JR Z,GS_HOME |           |
| * 6410 | 2827 | AND A | Test if Dock (0) |
| * 6411 |      | JR Z,GS_DOCK |          |
| 6413 |      |            |          |
|      |      |            | (Code for Expansion Banks not applicable) |
|      |      |            |          |
| * 6430 | OEFF | GS_EXT | LD C,OFFH | Assume none |
| * 6432 | DBFF |          | IN A,(OFFH) | Test if selected |
| * 6434 | E680 |          | AND 80H |           |
| * 6436 | 2812 | JR Z,GS_XT1 | Not active |
| * 6438 | 1808 | JR GETHS | Get Hor.Select |
| * 643A | OEFF | GS_DOCK | LD C,OFFH | Assume none |
| * 643C | DBFF |          | IN A,(OFFH) | Test if selected |
| * 643E | E680 |          | AND 80H |           |
| * 6440 | 2008 | JR NZ,GS_XT1 | Not active |
| * 6442 | DBF4 | GETHS | IN A,(OFFH) | Get Hor.Select Reg. |
| * 6444 | 2F | CPL | Invert to Low Active |
| * 6445 | 1802 | JR GS_XTO | Exit |
| * 6447 | DBF4 | GS_HOME | IN A,OF4H | All bits set are not active in Home Bank |
| 6449 | 4F | GS_XTO | LD C,A | Memory Select to C |
| 644A | D1 | GS_XT1 | POP DE | Restore Regs. |
| 644B | F1 | POP AF | Return |
| 644C | C9 | RET |            |

The asterisks mark the locations modified. See next page for list of corresponding POKE's for BASIC.
FIGURE 6.5-1

GET_STATUS CORRECTIONS

(continued)

From BASIC:

POKE 25610,40 (Location 640AH)
POKE 25611,36
POKE 25614,40 (Location 640EH)
POKE 25615,55
POKE 25617,40 (Location 6411H)
POKE 25618,39
POKE 25648,14 (Location 6430H)
POKE 25649,255
POKE 25650,219
POKE 25651,255
POKE 25652,230
POKE 25653,128
POKE 25654,40
POKE 25655,18
POKE 25656,24
POKE 25657,8
POKE 25658,14
POKE 25659,255
POKE 25660,219
POKE 25661,255
POKE 25662,230
POKE 25663,128
POKE 25664,32
POKE 25665,8
POKE 25666,219
POKE 25667,244
POKE 25668,47
POKE 25669,24
POKE 25670,2
POKE 25671,219
POKE 25672,244
POKE 25673,79
FIGURE 6.5-2
PUT_WORD CORRECTIONS

<table>
<thead>
<tr>
<th>LOCATION (HEX)</th>
<th>OBJ.CODE (HEX)</th>
<th>SOURCE STATEMENT</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>6338</td>
<td>F5</td>
<td>PUT_WORD</td>
<td></td>
</tr>
<tr>
<td>633C</td>
<td>C5</td>
<td>PUSH AF</td>
<td>Save Regs.</td>
</tr>
<tr>
<td>633D</td>
<td>CD5E64</td>
<td>CALL GET_NUMBER</td>
<td>Bank # of Owner</td>
</tr>
<tr>
<td>6340</td>
<td>D5</td>
<td>PUSH BC</td>
<td></td>
</tr>
<tr>
<td>6341</td>
<td>50</td>
<td>LD D,B</td>
<td>Save Target Bank #</td>
</tr>
<tr>
<td>6342</td>
<td>47</td>
<td>LD B,A</td>
<td>Bank # of Owner</td>
</tr>
<tr>
<td>6343</td>
<td>CD0564</td>
<td>CALL GET_STATUS</td>
<td>Get Bank Status</td>
</tr>
<tr>
<td>6346</td>
<td>C5</td>
<td>PUSH BC</td>
<td>Save It</td>
</tr>
<tr>
<td>6347</td>
<td>CD4D64</td>
<td>CALL GET_CHUNK</td>
<td>Get Bit Map</td>
</tr>
<tr>
<td>634A</td>
<td>2F</td>
<td>CPL</td>
<td>Set High Active</td>
</tr>
<tr>
<td>634B</td>
<td>42</td>
<td>LD B,D</td>
<td>Target Bank # to B</td>
</tr>
<tr>
<td>634C</td>
<td>4F</td>
<td>LD C,A</td>
<td>Memory Select Byte</td>
</tr>
<tr>
<td>634D</td>
<td>CD9964</td>
<td>CALL BANK_ENABLE</td>
<td>Enbl. Target Mem.</td>
</tr>
<tr>
<td>6350</td>
<td>C1</td>
<td>POP BC</td>
<td>Saved Bank Status</td>
</tr>
<tr>
<td>6351</td>
<td>D1</td>
<td>POP DE</td>
<td>Saved Data</td>
</tr>
<tr>
<td>6352</td>
<td>73</td>
<td>LD (HL),E</td>
<td>Write LSB</td>
</tr>
<tr>
<td>6353</td>
<td>23</td>
<td>INC HL</td>
<td>Increment Adrs.</td>
</tr>
<tr>
<td>6354</td>
<td>72</td>
<td>LD (HL),D</td>
<td>Write MSB</td>
</tr>
<tr>
<td>6355</td>
<td>28</td>
<td>DEC HL</td>
<td>Restore HL</td>
</tr>
<tr>
<td>6356</td>
<td>CD9964</td>
<td>CALL BANK_ENABLE</td>
<td>Restore Bank St.</td>
</tr>
<tr>
<td>6359</td>
<td>C1</td>
<td>POP BC</td>
<td>Restore Regs.</td>
</tr>
<tr>
<td>635A</td>
<td>F1</td>
<td>POP AF</td>
<td></td>
</tr>
<tr>
<td>635B</td>
<td>C9</td>
<td>RET</td>
<td>Return</td>
</tr>
</tbody>
</table>

Input: Data in DE, Address in HL, Bank # in B

The asterisks mark the locations modified.

From BASIC:

POKE 25408,213
POKE 25424,193
POKE 25425,209
POKE 25426,115
POKE 25427,35
POKE 25428,114
POKE 25429,43

NOTE: The corrections to GET_STATUS and BANK_ENABLE are also required.
### BANK_ENABLE AND RESTORE_STATUS CORRECTIONS

<table>
<thead>
<tr>
<th>BANK_ENABLE:</th>
<th>Location</th>
<th>Object Code</th>
<th>From BASIC POKE Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6499H</td>
<td>00</td>
<td>NOP</td>
<td>25753</td>
<td>0</td>
</tr>
<tr>
<td>649D</td>
<td>F3</td>
<td>DI</td>
<td>25757</td>
<td>243</td>
</tr>
<tr>
<td>651CH</td>
<td>FB</td>
<td>EI</td>
<td>25884</td>
<td>251</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESTORE_STATUS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>654AH</td>
</tr>
<tr>
<td>6570H</td>
</tr>
</tbody>
</table>

In both cases, the Disable Interrupt and Enable Interrupt are being done by deleting the preservation of the AF Registers (PUSH AF/POP AF). If your code requires AF to be saved, you must do it prior to calling either of these routines or any other system routines that use them. Note also that if you already have the interruption masked when these routines are entered, it will be enabled when they are exited. If this proves to be a problem, replace the Enable Interruption (EI) instruction with a NOP and do the enable at a more appropriate place in your own code.

6.5.6 GET_NUMBER- Always returns the Dock Bank # for any memory enabled in the ROM Extension. Unlikely to be a problem because of limited use of the ROM Extension.

6.5.7 XFR_BYTES- Improperly passes memory select byte for the case where source and destination are in the same bank. This is corrected by setting Location 676AH = 5FH (POKE 26474,95).

6.6 GENERAL

6.6.1 Pressing ENTER multiple times with an invalid tape command on the edit line (syntax error) causes the system to reset. This is due to overflowing the Bank Status Stack in RAM Chunk 3/7 due to the multiple calls to and from the Extension ROM via the Call Bank code without normal termination (the error causes a RESTART 8 to be executed out of Home ROM code called from the ROM Extension). It shouldn't take anybody that many tries to get a tape command right, so this is not a real problem, but you may want to keep it in mind. For any call made through the OS RAM services, you should have a corresponding return to keep the structures clean.
6.6.2 ON ERR GOTO - If a non-existent line number is specified, followed by an error, the system will hang. The ROM code is in an endless loop trying to report the absence of a valid error handler to the non-existent error handler!!! On some errors, you will get an unexpected 0 OK termination showing the line number of your Error Handler. This is because some ROM routines temporarily clear the INTPT Flag (Bit 7 of FLAGS). This flag is set to 0 when checking syntax and set to 1 when executing; if an error is detected while the Flag=0, the error handler code is branched to but is not executed.

6.6.3 Parameters to the SOUND command are not fully validated, therefore you can specify a number beyond the valid range for a given operation and not get an error, for example, you can write a value greater than 63 to the Enable Register (Reg.7), possibly changing the I/O Port used for reading the joysticks from input to output. If you specify a number larger than 255 (FFH), only the least significant byte will be actually written to the Programmable Sound Generator. Access to PSG Reg. 14 (IO-A) used for the Joysticks is also not precluded via the SOUND command.

If you experience difficulty in reading the joystick(s), do a write to PSG Reg. 7 clearing Bit 6 to 0 to guarantee that the joystick path is enabled for input (see Section 4.3). This write can be done by executing a SOUND 7,63 (or any value less than 63).

The INTEGER function for (-65536) gives an incorrect result of -1, and for other cases where the result should be -65536, it gives -1E-38. Since the ROM code cannot be changed, there is no correction.

6.6.4 If you respond to the SCROLL? message using multiple keys such as Cap Shift/2 or Cap Shift/Symbol Shift, you will get strange results like dumping of the Edit Line with the "C" or "E" cursor, display of ROM data, or multiple scrolls. Stick to single key responses and you won't have any problems!

6.6.5 When DELETE (Cap Shift/O) is held down to do deletion of characters in the Edit Line, sometimes it outputs the DELETE Keyword instead (it should not do this in auto-repeat mode). This is especially noticeable when the input line is long. Since the ROM code cannot be corrected, you must try releasing and pressing the DELETE key at differing frequencies and you will be able to get past this "Bug".
## APPENDIX A

### HOME ROM MAP

| LINK 1.7 | DATA 1E82 | SYNTAX |
| LOAD MAP | DEF 201D | SYNTAX |
| MODULE ORIGIN LENGTH | DEFREC 1750 | LIST |
| BLOCK 0000 0000 | DELSYM 087E | IO_2 |
| BASIC 0000 0227 | DEL.DE 1740 | LIST |
| KSCAN 0227 02D9 | DEL.K 08FD | IO_2 |
| IO_1 0500 0502 | DESLUG 000D | IO_2 |
| IO_2 0A02 031B | DEL_HL 1668 | LIST |
| EDIT 061D 0682 | DIGIT? 30D9 | INOUT |
| CHANS 139F 0142 | DIM 2FC0 | IDENT |
| LIST 1AE1 02D4 | DIVIDE 266E | SUMS |
| AROS 1785 0190 | DRAW 26DB | GRAPHS |
| SYNTAX 1945 050A | DRAWLN 2313 | GRAPHS |
| SYNTWO 214F 04B4 | DRAWL 2910 | GRAPHS |
| GRAPHS 2603 0251 | DUMPRL 0A23 | IO_2 |
| EXPRN 2854 04C0 | DYADIC 18DC | SYNTAX |
| IDENT 2C70 03E9 | ECHO 0C83 | IO_2 |
| INOUT 3059 0301 | EDIT_K 0492 | IO_2 |
| SUMS 335A 032A | END? 1844 | SYNTAX |
| CALC 3694 0437 | ENDSTT 1AB9 | SYNTAX |
| FUNCTS 3ABB 01CE | ENDTEM 184A | SYNTAX |
| TAPEMSG 3CB9 0053 | ERASE 2504 | SYNTWO |
| CH_SET 3D00 0300 | ERRZ 1891 | SYNTAX |
| ERR 07E1 0101 | ERPA 1FCF | SYNTAX |
| ERR5 07C1 0101 | ERR6 056C | SUMS |
| ERRB 1F29 | SYNTAX |
| ERRH 237E | SYNTWO |
| ERRO 1230 | EDIT |
| ACS 3C5E | FUNCTS |
| ADD 33D3 | SUMS |
| ALNUM? 3046 | IDENT |
| ALPHA? 3046 | IDENT |
| ANGLE 389F | FUNCTS |
| AROS 18C6 | AROS |
| ARRAY 37C5 | CALC |
| AR_LN 17EA | AROS |
| AR_NXT 17FF | AROS |
| ASN 345E | FUNCTS |
| ATN 58FD | FUNCTS |
| ATTMAS 0710 | IO_1 |
| BEEP 043A | KSCAN |
| BORDER 243E | SYNTWO |
| BREAK? 2009 | SYNTAX |
| CAT 25C5 | SYNTWO |
| CHCODE 0371 | KSCAN |
| CHINIT 11AA | EDIT |
| CHK_SZ 1FBB | SYNTAX |
| CIRCLE 2679 | GRAPHS |
| CLCHAN 13BE | CHANS |
| CLEAR 1F36 | SYNTAX |
| CLEL 13F3 | EDIT |
| CLOG 08A9 | IO_1 |
| CLOSE 139F | CHANS |
| CLPR 0355 | IO_2 |
| CLR_BC 1F39 | SYNTAX |
| CLS 08EA | IO_1 |
| CLS_B 097F | IO_1 |
| COMIT 23A6 | SYNTWO |
| COLOUR 23DE | SYNTWO |
| CONI 1EE4 | SYNTAX |
| COS 3BC5 | FUNCTS |
| CP_BC 16E8 | LIST |
| CTRO 371A | CALC |

| 136 | | |

### GLOBAL ADDRESS MODULE

- ACS 3C5E: FUNCTS
- ADD 33D3: SUMS
- ALNUM? 3046: IDENT
- ALPHA? 3046: IDENT
- ANGLE 389F: FUNCTS
- AROS 18C6: AROS
- ARRAY 37C5: CALC
- AR_LN 17EA: AROS
- AR_NXT 17FF: AROS
- ASN 345E: FUNCTS
- ATN 58FD: FUNCTS
- ATTMAS 0710: IO_1
- BEEP 043A: KSCAN
- BORDER 243E: SYNTWO
- BREAK? 2009: SYNTAX
- CAT 25C5: SYNTWO
- CHCODE 0371: KSCAN
- CHINIT 11AA: EDIT
- CHK_SZ 1FBB: SYNTAX
- CIRCLE 2679: GRAPHS
- CLCHAN 13BE: CHANS
- CLEAR 1F36: SYNTAX
- CLEL 13F3: EDIT
- CLOG 08A9: IO_1
- CLOSE 139F: CHANS
- CLPR 0355: IO_2
- CLR_BC 1F39: SYNTAX
- CLS 08EA: IO_1
- CLS_B 097F: IO_1
- COMIT 23A6: SYNTWO
- COLOUR 23DE: SYNTWO
- CONI 1EE4: SYNTAX
- COS 3BC5: FUNCTS
- CP_BC 16E8: LIST
- CTRO 371A: CALC
EXTENSION ROM MAP

**LINK 1.7**

<table>
<thead>
<tr>
<th>LOAD MAP</th>
<th>ORIGIN</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>XBASIC</td>
<td>0000</td>
<td>0068</td>
</tr>
<tr>
<td>TAPE</td>
<td>0068</td>
<td>007F</td>
</tr>
<tr>
<td>INIT</td>
<td>08E7</td>
<td>04C9</td>
</tr>
<tr>
<td>CHNO_VID</td>
<td>0DB0</td>
<td>0193</td>
</tr>
<tr>
<td>PASSING</td>
<td>0F43</td>
<td>0047</td>
</tr>
<tr>
<td>BS</td>
<td>0F8A</td>
<td>001E</td>
</tr>
</tbody>
</table>

**GLOBAL | ADDRESS | MODULE**

| AKEY      | 08AA   | TAPE   |
| BLDSCET   | 09F4   | INIT   |
| CALL_B    | 0F99   | BS     |
| CHNO_V    | 0E8E   | CHNO_VID |
| CLDFIL    | 0E27   | CHNO_VID |
| EXINIT    | 08E7   | INIT   |
| GOTO_B    | 0F8A   | BS     |
| LOAD      | 05CC   | TAPE   |
| MERGE     | 06E5   | TAPE   |
| ODFIL     | 0DB0   | CHNO_VID |
| PASSIN    | 0F43   | PASSING |
| RD_BIT    | 0189   | TAPE   |
| RESSCT    | 0C4C   | INIT   |
| R_EDGE    | 018D   | TAPE   |
| R_TAPE    | 00FC   | TAPE   |
| SAVE      | 0851   | TAPE   |
| SLVM      | 01AB   | TAPE   |
| W_BORD    | 00E5   | TAPE   |
| W_TAPE    | 0068   | TAPE   |

**PROGRAM XBASIC --- 1000 BYTES**

**ENTRY: 0000**

**DISPATCH**

0624  THIS MODULE IS COPIED TO RAM 6200 (space reserved 6200-683Fh).

<table>
<thead>
<tr>
<th>GLOBAL</th>
<th>ADDRESS</th>
<th>MODULE</th>
<th>ORIGIN</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BANK_E</td>
<td>6499</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BS_MAX</td>
<td>6315</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BS_SP</td>
<td>65CE</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALL_B</td>
<td>6500</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATE</td>
<td>66E8</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPAT</td>
<td>6200</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET_CH</td>
<td>6440</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET_NU</td>
<td>645E</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET_ST</td>
<td>6405</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GET_WO</td>
<td>6316</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTO_B</td>
<td>6572</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTO_E</td>
<td>6815</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT</td>
<td>62AE</td>
<td>DISPATCH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| TABLES: | FIXTBL | 1000 | 007C |
|         | JNPTBL | 1024 | 0124 |
| UNUSED: |        | 1624 | 06DC |

<table>
<thead>
<tr>
<th>ORIGIN</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>107C</td>
<td>0160</td>
</tr>
</tbody>
</table>

138
LOC   ORG  CODE M * STM % SOURCE STATEMENT

ACM 5.8

420  *LIST ON
421  *LIST ON
422  INCLUDE NMSYSVAR.C
423  ! HERE ARE SOME NEW SYSTEM VARIABLE DEFINITIONS
424  !
425  !
426  STKPS EQU 200H
427  SHROPT EQU 0F0H 1500H
428  SHATPT EQU 0F6H 1500H
429  HS EQU 40H
430  HS_LSN EQU HS
431  INN EQU 20H
432  HT_LSN EQU INN
433  ARN EQU 40H
434  ARNcplusplus EQU 17H
435  STALL EQU ARN
436  CMD EQU UCOM
437  STALL EQU CMD
438  LOWYNB EQU O00W
439  FREE_BYTES EQU 32
440  PRML_OUT EQU 0
441  HCR_SEL EQU 10
442  BMB EQU 11
443  UPDLK EQU 02EH
444  
445  
446  GLOBAL DISPATCH INT HNL PUT _ WORD GET _ WORD
447  GLOBAL WRITE _ BL _ RED _ FD _ RD _ GT _ STATUS _ GT _ NUMBER
448  GLOBAL _ Gmówi _ CHUNK _ BANK _ ENABLE _ GOUT _ BANK _ CALL _ BANK
449  GLOBAL_Create _ BITMAP _ MOV _ BYTES
450  GLOBAL _ XFER _ BYTES _ BMS _ BS _ SP
451  GLOBAL _ CREATE _ BITMAP _ MOVE _ BYTES
452  
453  ! DISPATCH (SRC _ CODE _ PASSED _ ON _ STACK)
454  !
455  ! SRC _ CODE _ IS _ A 16 _ BIT _ QUANTITY. _ BIT _ IS _ USED _ AS _ A _ JUMP _ FLAG. _ IF
456  ! SET, THE DISPATCHER WILL DO A GOTO _ BANK _ TO _ THE _ SPECIFIED _ ROUTINE.
457  ! OTHERWISE IT WILL DO A CALL _ BANK.
458  !
459  !
460  !
461  jmptrbl EQU 1FFFH
462  LAST _ EXT _ SVC EQU 13
463  LAST _ RAM _ SVC EQU 24
464  !
465  !
466  !

6200  DEC 10000
6209  DISP  ICH  0
6204  DE 30H  470  ADD  Jx  SP  17  =  SP
6206  C5  471  PXTH  A  1500H
6207  F5  472  FSTH  AF  1500H
620C  C5  473  FSTCH  BC
6209  03  474  PUSH  DE
620A  ES  475  PUSH  HL
620B  DD 00D2  476  LD  E  (1H  + 2)
620E  DD 4103  477  LD  D  (1H  + 2)  1DE  =  SVC _ CODE
6211  AF  478  XOR  A
6212  CB23  479  SRA  E
6214  CB12  480  RL  D  1DE  =  2DE
6216  17  481  RLA  1A  =  JUMP _ FLAG
6217  210000  482  LD  HL  LAST _ EXT _ SVC
621A  CB25  483  SLA  L
621C  CB11  484  RL  H  1HL  =  2HL
621E  A7  485  AND  A
621F  ED52  486  SBC  HL  DE  1COMPARE _ HL _ AND _ DE
6221  3015  487  JR  MC  D _ EXT  1IF  DE  =  00H
6222  211800  488  LD  HL  LAST _ RAM _ SVC
622C  CB25  489  SLA  L
6228  CB14  490  RL  H
622A  A7  491  AND  A
622B  ED52  492  SBC  HL  DE
622D  280F  493  JR  C  D _ HOME
622F  0AFF  494  LD  B  225  1HERE _ FOR _ RAM _ BASED _ SERVICES
6231  CD964  495  CALL  DET _ STATUS  1GET _ STATUS _ OF _ HOME _ BANK
6234  0AFF  496  LD  B  225  1BC  =  HOME _ BANK / HORIZ _ SELECT
6236  1804  497  JR  D _ SAVE
6238  0AFE  498  D _ EXIT  1BD  =  225  1HERE _ FOR _ EXT. _ RAM _ BASED _ SERVICES
623A  0EFE  499  LD  C  OFDM
623C  1004  500  JR  D _ SAVE
623E  0AFF  501  D _ HOME  1LD  B  225  1SET _ BANK _ ENABLE _ PARMS _ FOR _ HOME
6240  0E00  502  LD  C  0
6242  P5  503  D _ SAVE  PUSH  AF  1SAVE _ JUMP _ FLAG _ AND _ BANK _ ENABLE _ PARMS
6243  C5  504  PUSH  BC  1CALL _ ADDR _ OF _ TABLE _ ENTRY
6244  21FF  505  LD  HL  jmptrbl
6247  37  506  SCF
6248  ED52  507  SBC  HL  DE
624A  0AFF  508  LD  B  2154
624C  CD1463  509  CALL  DET _ WORD
624F  E6  510  EX  DE  HL
6250  C1  511  POP  BC
6251  F1  512  POP  AF  1RESTORE _ JUMP _ FLAG, _ ETC.
6252  A7  513  AND  A
6253  281F  514  JR  Z  D _ CALL
6255  D7F1E  515  LD  (1H  - 2)  C  1PUT _ BANK _ 0 _ AND _ HCR _ SEL _ ON _ STACK
6258  D7F1E  516  LD  (1H  - 1) B  1SAVE _ RET _ ADDR
625B  D6800  517  LD  L  (1H  + 4)
625E  D6441  518  LD  H  (1H  + 3)
6261  D7403  519  LD  (1H  + 3) H  1PUT _ RET _ ADDR _ BACK _ ON _ STACK

139
694E CB5F 720 SRP A
590D CB5F 721 SRP A
3922 CB5F 722 SRP A
4947 77 723 LD (HL), A
595D 0C07 724 LD A, 7
6925 DF5F 752 OUT (SADPT), A
6939 78 726 LD A, B
5974 D5F5 727 OUT (SADPT), A
695C 0E 728 LD A, OEH
695E DF5F 729 OUT (SADPT), A
6959 70 730 LD A, C
6941 DF45 731 OUT (SADPT), A
6A33 F1 732 POP AF
6A4A 77 733 LD (HL), A
6A45 F1 734 POP AF
5945 320000 735 LD (LOWBY), A
595E EI 736 POP HL
5A34 C1 737 POP BC
5A3F F1 738 POP AF
4A C9 739 RET
740 1
741 1
742 READ BS, REG: LIN,ActionTypes 1, at 0hex 120000: TRY INIT E1
743 1
744 1

6A40 FF 745 READ BS, REG: PUSH AF
4A 746 PUSH BC
4A C9 747 PUSH HL
4A 748 LD H, D
4A 749 LD L, 0
4A 74A 800000 750 LD A, (LOWBY).
4A 74B FF 751 PUSH AF
4A 74C 7E 752 LD A, (HL)
4A 74D F5 753 PUSH AF
4A 74E 7E 754 LD A, 7
4A 74F DF5F 755 OUT (SADPT), A
5A3D DB5F 756 N A, (SADPT)
5A3F 47 757 LD B, A
5A4C DEOE 758 LD A, OEH
5A58 D9F5 759 OUT (SADPT), A
5A54 DB5F 760 IN A, (SADPT)
5A5C AF 761 LD C, A
5A5D 75 762 PUSH BC
5A5E 8E07 763 LD A, 7
5A5F D9F5 764 OUT (SADPT), A
5A60 8E40 765 LD A, A0H
5A6E DB5F 766 OUT (SADPT), A
5A7A DEOE 767 LD A, OEH
5A7C DB5F 768 OUT (SADPT), A
5A7E 47 769 LD A, A
5A80 DB5F 76A OUT (SADPT), A
5A82 75 76B LD C, A
5A83 8000 76C LD D, E
5A84 C1 76D POP BC
5A85 FF 76E LD A, 7
5A86 DB5F 770 OUT (SADPT), A
5A87 7E 771 LD A, B
5A88 DB5F 772 OUT (SADPT), A
5A89 7E 773 LD A, 7
5A8A DB5F 774 OUT (SADPT), A
5A8B 7E 775 LD A, 7
5A8C DF5F 776 OUT (SADPT), A
5A8D 75 777 LD C, A
5A8E 80 778 LD D, E
5A8F C1 779 POP BC
5A90 FF 77A LD A, 7
5A91 DB5F 77B OUT (SADPT), A
5A92 7E 77C LD A, B
5A93 DB5F 77D OUT (SADPT), A
5A94 7E 77E LD A, 7
5A95 DF5F 77F OUT (SADPT), A
5A96 75 780 LD C, A
5A97 DB5F 781 OUT (SADPT), A
5A98 75 782 LD C, A
5A99 F1 783 POP AF
5A9A 77 784 LD (HL), A
5A9B 7F 785 POP AF
5A9C 320000 786 LD (LOWBY), A
5A9E EI 787 POP HL
5A9F C1 788 POP BC
5A9F F1 789 POP AF
4A C9 78A RET
800 1
801 1
802 1
803 1
804 1
805 1

6A45 F5 806 GET_STATUS: PUSH AF
6A44 DB 807 PUSH BC
6A41 7E 808 LD A, B
6A4D FEFE 810 CF OFFH
6A4F 00CE 810 JR Z, GET_LEFT
6A4F FEFF 811 CF OFFH
6A5E 2610 812 JR Z, GET_HOME
6A5E 0A 813 ADD A
6A5F 261F 814 JR Z, GET_LEFT
6A5F 1690 815 LD B, BNA
6A5F 56 816 LD E, B
6A61 CD83 817 CALL WRITE_BS, REG
6A63 164C 818 LD D, MS, MSN
6A61 1660 819 LD E, MS, MSN
142
143
649E 3A563 920 LD A, (BX) (MAX_BANK) TO GET LARGEST BANK NUMBER
649F 43 924 AND A
64A0 221 925 JR Z, BE_SKIP IF NO EXP. BANKS
64A1 1680 928 LD D, BNA
64A2 1E00 928 LD E, C
64A3 5C633 929 CALL WRITE_BS_REG
64A4 1680 929 LD D, HSP
64A5 FF 930 PUSH AF
64A6 79 930 LD A, C
64A8 FF 930 CPL
64A9 8F 932 LD E, A
64AA FF 932 POP AF
64AB 5C633 933 CALL WRITE_BS_REG TURN OFF APPROPRIATE BITS OF ALL EXP. BANKS
64AC 78 934 BE_SKIP LD A, B
64AD 87 935 AND A
64AE 211 936 JR NZ, BE_NTDOK
64AF 9D 936 LD A, C
64B0 FF 938 CP OFDH
64B1 2606 939 JR I, BE.EXIT.OK HERE FOR DOCK
64B2 FF 93A IN A, (I.REPT) (HERE FOR DOCK)
64B3 63 940 RES 7, A
64B4 FF 942 OUT (I.REPT), A
64B5 7F 943 BE(EXIT.OK) LD A, C
64B6 7F 944 CPL
64B7 FF 945 OUT (DIHSP), A ENABLE DOCK
64B8 FF 946 JR BE.EXIT
64B9 7D 947 BE.NTDOK LD A, B CHECK IF EIT.
64BA FF 948 CP OFDH
64BB 2010 949 JR NZ, BE.NTEXT HERE FOR EIT.
64BC FF 950 IN A, (I.REPT) DISABLE DOCK
64BD 17 951 MLA C
64BE 00 952 RPM
64BF 1F 954 PRA
64C0 FF 955 OUT (I.REPT), A
64C1 63 956 IN A, (I.REPT)
64C2 3003 957 JR NZ, BE.SET
64C3 05 958 IN A, (DIHSP)
64C4 FF 959 CP OFDH
64C5 FF 960 OUT (DIHSP), A
64C6 FF 961 JR BE.EXIT
64C7 8F 962 BE.SET IN A, (DIHSP)
64C8 CD 963 SET O, A
64C9 FF 964 OUT (DIHSP), A
64CA 1E2D 965 JR BE.EXIT
64CB FF 966 BE.NTEXT IN A, (DIHSP) DISABLE DOCK
64CC 2F 967 CPL
64CD 5F 968 LD E, A
64CE 7F 969 LD A, C
64CF 7F 970 CPL
64D0 FF 971 OR E
64D1 7F 972 CPL
64D2 FF 973 OUT (DIHSP), A
64D3 63 974 IN A, (DIHSP)
64D4 FF 975 CP OFDH
64D5 FF 976 OUT (DIHSP), A
64D6 FF 977 IN A, (DIHSP)
64D7 79 978 RES 7, A
64D8 FF 979 OUT (I.REPT), A
64D9 FF 980 IN A, (DIHSP)
64DA 7F 981 RES 0, A
64DB 10F4 982 OUT (DIHSP), A
64DC 78 983 BE.CH.HOME LD A, B CHECK IF HOME
64DD FF 984 CP OFDH
64DE 260E 984 JR I, BE.EXIT IS HOME, GO DONE
64DF 1E00 986 LD D, BNA WRITE NEW EXP. BANK STATUS
64E0 5C 986 LD E, B
64E1 CD363 987 CALL WRITE_BS_REG
64E2 1E40 988 LD E, E
64E3 79 989 LD A, C
64E4 2F 990 CPL
64E5 FF 991 OR E
64E6 CD633 992 CALL WRITE_BS_REG
64E7 FF 993 BE.EXIT POP HL RESTORE REGISTERS
64E8 D1 994 POP DE
64E9 C1 995 POP BC
64EA FF 996 POP AF
64EB FF 997 RET
64EC 8F 998 SAVE_BK_STATUSES (STATUS_ADDRH: IX)
64ED 9D21 1000 PUSH HL PUSH STATUS
64EE 9F 1000 SAVE_STATUS PUSH AF SAVE REGS
64EF E5 1000 SAVE_STATUS PUSH AF
64F0 E5 1000 SAVE_STATUS PUSH BD
64F1 E5 1000 SAVE_STATUS PUSH DE
64F2 521 1000 DBNF SAVE STATUS: SAVE BANK STATUS
64F3 1E00 1000 IN A, (I.REPT) LEAVE BITS 0-6 ALONE; NOF PUT IN
64F4 CD363 1000 CALL WRITE_BS_REG
64F5 1E40 1000 LD E, E
64F6 1D1 1001 INC X
64F7 1F4 1001 IN A, (DIHSP) GET DOCK BANK STATUS
64F8 CD7700 1001 LD (IX), A
64F9 D022 1022 INC X
64FA 1E40 1001 IN A, (DIHSP) GET DOCK BANK STATUS
64FB CD7700 1001 LD (IX), A
64FC CD7700 1001 LD (IX), A
64FD CD7700 1001 LD (IX), A
64FE CD7700 1001 LD (IX), A
64FF CD7700 1001 LD (IX), A
6500 1001 AND A
6501 1001 AND A
6502 1001 AND A
6503 1001 AND A
6504 1001 AND A
6505 1001 AND A
6506 1001 AND A
6507 1001 AND A
6508 1020 95, LOOP LD E, B 15 BANK NUMBER INTO E
6509 CD05A4 1022 CALL GET_STATUS GET STATUS OF BANK 88
6732 4F 1322 6733 06 00 1323 6735 21 00 00 1324 6728 2F 1325 6729 A7 1326 673A ED 42 1327 673C 2B 1328 672D 2B 1329 673E E5 1330 673F DE 1 1331 6741 00 00 1332 6743 C1E 65 1333 6745 09 1334 6747 D1 1335 6749 B3 1336 674F C066 1337 6752 F9 1338 6753 015 1339 6756 DD 61 05 1340 6759 C066 1341 675C 4F 1342 675D F1 1344 675F D07E 9 1346 6762 D050 1347 6765 44 1348 6766 2005 1349 6768 7B 1350 6769 91 1351 676A 47 1352 676B 1008 1353 676E 7B 1354 6770 5F 1355 6772 1276 1356 6773 2B 1357 6774 6B 1358 6775 CD96 1360 6778 1245 1361 6779 30 1362 677B E3 1363 677C E9 1364 677E 12B 1365 6780 1355 1366 6781 21 00 9C 1367 6783 0 1368 6784 1F 1369 6786 C1E 1370 6787 0 1371 6789 C0 1372 678B 3004 1373 678D 2B 1374 678E 19 1375 678F 1376 6790 1377 6791 E2 1378 6793 18 1379 6794 1346 1380 6795 1346 1381 6797 1378 1382 6798 0 1383 6799 18 1384 679A 1346 1385 679C 1346 1386 679D 0 1387 679E 1346 1388 679F 1346 1389 67A0 0 1390 67A1 1346 1391 67A2 0 1392 67A3 1346 1393 67A4 0 1394 67A5 1346 1395 67A6 1346 1396 67A7 0 1397 67A8 1346 1398 67A9 0 1399 67AA 1346 139A 67AB 0 139B 67AC 1346 139C 67AD 0 139D 67AE 1346 139E 67AF 0 139F 67B0 1346 13A0 67B1 0 13A1 67B2 1346 13A2 67B3 0 13A3 67B4 1346 13A4 67B5 0 13A5 67B6 1346 13A6 67B7 0 13A7 67B8 1346 13A8 67B9 0 13A9 67BA 1346 13AA 67BB 0 13AB 67BC 1346 13AC 67BD 0 13AD 67BE 1346 13AE 67BF 0 13AF 67C0 1346 13B0 67C1 0 13B1 67C2 1346 13B2 67C3 0 13B3 67C4 1346 13B4 67C5 0 13B5 67C6 1346 13B6 67C7 0 13B7 67C8 1346 13B8 67C9 0 13B9 67CA 1346 13BA 67CB 0 13BB 67CC 1346 13BC 67CD 0 13BD 67CE 1346 13BE 67CF 0 13BF 67D0 1346 13C0 67D1 0 13C1 67D2 1346 13C2 67D3 0 13C3 67D4 1346 13C4 67D5 0 13C5 67D6 1346 13C6 67D7 0 13C7 67D8 1346 13C8 67D9 0 13C9 67DA 1346 13CA 67DB 0 13CB 67DC 1346 13CC 67DD 0 13CD 67DE 1346 13CE 67DF 0 13CF 67E0 1346 13D0 67E1 0 13D1 67E2 1346 13D2 67E3 0 13D3 67E4 1346 13D4 67E5 0 13D5 67E6 1346 13D6 67E7 0 13D7 67E8 1346 13D8 67E9 0 13D9 67EA 1346 13DA 67EB 0 13DB 67EC 1346 13DC 67ED 0 13DD 67EE 1346 13DE 67EF 0 13DF 67F0 0 13E0 67F1 1346 13E1 67F2 0 13E2 67F3 1346 13E3 67F4 0 13E4 67F5 1346 13E5 67F6 0 13E6 67F7 1346 13E7 67F8 0 13E8 67F9 1346 13E9 67FA 0 13E9 67FB 1346 13EA 67FC 0 13EB 67FD 1346 13EC 67FE 0 13ED 67FF 0 13EE

148
```assembly
6714 D523 1423 INC IX
671A A5FF 1424 LD SP, IX
671C E1 1425 POP HL
671F 81 1426 POP DE
6725 C1 1427 POP BC
672B F1 1428 POP AF
6733 D813 1429 POP IX
6902 DE32 1430 EX (SP), IX
6904 DE33 1431 POP IX
6906 DE34 1432 EX (SP), IX
6908 DE11 1433 POP IX
690A DE53 1434 EX (SP), IX
690C DE35 1435 POP IX
6910 DE3F 1436 EX (SP), IX
6912 DE37 1437 POP IX
6914 D9 1438 EX (SP), IX
1440 1
1441 1
1442 1
1443 1
1444 1
6915 DE15 1445 GOTO EXT TOF IX
6917 F5 1446 PUSH AF
6918 DEFF 1447 IN A, (HL)
691A C9FF 1448 SET 7, A
691C D9F3 1449 OUT (HL), A
691E 3E01 1450 LD A, 1
6920 DEFA 1451 OUT (HL), A
6922 F1 1452 POP AF
6923 EE 1453 JP (HL)
1454 END

FIXTAB
LOC OBJ CODE M STMT SOURCE STATEMENT
1 DISPATCH EOU 6200H
2 INT EOU 6204H
3 GET_WORD EOU 6206H
4 PUT_WORD EOU 6208H
5 GET_STATUS EOU 6405H
6 GET_NUMBER EOU 6406H
7 BANK_ENABLE EOU 6407H
8 SAVE_STATUS EOU 651EH
9 RESTORE_STATUS EOU 651FH
10 B5_STACK EOU 6540H
11 B5_SP EOU 6541H
12 GOTO_BNK EOU 6572H
13 CALL_BNK EOU 6590H
14 MOVE_BYTES EOU 6608H
15 CREATE_BITMAP EOU 6609H
16 XFER_BYTES EOU 6722H
17 18 HERE IS THE FIXUP TABLE FOR THE VIDEO MODE CHANGER. IT DEFINES THE
19 LOCATIONS IN RAM WHICH MUST BE UPDATED WHEN MOVED FROM CHUNK 3 TO CHUNK 7
20 OR V CE-VERS. THE ADDRESSES IN THE TABLE ARE DEFINED AS CHNK: 3 ADDRESSES
21
22 23 ORG 1DO0H
24
25 1DO0 3262 26 FIXTAB
1D02 4862 27
1D04 7262 28
1D06 4862 29
1D08 8662 30
1D0A 2462 31
1D0C 6362 32
1D0E 6C62 33
1D10 FB62 34
1D12 1A62 35
1D14 2662 36
1D16 2462 37
1D18 2662 38
1D1A 2662 39
1D1C 2662 40
1D1E 4662 41
1D20 4862 42
1D22 4062 43
1D24 5762 44
1D26 1764 45
1D28 1564 46
1D2A 2964 47
1D2C 6164 48
1D2E 6564 49
1D30 6064 50

149
APPENDIX B

System Variables Definition File

2068 HOME ROM

TS2000 HOME ROM OBJ CODE M STMT SOURCE STATEMENT

13 #EJECT
14 #INCL SYSVAR
15 #PAGESIZE 54

16
17 RST: MACRO #ROUT
18 RST #ROUT
19 ENDM

20
21 ASSERT: MACRO #COND
22 COND .NOT.(#COND)
23 ERROR IN ASSERTION #COND
24 ENDC
25 ENDM

26
27 1 SYSTEM VARIABLES
28
29 L_LEN  EQU 32 1 # CHARS PER LINE ON THE DISPLAY
30 TV_LNS: EQU 24 1 NO. OF LINES ON TV SCREEN
31 D_FILE: EQU 4000H 1 ADDRESS OF DISPLAY FILE
32 ATTRS: EQU D_FILE+L_LEN*TV_LNS*8 1 SCREEN ATTRIBUTES
33 PRBUFF: EQU ATTRS+TV_LNS*#LEN 1 PRINTER BUFFER

34 ASSERT PRBUFF,AND,OFFH=0
COND .NOT.(PRBUFF,AND,OFFH=0)
ERROR IN ASSERTION PRBUFF,AND,OFFH=0
ENDC

150
35 KSTATE EQU PDBUFF+L_LEN+8  ; SEE KB DOCUMENTATION
36 KS.AI EQU 0  ; 1ST BYTE IS A CHAR KEY PRESSED
37 KS.CI EQU 1  ; 2ND IS TIME TILL COUNTS AS RELEASED
38 KS.BI EQU 2  ; 3RD IS TIME (IN FRAMES) TILL REPEAT
39 KS.DI EQU 3  ; 4TH IS CODE WHEN ENTERED
40          ; 5TH - 6TH ARE A SECOND SET OF 1ST FOUR
41 LAST_KI EQU KSTATE+8  ; NEWLY PRESSED KEY
42 REPDEL EQU LAST_KI+1  ; DELAY BEFORE 1ST REPEAT (INITIALIZED TO 35)
43 REPHER EQU REPDEL+1  ; DELAY BEFORE SUBSEQUENT REPEATS (INITIALIZED TO 5)
44 DEFADD EQU REPHER+1  ; -> CHAN AFTER `'` IN FORMAL PARAMETER LIST; MUST BE
45          ; 1ST WHEN NO USER-DEFINED IF BEING EVALUATED
46 K_DATA EQU DEFADD+2  ; DATA BYTE IN COMPOSITE CHAR FROM KEYBOARD
47 TV_DATA EQU K_DATA+1  ; USED FOR STORING BYTES IN COMPOSITE CHARACTERS!
48          ; (TV_DATA) = KEY BYTE...
49          ; (TV_DATA+1) = 1ST DATA BYTE FOR AT OR TAB.
50 STRMS EQU TV_DATA+2  ; STREAM DATA POINTERS (OFFSETS FROM (CHANS)-1) TO
51          ; CHANNELS,-1 = STREAM NOT OPEN.
52 HI_DRE EQU 3  ; NO. STREAMS HIDDEN FROM USER. THESE ARE TIED
53          ; UNALTERABLY TO SPECIFIC CHANNELS.
54 HID_K EQU -3  ; KEYBOARD
55 HID_S EQU -2  ; TV, UPPER HALF OF SCREEN
56 HID_R EQU -1  ; INSERTION IN RAM
57 COM_STA EQU 0  ; STREAM FOR COMMANDS
58 INF_STA EQU 1  ; STREAM FOR INPUT DATA
59 STPT EQU 2  ; STREAM FOR PRINT
60 LPR_STA EQU 3  ; STREAM FOR LPRINT
61 CHARS EQU STRMS+(HI_DRE+16)*2+1 -> 0+20H BYTES BEFORE CHARACTER SET
62 FART EQU CHARS+2  ; NO. CYCLES OF ERROR NOISE (2 BYTES BELOW MIDDLE C)
63 PIP EQU FART+1  ; NO. CYCLES OF KEYBOARD NOISE (3 BYTES ABOVE MIDDLE C)
64 Y EQU PIP+1  ; VALUE ALWAYS HELD IN Y
65 ERR_NR EQU Y  ; IRUN TIME ERROR #1 - 1
66 FLAGS EQU ERR_NR+1  ; VARIOUS FLAGS
67 SPC EQU 0  ; SUPPRESS SPACE BEFORE TOKENS
68 PRI EQU 1  ; PRINTING TO PRINTER, NOT TV
69 LMODE EQU 2  ; L MODE, NOT K, AT CURRENT CHARACTER
70 LMODE EQU 3  ; L MODE, NOT K, AT CURSOR
71 KEY_SOFT EQU 5  ; KEY HIT FOUND
72 NO EQU 6  ; EXPRESSION IS NUMERICAL... NOT STRING
73 INTPT EQU 7  ; REG INTERPRET RATHER THAN CHECK SYNTAX
74 TVFLAG EQU FLAGS+1  ; FLAGS ASSOCIATED WITH THE TV
75 LHS EQU 0  ; PRINTING TO LOWER HALF OF SCREEN
76 EDIT EQU 1  ; OUTPUTTING LINE FOR EDIT OR NO. FOR STRING
77 ECHEQ EQU 2  ; ECHO REQUESTED IF INPUTTING FROM KEYBOARD
78 LIST EQU 3  ; OUTPUTTING AN AUTOMATIC LISTING
79 CLHS EQU 4  ; CLEAR LOWER HALF WHEN KEY PRESSED
80 ERR_SP EQU TVFLAG+1  ; -> BOTTOM ITEM ON MACHINE STACK.
81 LISTSP EQU ERR_SP+2  ; RETURN ADDRESS FROM AUTOMATIC LISTING
82 MODE EQU LISTSP+2  ; O = K OR L; 1 = F; 2 = 0.
83 NEWPPC EQU MODE+1  ; LINE TO BE JUMPED TO
84 NSPPC EQU NEWPPC+2  ; SUBLINE TO BE JUMPED TO (BIT 7 OFF FORCES JUMP)
85 PPC EQU NSPPC+1  ; LINE # OF INSTR BEING-interPRETED
86 SUBPPC EQU PPC+2  ; NO. WITHIN LINE OF INSTR BEING INTERPRETED
87 BORDCR EQU SUBPPC+1  ; BORDER COLOUR (SHIFTED LEFT BACKG BITS WITH OS IN
88 0  ; BITS 0-2 & 6-7)
89 E_PWM EQU BORDCR+1  ; LINE # OF "CURRENT" LINE IN LISTING
90 VARS EQU E_PWM+2  ; THE VARIABLES FROM (VARS) UP TO & INCLUDING (STKEND)
91          ; ARE "MOVABLE" IN THE SENSE THAT THEY ARE ADJUSTED
92          ; (BY REMSGZ IN MODULE EDIT) WHENEVER STUFF IS
93          ; INSERTED IN OR DELETED FROM RAM.
94 DEST EQU VARS+2  ; -> 1ST RECORD FOR A VARIABLE (LAST IS 1 BYTE 80H)
95 CHANS EQU DEST+2  ; CHANNEL DATA (INCLUDING FLOPPY BUFFERS)
96          ; EACH ITEM COMPRIS;
97          ; THE ADDRESS OF AN OUTPUT ROUTINE FOR WRCH
98          ; THE ADDRESS OF AN INPUT ROUTINE FOR INCH
99          ; A 1-BYTE CODE FOR THE DEVICE TYPE
100          ; &, WHERE APPROPRIATE, A FILE NAME, ADDITIONAL
101          ; DATA & A BUFFER
102 CURCHL EQU CHANS+2  ; -> DATA FOR CURRENT CHANNEL
103 PROG EQU CURCHL+2  ; -> BASIC PROGRAM
104 NXTLN EQU PROG+2  ; -> NEXT LINE OF SOURCE CODE

151
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>DATADD: EQU NXTLIN+2</td>
</tr>
<tr>
<td>107</td>
<td>E_LINE EQU DATADD+2</td>
</tr>
<tr>
<td>108</td>
<td>K_CUR: EQU E_LINE+2</td>
</tr>
<tr>
<td>109</td>
<td>CH_ADD EQU K_CUR+2</td>
</tr>
<tr>
<td>110</td>
<td>X_PTR EQU CH_ADD+2</td>
</tr>
<tr>
<td>111</td>
<td>WORKSP: EQU X_PTR+2</td>
</tr>
<tr>
<td>112</td>
<td>STKBO: EQU WORKSP+2</td>
</tr>
<tr>
<td>113</td>
<td>STKNX: EQU STKBO+2</td>
</tr>
<tr>
<td>114</td>
<td>STKEND: EQU STKNX+2</td>
</tr>
<tr>
<td>115</td>
<td>BREG: EQU STKEND+2</td>
</tr>
<tr>
<td>116</td>
<td>MEM: EQU BREG+1</td>
</tr>
<tr>
<td>117</td>
<td>FLAGS: EQU MEM+2</td>
</tr>
<tr>
<td>118</td>
<td>ALDS: EQU 0</td>
</tr>
<tr>
<td>119</td>
<td>PRED: EQU 1</td>
</tr>
<tr>
<td>120</td>
<td>L_ST: EQU 2</td>
</tr>
<tr>
<td>121</td>
<td>CAPS: EQU 3</td>
</tr>
<tr>
<td>122</td>
<td>RETPOS: EQU 4</td>
</tr>
<tr>
<td>123</td>
<td>DELREP: EQU 5</td>
</tr>
<tr>
<td>124</td>
<td>DF_SZ: EQU FLAGS+1</td>
</tr>
<tr>
<td>125</td>
<td>OLDDPC: EQU S_TOP+2</td>
</tr>
<tr>
<td>126</td>
<td>OSPC: EQU OLDDPC+2</td>
</tr>
<tr>
<td>127</td>
<td>FLEX: EQU 0</td>
</tr>
<tr>
<td>128</td>
<td>UNFIND: EQU 1</td>
</tr>
<tr>
<td>129</td>
<td>INFL: EQU 5</td>
</tr>
<tr>
<td>130</td>
<td>L_ST: EQU 7</td>
</tr>
<tr>
<td>131</td>
<td>STRLEN: EQU FLEX+1</td>
</tr>
<tr>
<td>132</td>
<td>T_ADDR: EQU STRLEN+2</td>
</tr>
<tr>
<td>133</td>
<td>SEED: EQU T_ADDR+2</td>
</tr>
<tr>
<td>134</td>
<td>FRAME: EQU SEED+2</td>
</tr>
<tr>
<td>135</td>
<td>UDO: EQU FRAME+2</td>
</tr>
<tr>
<td>136</td>
<td>Coord: EQU UDO+2</td>
</tr>
<tr>
<td>137</td>
<td>P_POSN: EQU Coord+2</td>
</tr>
<tr>
<td>138</td>
<td>PR_CC: EQU P_POSN+1</td>
</tr>
<tr>
<td>139</td>
<td>ECHO_E: EQU PR_CC+2</td>
</tr>
<tr>
<td>140</td>
<td>S_POSN: EQU DFCC+2</td>
</tr>
<tr>
<td>141</td>
<td>SPOSN: EQU S_POSN+2</td>
</tr>
<tr>
<td>142</td>
<td>SCR_CT: EQU SPOSN+2</td>
</tr>
<tr>
<td>143</td>
<td>ATTR_P: EQU SCR_CT+1</td>
</tr>
<tr>
<td>144</td>
<td>FOREG: EQU 0</td>
</tr>
<tr>
<td>145</td>
<td>BLUE: EQU 0</td>
</tr>
<tr>
<td>146</td>
<td>RED: EQU 1</td>
</tr>
<tr>
<td>147</td>
<td>GREEN: EQU 2</td>
</tr>
<tr>
<td>148</td>
<td>BACKG: EQU 3</td>
</tr>
<tr>
<td>149</td>
<td>BLUEB: EQU 4</td>
</tr>
<tr>
<td>150</td>
<td>REDB: EQU 5</td>
</tr>
<tr>
<td>151</td>
<td>FORENB: EQU 6</td>
</tr>
<tr>
<td>152</td>
<td>FLASH: EQU 7</td>
</tr>
<tr>
<td>153</td>
<td>MASK_P: EQU ATTR_P+1</td>
</tr>
<tr>
<td>154</td>
<td>ATTR_T: EQU MASK_P+1</td>
</tr>
<tr>
<td>155</td>
<td>P_FLAGS: EQU ATTR_T+1</td>
</tr>
<tr>
<td>156</td>
<td>XOR_CHE: EQU 0</td>
</tr>
<tr>
<td>157</td>
<td>INVCHE: EQU 2</td>
</tr>
<tr>
<td>158</td>
<td>FCB: EQU 4</td>
</tr>
<tr>
<td>159</td>
<td>BCF: EQU 6</td>
</tr>
<tr>
<td>160</td>
<td>MEMBOT: EQU P_FLAGS+1</td>
</tr>
<tr>
<td>161</td>
<td>NMIADD: EQU MEMBOT+30</td>
</tr>
<tr>
<td>162</td>
<td>RANTOP: EQU NMIADD+2</td>
</tr>
<tr>
<td>163</td>
<td>P_RANT: EQU RANTOP+2</td>
</tr>
</tbody>
</table>

152
177 1**** ADDITIONAL
178 ERR_LN: EQU P_RAM+2 1POINTER TO ON ERROR LINE NUMBER FOR A GO-TO.
179 ERR_CLI EQU ERR_LN+2 1STORE LINE NUMBER IN WHICH ERROR OCCURRED.
180 ERR_SI EQU ERR_CLI+2 1STORES STATEMENT NUMBER IN WHICH ERROR OCCURRED
181 ERR_Ti EQU ERR_SI+1 1STORE FOR 'ERROR TYPE' AFTER A 'ON ERR'
182 SYSCON EQU ERR_Ti+1 1SYSTEM CONFIGURATION TABLE.
183 MAX_BANK EQU SYSCON+2 1LARGEST BANK NUMBER ASSIGNED
184 CURCBN EQU MAX_BANK+1 1BANK NUMBER OF THE CURRENT CHANNEL.
185 MSTBOT EQU CURCBN+1 1ADDRESS OF LOCATION ABOVE MACHINE STACK
186 VIDMOD EQU MSTBOT+2 1NOTE: UNUSED BYTE AFTER VIDMOD
187
188 1
189 ARSBUF1 EQU VIDMOD+2 1POINTER TO ARS BUFFER.
190 ARSFLO1 EQU ARSBUF1+2 1AROS FLAG - BIT 7 SET INDICATES AROS PRESENT.
191 BIT 4 SET INDICATES TXTLN POINTING TO AROS.
192 BIT 3 SET INDICATES DATADD POINTING TO AROS.
193 1these bits become important for the insert routine
194 1(pointers pointing to AROS should not be updated
195 1(for an insertion into ram).
196 ADATLN EQU ARSFLO1+1 1POINTER TO THE START OF THE CURRENT DATA LINE
197 1(AROS ONLY)
198 DTNLN EQU ADATLN+2 1LENGTH OF THE CURRENT DATA LINE (AROS ONLY).
199 STRMP1 EQU DTNLN+2 1CURRENT STREAM NUMBER, USED FOR BUS EXPANSION
200 .UNIT.DEVICES.
201 MSTACK EQU 6200H 1LOCATION ABOVE MACHINE STACK
202 DRIVE EQU 6840H 1START OF 'DRIVES' AREA
203 BANK_ENABLE EQU 6499H
204 CALL_BANK EQU 6500H
205 MOVE_SZ EQU DRIVE-6000H
206 DEST7 EQU OFFFH-MOVE_SZ+1
207 FIX EQU DEST7-6000H
208 CALL_BANK EQU CALL_BANK+FIX
209 GOTO_BANK EQU 6572H 1ADDRESS OF "GO TO BANK" BANK SWITCHING
210 1AWARD.
211 XFER_BYTES EQU 6722H 1INDIRECT DATA TRANSFER BETWEEN BANKS.
212 GOTO_EXT EQU 6815H 1FOR INITIALIZATION CODE IN HOME BANK
213 1EXTENTION.
214 SVLM EQU 01ABH 1ADDRESS OF TAPE ROUTINES FOR SAVE, LOAD
215 1VERIFY AND MERGE COMMANDS.
216 BLDSCT EQU 09F4H 1ADDRESS OF Initialization ROUTINE TO
217 1BUILD THE SYSTEM CONFIGURATION TABLE.
218 RESCRI EQU 0C4CH 1ADDRESS OF RESET ROUTINE TO ADD DEVICES.
219 PASSING EQU 0FO9H 1ADDRESS OF ROUTINE TO PUSH PARAMETERS TO
220 1THE BEU ROUTINES ONTO THE MACHINE STACK.
221
222 1***
223
224 1OTHER EQUATES
225
226 1RESTARTS
227
228 ERROR1 EQU 8
229 WRCH1 EQU 16
230 INL_SP1 EQU 24
231 NXT_1SH EQU 32
232 CALCTR1 EQU 40
233 COPYUP1 EQU 48
234
235 NOSIZE EQU 5 18 OF BYTES IN A FLOATING POINT NUMBER
236 DIGT EQU '0' 1DIGIT+N IS CODE FOR DIGIT N
237 LETTER EQU 0 1LETTER-'ALPHA' IS CODE FOR LETTER ALPHA
238 DEBDEL EQU 5 1NO. CONSECUTIVE TIMES KB SWITCH FOUND OPEN BEFORE
239 1KEY RECONIGNED RELEASED.
240
241 1CONTROL CHARACTERS (APPEARING ON STREAM)
242
243 COM_CCI EQU 6 1PRINT COMMA
244 EDIT_CCI EQU 7 1EDIT
245 BS_CCI EQU 8 1BACKSPACE (CURSOR LEFT)
246 CRT_CCI EQU 9 1CURSOR RIGHT
247 CD_CCI EQU OAH 1CURSOR DOWN
248 CU_CCI EQU OBH 1CURSOR UP
249
250
251 153
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>RUB_CC</td>
<td>EQU 0CH</td>
</tr>
<tr>
<td>251</td>
<td>CR_CC</td>
<td>EQU 0DH</td>
</tr>
<tr>
<td>252</td>
<td>NL</td>
<td>EQU CR_CC</td>
</tr>
<tr>
<td>253</td>
<td>SLUG</td>
<td>EQU 0EH</td>
</tr>
<tr>
<td>254</td>
<td>FORECC</td>
<td>EQU 10H</td>
</tr>
<tr>
<td>255</td>
<td>THE CONTROL CHARS FOR FORE, BACK, FLASH, BRIGHT, INVERT &amp; OVER ARE CONSECUTIVE IN THAT ORDER.</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>AT_CC</td>
<td>EQU 16H</td>
</tr>
<tr>
<td>257</td>
<td>TAB_CC</td>
<td>EQU 17H</td>
</tr>
<tr>
<td>258</td>
<td></td>
<td>CONTROL CHARACTERS (RECEIVED FROM KEYBOARD)</td>
</tr>
<tr>
<td>259</td>
<td></td>
<td>STEADY</td>
</tr>
<tr>
<td>260</td>
<td>STY_KC</td>
<td>EQU 0</td>
</tr>
<tr>
<td>261</td>
<td>FLASH</td>
<td>EQU 1</td>
</tr>
<tr>
<td>262</td>
<td>MSL_KC</td>
<td>EQU 2</td>
</tr>
<tr>
<td>263</td>
<td>LOW_KC</td>
<td>EQU 3</td>
</tr>
<tr>
<td>264</td>
<td>NLV_KC</td>
<td>EQU 4</td>
</tr>
<tr>
<td>265</td>
<td>INV_KC</td>
<td>EQU 5</td>
</tr>
<tr>
<td>266</td>
<td>CSL_KC</td>
<td>EQU 6</td>
</tr>
<tr>
<td>267</td>
<td>TML_KC</td>
<td>EQU 0EH</td>
</tr>
<tr>
<td>268</td>
<td>TKN_MODE</td>
<td>EQU 0FH</td>
</tr>
<tr>
<td>269</td>
<td>ORG_KC</td>
<td>EQU 1OH</td>
</tr>
<tr>
<td>270</td>
<td>FNL_KC</td>
<td>EQU 18H</td>
</tr>
<tr>
<td>271</td>
<td>BG_KC</td>
<td>EQU 18H</td>
</tr>
<tr>
<td>272</td>
<td>SPACE</td>
<td>EQU 1</td>
</tr>
<tr>
<td>273</td>
<td>STRING</td>
<td>QUOTE</td>
</tr>
<tr>
<td>274</td>
<td>DOLLAR</td>
<td>EQU 1</td>
</tr>
<tr>
<td>275</td>
<td>DOLLAR_SIGN</td>
<td>EQU 1</td>
</tr>
<tr>
<td>276</td>
<td>COLON</td>
<td>EQU 1</td>
</tr>
<tr>
<td>277</td>
<td>COMMA</td>
<td>EQU 1</td>
</tr>
<tr>
<td>278</td>
<td>KET</td>
<td>EQU 1</td>
</tr>
<tr>
<td>279</td>
<td>RESTARTS</td>
<td></td>
</tr>
<tr>
<td>280</td>
<td>ERFOR</td>
<td>EQU 8</td>
</tr>
<tr>
<td>281</td>
<td>ERR</td>
<td>WRCH</td>
</tr>
<tr>
<td>282</td>
<td>ERR</td>
<td>IGN_SP</td>
</tr>
<tr>
<td>283</td>
<td>ERR</td>
<td>NXT_ISI</td>
</tr>
<tr>
<td>284</td>
<td>ERR</td>
<td>CALCCTR</td>
</tr>
<tr>
<td>285</td>
<td>ERR</td>
<td>COPYUP</td>
</tr>
<tr>
<td>286</td>
<td>NOSIZE</td>
<td>EQU 5</td>
</tr>
<tr>
<td>287</td>
<td>0 OF BYTES IN A FLOATING POINT NUMBER</td>
<td></td>
</tr>
<tr>
<td>288</td>
<td>DIGIT</td>
<td>EQU '0'</td>
</tr>
<tr>
<td>289</td>
<td>DIGIT+N IS CODE FOR DIGIT N</td>
<td></td>
</tr>
<tr>
<td>290</td>
<td>LETTER</td>
<td>EQU 0</td>
</tr>
<tr>
<td>291</td>
<td>LETTER=&quot;ALPHA&quot; IS CODE FOR LETTER ALPHA</td>
<td></td>
</tr>
<tr>
<td>292</td>
<td>DEBDEL</td>
<td>EQU 5</td>
</tr>
<tr>
<td>293</td>
<td>NO. CONSECUTIVE TIMES KB SWITCH FOUND OPEN BEFORE</td>
<td></td>
</tr>
<tr>
<td>294</td>
<td>KEY RECKONED RELEASED.</td>
<td></td>
</tr>
<tr>
<td>295</td>
<td>CONTROL CHARACTERS (APPEARING ON STREAM)</td>
<td></td>
</tr>
<tr>
<td>296</td>
<td>COM_LCI</td>
<td>EQU 6</td>
</tr>
<tr>
<td>297</td>
<td>PRINT COMMA</td>
<td></td>
</tr>
<tr>
<td>298</td>
<td>EDT_LCI</td>
<td>EQU 7</td>
</tr>
<tr>
<td>299</td>
<td>EDIT</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>BS_LCI</td>
<td>EQU 8</td>
</tr>
<tr>
<td>301</td>
<td>BACKSPACE (CURSOR LEFT)</td>
<td></td>
</tr>
<tr>
<td>302</td>
<td>CRT_LCI</td>
<td>EQU 9</td>
</tr>
<tr>
<td>303</td>
<td>CURSOR RIGHT</td>
<td></td>
</tr>
<tr>
<td>304</td>
<td>CD_LCI</td>
<td>EQU 0AH</td>
</tr>
<tr>
<td>305</td>
<td>CURSOR DOWN</td>
<td></td>
</tr>
<tr>
<td>306</td>
<td>CU_LCI</td>
<td>EQU 0BH</td>
</tr>
<tr>
<td>307</td>
<td>CURSOR UP</td>
<td></td>
</tr>
<tr>
<td>308</td>
<td>RUB_LCI</td>
<td>EQU 0CH</td>
</tr>
<tr>
<td>309</td>
<td>RUBOUT</td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>CR_LCI</td>
<td>EQU 0DH</td>
</tr>
<tr>
<td>311</td>
<td>CARRIAGE RETURN (NEWLINE)</td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>NL</td>
<td>EQU CR_LCI</td>
</tr>
<tr>
<td>313</td>
<td>SLUGL</td>
<td>EQU 0EH</td>
</tr>
<tr>
<td>314</td>
<td>THE CONTROL CHARS FOR FORE, BACK, FLASH, BRIGHT, INVERT &amp; OVER ARE CONSECUTIVE IN THAT ORDER.</td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>AT_LCI</td>
<td>EQU 16H</td>
</tr>
<tr>
<td>316</td>
<td>PRINT TAB</td>
<td></td>
</tr>
<tr>
<td>317</td>
<td>AT_LCI</td>
<td>EQU 17H</td>
</tr>
<tr>
<td>318</td>
<td>PRINT TAB</td>
<td></td>
</tr>
<tr>
<td>319</td>
<td>CONTROL CHARACTERS (RECEIVED FROM KEYBOARD)</td>
<td></td>
</tr>
<tr>
<td>320</td>
<td>STY_KC</td>
<td>EQU 0</td>
</tr>
<tr>
<td>321</td>
<td>FLASH</td>
<td></td>
</tr>
<tr>
<td>322</td>
<td>MSL_KC</td>
<td>EQU 1</td>
</tr>
<tr>
<td>323</td>
<td>LOW_KC</td>
<td>EQU 2</td>
</tr>
<tr>
<td>324</td>
<td>NLV_KC</td>
<td>EQU 3</td>
</tr>
<tr>
<td>325</td>
<td>INV_KC</td>
<td>EQU 4</td>
</tr>
<tr>
<td>326</td>
<td>NORMAL VIDEO</td>
<td></td>
</tr>
<tr>
<td>327</td>
<td>INVERSE VIDEO</td>
<td></td>
</tr>
</tbody>
</table>
268 CSL_KCI EQU $E  ; CAPS SHIFT LOCK TOGGLE
269 THL_KCI EQU $0E ; TOKEN MODE
270 ORLK_ KCI EQU $OFH ; GRAPHICS MODE
271 FOK_ KCI EQU $10H ; FOREGROUND BLACK
272 BOK_ KCI EQU $18H ; BACKGROUND BLACK
273 274 SPACE EQU ’ ’ ; STRING "QUOTE"
275 QUOTE EQU ” ” ; DOLLAR SIGN
276 DOLLAR EQU ’$’ ; DOLLAR SIGN
277 COLON1 EQU ’:’ ;<
278 COMMA EQU ’,’ ;<
279 KET EQU ’>’ ; <
280 BRA EQU ’<’ ; <
281 GTI EQU ’>’ ;<
282 MINUS EQU ‘-’ ;<
283 EQUAL EQU ‘=’ ;<
284 PLUS EQU ‘+’ ;<
285 STROKE EQU ’/’ ;
286 POWER EQU ’^’ ;
287 POINT EQU ’.’ ;
288 SHARP EQU ’#’ ;
289 STD_ ORI EQU $0H ; 1ST STANDARD GRAPHIC
290 UD_ ORI EQU $90H ; 1ST USER-DEFINED GRAPHIC
291 292 1TOKENS
293 294 TOKI EQU $0AH ; 1ST TOKEN
295 RNDTOKI EQU $0AH ; ’RND’
296 INKEYI EQU $0AH ; ’INKEY’
297 PII EQU $0AH ; ’PI’
298 FNLTKI EQU $0AH ; ’FN’
299 PNT TKI EQU $09H ; ’POINT’
300 SCRNTKI EQU $0AH ; ’SCREEN’
301 ATTRKI EQU $0AH ; ’ATTR’
302 ATI EQU $0AH ; ’AT’
303 TIKFN EQU FNL TK ; 1ST TOKEN TO REQUIRE A SPACE AFTER
304 TABI EQU $0AH ; ’TAB’
305 VALSTK EQU $0AH ; ’VAL’
306 LO_MONI EQU $0FH ; TOKEN FOR 1ST MONADIC OPTR AFTER VAL$ (CODE)
307 BINTKI EQU $04H ; ’BIN’
308 ORTI EQU $05H ; ’OR’ NB THE TOKENS FOR OR, AND, <=, >=, <> ARE
309 ’CONSECUTIVE IN THAT ORDER.
310 LINTKI EQU $0AH ; ’LINE’
311 THENI EQU $02H ; ’THEN’
312 TOI EQU $02H ; ’TO’
313 STEP EQU $02H ; ’STEP’
314 DFNTKI EQU $02H ; ’DEF’
315 MIKWKI EQU DFNT K ; 1ST TOKEN THAT IS A KEYWORD RATHER THAN... OPERATOR
316 CAT_TKI EQU $0FH ; ’CAT’
317 FORMTKI EQU $0DH ; ’FORMAT’
318 MOVE TKI EQU $01H ; ’MOVE’
319 DEL TKI EQU $02H ; ’DELETE’
320 OPN_TKI EQU $03H ; ’OPEN’
321 CLCTKI EQU $04H ; ’CLOSE’
322 MOE_TKI EQU $05H ; ’MERGE’
323 VFY_TKI EQU $06H ; ’VERIFY’
324 BEEP TKI EQU $07H ; ’BEEP’
325 ARC_TKI EQU $08H ; ’ARC’
326 FOTKI EQU $09H ; ’FOREGROUND’ NB THE TOKENS FOR FORE, BACK, FLASH,
327 328 ’ORDER.
329 INVTOKI EQU FOTKIS ; ’INVERT’
330 OUT TKI EQU $0FH ; ’OUT’
331 LPL_TKI EQU $0EH ; ’LPRINT’
332 L_ TKI EQU $0EH ; ’LLIST’
333 STOPTKI EQU $0EH ; ’STOP’
334 READTKI EQU $03H ; ’READ’
335 DATATKI EQU $04H ; ’DATA’
336 RESTTKI EQU $05H ; ’RESTORE’
337 NEX TOKI EQU $0FH ; ’NEXT’
338 DUMPTKI EQU $0FH ; ’COPY’
339 340 BORDPT EQU $0FEH ; OUTPUT PORT FOR SETTING BORDER COLOUR
PR_INT: EQU 0FBH  
PR_OUT: EQU 0FEH  
KB_PORT: EQU 0FEH  
O_PORT: EQU 0FEH  
I_PORT: EQU 0FEH  
TAPE_INPUT: EQU 0  
DOCK_HORIZ: EQU 04H  
BANK_DATA: EQU 06H  
BANK_COMMANDS: EQU 0DH  
HOME_ROM: EQU 0FFH  
ADDRESS: EQU 4000H-96*8  
EJECT  

; CALCULATOR COMMANDS. THE DESCRIPTIONS, T & S STAND FOR 
; THE TOP & SECOND FROM TOP ON THE CALCULATOR STACK, 
; WHERE NECESSARY, FULLER DESCRIPTIONS CAN BE FOUND AT THE 
; CODE FOR THE RELEVANT ROUTINES. 

; THE FOLLOWING COMMANDS HAVE THE TOP POINTERS HL & DE (BUT 
; NOT (STKNN)) DECREMENTED FOR THEM BY CALCTR BEFORE THEY 
; ARE CALLED (STKDN).  

IFJUMP: EQU 0  
IS,T -> S1 RELATIVE JUMP CONDITIONAL ON VALUE OF T.  
EXCH: EQU IFJUMP+1  
SUBI: EQU EXCH+1  
DIVI: EQU TIMES+1  
DIVE: EQU TIMES+1  
ORI: EQU POWER+1  
ANDI: EQU OR+1  
GTI: EQU AND+4  
ADDI: EQU AND+7  
STOG: EQU ADD+1  
CONCAT: EQU STOG+7  
IT0 -> VALS T0  
VALS: EQU CONCAT+1  
IT0 -> ADDRESS OF BIT PATTERN FOR CORRESPONDING 
USSRS: EQU VAL+1  
INKEY: EQU USRS+1  
INKEY: EQU INKEY+1  
CODE: EQU INKEY+1  
VAL: EQU CODE+1  
LEN: EQU VAL+1  
SIN: EQU LEN+1  
COS: EQU SIN+1  
TAN: EQU COS+1  
ASIN: EQU TAN+1  
ACOS: EQU ASIN+1  
ATAN: EQU ACS+1  
ATN: EQU ATAN+1  
EXP: EQU LN+1  
INT: EQU EXP+1  
ROOT: EQU INT+1  
SQR: EQU ROOT+1

156
413 ABS: EQU SGN+1  \{ABSOLUTE\} T -> IT
414 PEEK: EQU ABS+1  IT -> PEEK T
415 IN: EQU PEEK+1  IT -> IN T
416 USR: EQU IN+1  IT -> USR T
417 STR: EQU USR+1  IT -> STR* T
418 CHR: EQU STR+1  IT -> CHR* T
419 NOT: EQU CHR+1  IT -> BOOLEAN (T = 0)
420 ZERO?: EQU NOT
421 DUP: EQU NOT+1  \{DUPLICATE\} T -> T,T
422 INTDIV: EQU DUP+1  \{INTEGRAL DIVISION\} S,T -> S MOD T, INT(S/T)
423 JUMP: EQU INTDIV+1  \{PROGRAM CONTROL \= RELATIVE JUMP BY FOLLOWING BYTE
424 LITERAL: EQU JUMP+1  \{STACKS FOLLOWING NUMBER.
425 LOOP: EQU LITERAL+1  \{LIKE IZLOG JNZ
426 MINUS?: EQU LOOP+1  IT -> BOOLEAN (T < 0)
427 PLUS?: EQU MINUS+1  IT -> BOOLEAN (T > 0)
428 QUIT: EQU PLUS?+1  \{RETURNS CONTROL TO IBO
429 ANOLE: EQU QUIT+1  IT -> Y WHERE -1 <= Y <= +1 & SIN T = SIN (PI/2*Y)
430 IN 1  MEMORY 0 I = TRUE IF T IN 2ND OR 3RD QUADRANT
431 TRUNC: EQU ANOLE+1  \{TRUNCATE\} T -> INTEGER TRUNCATION OF T TOWARDS 0.
432 XEQF: EQU TRUNC+1  \{EXECUTES (BREG) AS A CALCULATOR INSTRUCTION
433 XEQ: EQU XEQF+1  1S,T -> S, 10**T
434 FLOAT: EQU XEQ+1  \{FORCED INTO FLOATING POINT FORM
435 \}
436 THE FOLLOWING COMMANDS HAVE ADDED TO THEM AN OPERAND, N.
437 438 CBSV: EQU 80H  \{SUMS N TERMS OF CHEBYSHEV SERIES (SEE CBSV).
439 CONST: EQU CBSV+20H \{CONSTANT\} T -> T, NTH CALCULATOR CONSTANT
440 MINUS: EQU CONST+6  \{CALCCTR CONSTANT EQUAL TO -1
441 COPY: EQU CONST+20H  IT -> T T COPIED TO NTH CALCULATOR MEMORY
442 MEMORY: EQU COPY+20H  IT -> T, CONTENTS OF NTH CALCULATOR MEMORY
443 444 OP_TK: EQU LO_MON-LO_MON  \{TOKEN FOR MONADIC Optr C IS OP_TK+C
445 HI_MON: EQU OP_TK-CHR  \{TOKEN FOR LAST MONADIC Optr EXCEPTING NOT
446 MONOP: EQU LO_MON.OR.OCNH  \{OPERATION CODE FOR LO_MON, TOP 2 BITS SET.
447 LO_NOP: EQU OP_TK+SN  \{TOKEN FOR 1ST (NUMBER) NUMBER Optr AFTER -
448 HI_NOP: EQU OP_TK+USR  \{TOKEN FOR LAST (NUMBER) NUMBER Optr
449 450 #LIST ON

157
APPENDIX C-1

64 COLUMN MODE

TIMEX COMPUTER CORPORATION

APPLICATION DEVELOPMENT LIBRARY

Application Software Command 001
64-COLUMN MODE

Date: 12/15/83
LSI Number: 001
Version: 002
Author: Carol Cataran

Name: 64 Column Mode Support

Description:

This component provides support to the application programmer for using the 64-column mode feature of the T5 2068. The services include opening/closing the second display file (moving the machine stack), QS RAM routines and BASIC structures), PRINT position control, attribute control, clear screen and scroll screen services and display of characters. For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and designated non-zero values for other conditions. The interface from BASIC is by means of POKING the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code located starting at chunk 7 (E860n/$7344).

Version
Description
Date

001
Original
11/28/83

002
ADD
12/15/83

1) Set and Use-
Defined Graphics
to INTERRUPT and
GETCHAR.

2) WRITE (Write String)

---
APPLICATION ARCHIVE

Name: SETMODE (SETMOD from BASIC - parameter to VIMODE)

Input: MODE (0=normal, 6=64 column mode)

From machine code: Register A
From BASIC: In VIMODE

Description:

Sets specified video mode, opening or closing the second display file where needed. This involves determining if there is enough free RAM to open the second display file and if so, moving the BASIC structures and machine language variables area up and the UDG area down to make space for the machine stack and S$ RAM routines at the top of memory. The second display file is cleared to zeroes. The affected system and internal variables are updated or initialized (see Usage Section). When returning to Mode 0 from 64 Column mode, the structures are returned to their normal locations.

Output: BC = 0 Successful
BC = 1 Invalid parameters (not equal to 0 or 6)
BC = 2 Not enough memory

Name: CLRSCN (CLRSCB from BASIC - parameters to CLSCTL)

Input: Line Count (1-26)
Starting Line Number (0-255)

From Machine Code: Line Count in Register B
Starting Line in Register C

From BASIC: Starting Line Number in CLSCTL
Line Count in CLSCTL + 1

Description:

Clears to background color (PAPER) the designated number of lines, beginning with the starting line number. Line 0 with a clears the entire screen. Upon return the cursor position is at the beginning of the first line cleared.

Output: BC = 0 for successful completion
BC = 1 invalid parameters
(Line Number = Line Count (1 or 326)

Name: SETCM (SETCB from BASIC - parameters to LINCOL)

Input: Line Number (0-23)
Column Number (0-63)

From Machine Code: Line Number in Register B
Column Number in Register C

From BASIC: Column Number in LINCOL
Line Number in LINCOL + 1

Description:

Converts the requested position to internal format, determines display file address, and stores the values for use by the next display character operation. Note that once established, the position is updated automatically when a character is displayed so that it is only necessary to set the position when sequential display is not desired.

Output: BC = 0 for successful completion
BC = 1 for invalid parameters (Line Number > 23,
Column Number > Line Length-1)
Name: SETATT (SETAT from BASIC - parameter to ATCTL)

Input: Attribute byte - bit 7 - FLASH
       bit 6 - BRIGHT
       bit 5 - P
       bit 4 - A
       bit 3 - PER
       bit 2 - I
       bit 1 - N
       bit 0 - K

From Machine Code: Register A
From BASIC: IN ATCTL

Description:
The specified INK color (0-7) is used to set the video mode
hardware and to update VIDMODE. The complementary PAPER color is
fixed by the INK selection. FLASH and BRIGHT are fixed at zero
by the hardware. Note that in 64 column mode the entire screen
has the same attributes.

Output: BC = 0 Successful

Name: SETMSK (SETMS from BASIC - parameters to MSKCTL)

Input: Mask byte - bit 0 - OVER
       bit 2 - INVERSE

From Machine Code: Register A
From BASIC: MSKCTL

Description:
The specified mask is stored for application to all subsequent
display character operations. (OVER = 1 implies new character
combined with old using an XOR operation; INVERSE = 1 implies
character is inverted).

Output: BC = 0 for successful completion

Name: WCHR (WCHR from BASIC - parameter to DATAB)

Input: Character code for character to be displayed

  2OH TO 7FH - Std. 72x9 Character Set
  80H TO 9FH - Std. Graphics Set
  A0H TO A5H - User-Defined Graphics Set

From Machine Code: Register A
From BASIC: IN DATAB

Description:
Displays character at current cursor position, applying current
mask. Moves cursor position to next sequential position. If
character would start a new line after DOTLM (see Usage section)
and the scroll count (variable SCRLCT) decrements to zero, the
character will not be displayed and return will be made with BC =
3 (screen full). If the scroll count does not decrement to zero,
the screen will be scrolled up one line using the information in
SCRLCT and the new line started at the vacated line.

Output: BC = 0 for successful completion
BC = 1 Invalid character code
BC = 3 for screen Full

160
NAME: WSTRS (WSTBS from BASIC - String Identifier in PARMS)

Input: Character Code String

From machine code Address of string in HL
Count in BC
From BASIC: String Variable Identifier in
System Variable PARMS = 23747 (SCC3H)

Description:

Displays the characters from the string, beginning at the current
cursor location and continuing sequentially until the count
expires, or "Screen Full" is detected (see WCHR description and
Usage Section on Automatic Scrolling). For the Screen Full
condition, the remaining count is stored in the internal variable
Sقرأ the var for access by the user.

Note: Characters within the string which are outside of the
supported range (32 through 126, 20H-40H) will be
ignored. E.g., BASIC Tab codes and control codes
embedded in an INPUT string will not be displayed or
decoded.

From BASIC, POKE the code for the string variable identifier into
PARMS prior to invoking WSTRS, e.g.

0000 LET aa=----------string-------
0010 POKE 23747,CODE "aa"
0015 IF USR (WSTBS)<>0 THEN ------
(continue)

Output:

BC = 0 Successful
BC = 2 BASIC - String not found
BC = 3 Screen Full - Remaining Count in Sقرأ the var (HL=Current Address in String)

NAME: SCROLL (SCRLB from BASIC - parameters to SCRTC)

Input: Line Count (1-23)
Starting Line Number (1-23)

From machine code:
Line Count in B
Starting Line in C
From BASIC:
Starting Line in SCRTL
Line Count in SCRTC + 1

Description:

Scrolls the designated number of lines up 1 position, starting at
the specified line number and inserts a blank line at the bottom
of the scrolled area. Line 1 with a count of 23 will scroll
the entire screen up 1 line. Upon return the cursor position is at
the beginning of the inserted blank line.

Note: See Usage Section on "automatic" scrolling.

Output: BC = 0 Successful
BC = 1 Invalid Parameters
(Line Number + Line Count < 1 or > 24)

NAME: GETATT (GETATB from BASIC - parameters to GETCTL)

Input: As for GTERM

Description:

Returns in register C of the BC register pair the attribute byte
for the character at the designated screen position. Note that
in 4x column mode the entire screen has common attributes, The
value returned will describe the current selection.

Output: BC = 1 for invalid parameters
BC = attribute byte (as for GETATT)
Name: GETCHAR (GETCHAR from BASIC — parameters to GETCTL)

Input: Line/Column position as for SETCUR

From Machine Code: Line Number in B
Column Number in C

From BASIC: Column Number in GETCTL
Line Number in GETCTL + 1

Description:

Returns in register C of the BC register pair the character code for the character at the designated screen position. If no match against the character set (including the standard and user-defined graphics) is found, zero is returned. (Character code or zero also returned in B.)

Note: Positions "printed" using the OVER technique will return zero if they do not match against any single character.

Output: BC = 0 for no find
BC = 1 invalid parameters
BC = character code (20H-64H)

Name: GETCUR (GETCUR from BASIC)

Input: None

Description:

Returns in the BC register pair and in the BASIC parameter location LINCOL, the current print position (where the next character would be displayed).

Output: B = Line Number (0-23)
C = Column number (0-63)

BASIC: LINCOL = Column number
LINCOL + 1 = Line number

Note: If the last character was printed at Col.63 of Line 23 (last position on the screen), then Col.0/Line 23 will be returned.

USASI

Memory Usage:

This package of machine code routines includes the following internal variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
</table>
| SCRCTL| 2    | Scroll Control
LSB = Starting line number
MSB = Number of lines to be scrolled |
| SOTLM | 1    | Bottom Line - Line number (0-23) after which test for scroll will be made. |
| SCRLCT| 1    | Scroll Count - Number of times + 1 that automatic scroll will be done. When decremented to zero, data will not be displayed and a condition code will be returned to user. |
| CNTBL | 2    | Character Table (Base Address - 100H) |
| CNTBL | 2    | Std.Graphics Character Table (Base - 100H) |
| LIMLEN| 1    | Line Length - 64 when in 64-Column Mode |
| CURPOS| 2    | Current Position (Internal Format)
LSB = Column Position
MSB = Line Position |
| DFADDR | 2   | Current Display File Address |
| MASKB | 1    | Mask Byte (bit 0 = OVER)
(bit 1 = INVERSE) |

162
ATTGET 1 Attribute Byte (bits 0 - 2 - INK)
(bits 3 - 5 - PAPER)
(bit 6 - BRIGHT (Set to zero)
(bit 7 - FLASH by M/W in
64-col.mode)

GETMNE 1 "Get" Index - Used by GETCHAR

STRICT 2 String Count - Contains remaining byte count
when BC=3 (Screen Full) is
returned from the write String
service WRSTRG (WRSTG).

== == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == == =
NOTES:

1. All screen operations done by the SYSTEM ROM (PRINT, LIST, EDIT, FORMAT, CLS, scrolling, etc.) relate only to the main display file. This means that only the even columns on the 80-column main screen will be affected. You will want to execute the CLEAR SCREEN function in this module to guarantee that no data in the second display file interferes with the use of a system screen service, e.g., prior to using a LIST.

2. The COPY command will print only the even columns of the screen to the 1040 Printer.

3. During tape operations, the border will not change while in 64-column mode since this is fixed by the hardware to conform to the paper color.

4. The SAVE filename SCREEN will save only the main Display File data. The second can be saved by a SAVE filename CODE 24575. Be careful that you have the computer in 64-column mode when you load this data or you will overwrite the OS ROM routines and "crash" the system! (The count for saving the display file is for the data portion only since the attribute file area from 7800H-7AFFF (30720-31487) is not used by the video mode M/N in 64-column mode.)

```
464 = ASC 001 64+COL.MODE SUPPORT
IO44.14C

CRI200/11 version 10.38.16 16-MAR-94 13:01:13

NAME 464 = ASC 001 64+COL.MODE SUPPORT

SUBTL VERSION LEVEL CONTROL

SUBTL DEFINITIONS

```

164
AND D7A
JN 1,NDSTRG
CP D
JL,NDSTRG
JL,NDSTRG
CALL RCL
ADD CPX, CPY
CALL WP
OD C
SJ
AGAIN
WSTRE IC,ML
LD C,(ML)
INC ML
INC ML
LD A,B
LD A,B
OR A,G
RET 2
RETURN IP SO
ENTRY FROM MACHINE CODE WITH
ADDRESS OF CHAR.CDE "STRING" IN HL AND LENGTH IN RC
WSTRE LO A,(ML)
GET CODE
CALL WCHR
WRITE A
LD A,C
GET ERROR
JR HL,WP
EXIT IF INVALID CODE OR IP SCREEN FULL
WSTRE PDP BC
COUNT
WSTRE RC
ADDRESS
BC
HL
WSTRE RC
BC
HL
WRITE NEXT CHAR.
RET
RETURN BC+8
WSTRE LO A,C
SAVE ERROR
WSTRE J,WP
JL,WP
WSTRE RC
BC
HL
ADDRESS OF REMAINING COUNT TO ML
STORE REMAINING COUNT
ADDRESS TO ML
WSTRE RC
BC
RETURN BC+3
WSTRE RC
BC
RETURN BC+1
WSTRE LO C,2
RETURN BC+2 IF STRING NOT FOUND
SUBTL POSITION CONTROL
HERE FROM BASIC ENTRY, PARAMETERS
LOAD BC),(LINCOLD)
ENTRY WITH LINE/COL, IN BC REG.
CALL TSTPAR
CALL UCYPN
CALL U4D3N
CALL U4D4N
CALL SCRL
CALL SCRL
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
MORE WITH CONTROLS IN BC
LOAD, OF LINES
GETTING STARTING LINE ND.
GET VALIDITY
ERROR IF COUNTS
ERROR IF BC+1
ERROR IF BC+2A
DO SCRL
RETURN BC+0
SCROLL SCROLL
MORE FROM BASIC ENTRY TO SCROLL
SCREEN
GET CONTROL IMP.
SUB C
LD B,A
SG A, (CLINK)
INC A
LD C,A
CALL LDW
LD D,A
LD E,C
POP BC
.Push DE
CLSC
LD A,L
ALCA
CLCA
MUL A
LD A,

Look at # of lines this block

SUB C
LD B,A
LD A,A
LD A,A
SUB C
LD B,A

Compare to total line count
Next line

CLSC
LD A,B

No. of lines

Remainning lines

Scan count

CLSC
LD A,B

clear dp

CLSC
LD A,B

Get control info.

Inc DE
LD B,A

Restore dp

CLSC and 00H

Next scan row

Push BC

Scan count

Total line count

TOTAL LINES

Save DE, lines this block

Sub total line count

Remaining line count

Clear this block

.JR CLSE

Subtotal attribute control

Here from basic entry to set attributes

LD A, (ATTCTL)

Here with attribute byte in A

LD A, (WIDMOD)

Test if dpz open

Save DE

Invalid if dpz not open

Sweep to H/W position

Set h/w mode value

Set h/w mode

Value for h/w

Preserve bits 0, 1, 2, 3

Set video mode value

Update v/0 mode

Ink selection in bits 3-5

Get complementary color

Combine with ink

Save in variable

Return bc0

Here from basic entry to set mask

LD A, (MSKCTL)

Get control info.

Entry with mask value in A

Store mask for application to future
SUBTL "GET" ROUTINES

0240"  FD 48 0019" 1 HERE FROM BASIC ENTRY TO GET CHARACTER

031"  GE 0019" 1 (RETURNS CODE FOR CHARACTER AT

032"  LE 0019" 1 POSITION SPECIFIED IN GE.)

033"  SD 0019" 1 GET CONTROL SIGNAL.

034"  IE 0019" 1 ENTRY POS. POSITION IN GE.

035"  GE 0019" 1 CALL GETPAR.

036"  SD 0019" 1 I TEST PARAMETERS.

037"  ID 0019" 1 CALL CONVR.

038"  SD 0019" 1 CONVERT TO INTERNAL FORMAT.

039"  ID 0019" 1 (GET DISPLAY FILE ADDRESSES.)

040"  ID 0019" 1 (CHARACTER TABLE.)

041"  ID 0019" 1 (NO. OF UNTRACABLE CHARACTERS)

042"  ID 0019" 1 (ADJUSTMENT INDEX)

043"  ID 0019" 1 (GETPOSN.

044"  ID 0019" 1 (ADJUST TO START OF TABLE)

045"  ID 0019" 1 (IF ADRSS. IN DISPLAY.SET IN ML)

046"  ID 0019" 1 (SAVE COUNT)

047"  ID 0019" 1 (SAVE ADRS. IN CP)

048"  ID 0019" 1 (SAVE ADRS. IN CHAR.TAB.

049"  ID 0019" 1 (SCAN ROW FROM CP)

050"  ID 0019" 1 (TEST AGAINST CHAR. SET)

051"  ID 0019" 1 (MATCH)

052"  ID 0019" 1 (MATCH)

053"  ID 0019" 1 (MATCH)

054"  ID 0019" 1 (MATCH)

055"  ID 0019" 1 (MATCH)

056"  ID 0019" 1 (MATCH)

057"  ID 0019" 1 (MATCH)

058"  ID 0019" 1 (MATCH)

059"  ID 0019" 1 (MATCH)

060"  ID 0019" 1 (MATCH)

061"  ID 0019" 1 (MATCH)

062"  ID 0019" 1 (MATCH)

063"  ID 0019" 1 (MATCH)

064"  ID 0019" 1 (MATCH)

065"  ID 0019" 1 (MATCH)

066"  ID 0019" 1 (MATCH)

067"  ID 0019" 1 (MATCH)

068"  ID 0019" 1 (MATCH)

069"  ID 0019" 1 (MATCH)

070"  ID 0019" 1 (MATCH)

071"  ID 0019" 1 (MATCH)

072"  ID 0019" 1 (MATCH)

073"  ID 0019" 1 (MATCH)

074"  ID 0019" 1 (MATCH)

075"  ID 0019" 1 (MATCH)

076"  ID 0019" 1 (MATCH)

077"  ID 0019" 1 (MATCH)

078"  ID 0019" 1 (MATCH)

079"  ID 0019" 1 (MATCH)

080"  ID 0019" 1 (MATCH)

081"  ID 0019" 1 (MATCH)

082"  ID 0019" 1 (MATCH)

083"  ID 0019" 1 (MATCH)

084"  ID 0019" 1 (MATCH)

085"  ID 0019" 1 (MATCH)

086"  ID 0019" 1 (MATCH)

087"  ID 0019" 1 (MATCH)

088"  ID 0019" 1 (MATCH)

089"  ID 0019" 1 (MATCH)

090"  ID 0019" 1 (MATCH)

091"  ID 0019" 1 (MATCH)

092"  ID 0019" 1 (MATCH)

093"  ID 0019" 1 (MATCH)

094"  ID 0019" 1 (MATCH)

095"  ID 0019" 1 (MATCH)

096"  ID 0019" 1 (MATCH)

097"  ID 0019" 1 (MATCH)

098"  ID 0019" 1 (MATCH)

099"  ID 0019" 1 (MATCH)

100"  ID 0019" 1 (MATCH)

101"  ID 0019" 1 (MATCH)

102"  ID 0019" 1 (MATCH)

103"  ID 0019" 1 (MATCH)

104"  ID 0019" 1 (MATCH)

105"  ID 0019" 1 (MATCH)

106"  ID 0019" 1 (MATCH)

107"  ID 0019" 1 (MATCH)

108"  ID 0019" 1 (MATCH)

109"  ID 0019" 1 (MATCH)

110"  ID 0019" 1 (MATCH)

111"  ID 0019" 1 (MATCH)

112"  ID 0019" 1 (MATCH)

113"  ID 0019" 1 (MATCH)

114"  ID 0019" 1 (MATCH)

115"  ID 0019" 1 (MATCH)

116"  ID 0019" 1 (MATCH)

117"  ID 0019" 1 (MATCH)

118"  ID 0019" 1 (MATCH)

119"  ID 0019" 1 (MATCH)

120"  ID 0019" 1 (MATCH)

121"  ID 0019" 1 (MATCH)

122"  ID 0019" 1 (MATCH)

123"  ID 0019" 1 (MATCH)

124"  ID 0019" 1 (MATCH)

125"  ID 0019" 1 (MATCH)

126"  ID 0019" 1 (MATCH)

127"  ID 0019" 1 (MATCH)

128"  ID 0019" 1 (MATCH)

129"  ID 0019" 1 (MATCH)

130"  ID 0019" 1 (MATCH)

131"  ID 0019" 1 (MATCH)

132"  ID 0019" 1 (MATCH)

133"  ID 0019" 1 (MATCH)

134"  ID 0019" 1 (MATCH)

135"  ID 0019" 1 (MATCH)

136"  ID 0019" 1 (MATCH)

137"  ID 0019" 1 (MATCH)

138"  ID 0019" 1 (MATCH)

139"  ID 0019" 1 (MATCH)

140"  ID 0019" 1 (MATCH)

141"  ID 0019" 1 (MATCH)

142"  ID 0019" 1 (MATCH)

143"  ID 0019" 1 (MATCH)

144"  ID 0019" 1 (MATCH)

145"  ID 0019" 1 (MATCH)

146"  ID 0019" 1 (MATCH)

147"  ID 0019" 1 (MATCH)

148"  ID 0019" 1 (MATCH)

149"  ID 0019" 1 (MATCH)

150"  ID 0019" 1 (MATCH)

151"  ID 0019" 1 (MATCH)

152"  ID 0019" 1 (MATCH)

153"  ID 0019" 1 (MATCH)

154"  ID 0019" 1 (MATCH)

155"  ID 0019" 1 (MATCH)

156"  ID 0019" 1 (MATCH)

157"  ID 0019" 1 (MATCH)

158"  ID 0019" 1 (MATCH)

159"  ID 0019" 1 (MATCH)

160"  ID 0019" 1 (MATCH)

161"  ID 0019" 1 (MATCH)

162"  ID 0019" 1 (MATCH)

163"  ID 0019" 1 (MATCH)

164"  ID 0019" 1 (MATCH)

165"  ID 0019" 1 (MATCH)

166"  ID 0019" 1 (MATCH)

167"  ID 0019" 1 (MATCH)

168"  ID 0019" 1 (MATCH)

169"  ID 0019" 1 (MATCH)

170"  ID 0019" 1 (MATCH)
SUBTL INTERNAL SUB-ROUTINE.

INPUT: VIDEO MEMORY IN A

OUTPUT: MEMORY SETTING IN B

RETURNS: REG STATUS IF MIFE INVALID

VALID VALUES: INPUT MIFE VIDEO

DISPLAY FILE ACTIVE AT SCREEN

DIAGNOSTIC ONLY 0 0 0

DIAGNOSTIC 80 8 0

DIAGNOSTIC 0 0 0

HIGH RES. 0 0 0

44/48-COLUMN 46 = 38 06 = 38 46 = 78

SETVAL LD C,A

RET B,A

JR NL,TST81

CPL R8

RET T,A

RET NE

LD R,A

RET Z

AND BCH

RBR G

RET NE

LD A,40H

CH C

LD C,A

TST81 RET

CP L

LD R,A

LD C,A

0118 C0 01

RET

0118 C6 01

RET

0118 C6 01

LD R,A

0117 C0 01

SFOH RDA A

0117 C6 01

RET

0117 C6 01

RET

0117 C6 01

RET

0117 C0 01

RET

0117 C6 01

RET

0117 C6 01

RET

0117 C6 01

RET

0117 C6 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C6 01

RET

0117 C6 01

RET

0117 C6 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET

0117 C0 01

RET
947 1 \text{return of}
948 1 \text{ops, in ml, preserves}
949 1 line/column position in bc
950 1
951 1 \text{calcs: call lnum}
952 1 \text{test which op}
953 1
954 1 \text{get display file ops.}
955 1 \text{in op2}
956 1
957 1 \text{col/s is position in display file}
958 1
959 1 \text{get display file address}
960 1 \text{for start of line in b}
961 1
962 1 \text{load cursor position}
963 1
964 1 \text{store cursor position}
965 1
966 1 \text{pixel patterns for std.graphics set}
967 1 *background(paper)* x=foreground(in)
968 1
969 1
970 1
971 1
972 1
973 1
974 1
975 1
976 1
977 1
978 1
979 1
980 1
981 1
982 1
983 1
984 1
985 1
986 1
987 1
988 1
989 1
990 1
991 1
992 1
993 1
994 1
995 1
996 1
997 1
998 1
999 1
1000 1
1001 1
1002 1
1003 1
1004 1 \text{get pixel}
1005 1 \text{defa 00000000h}
1006 1 \text{defa 00000000h}
1007 1 \text{defa 00000000h}
1008 1 \text{defa 00000000h}
1009 1 \text{defa 00000000h}
1010 1 \text{defa 00000000h}
1011 1 \text{defa 00000000h}
1012 1 \text{defa 00000000h}
1013 1 \text{defa 00000000h}
1014 1 \text{defa 00000000h}
1015 1 \text{defa 00000000h}
1016 1 \text{defa 00000000h}
1017 1 \text{defa 00000000h}
1018 1 \text{defa 00000000h}
1019 1 \text{defa 00000000h}
1020 1 \text{defa 00000000h}
1021 1
1022 1
1023 1
1024 1
1025 1
1026 1
1027 1
1028 1
1029 1
1030 1
1031 1
1032 1
1033 1
1034 1
1035 1
1036 1
1037 1
1038 1
1039 1
1040 1
1041 1
1042 1
1043 1
1044 1
1045 1
1046 1
1047 1
1048 1
1049 1
1050 1
173
APPENDIX C-2

80 COLUMN MODE

Date: 12/16/83
ASC Number: 002
Version: 001
Author: C. Corcoran/C. Boyle

Name: 80 Column Mode Support

Description:
This component provides support to the application program for using the 80-column mode features of the TI 99/4A. 80-Column Mode is implemented by using the 40-Column Mode feature of the 2060 and modifying the character width from 8 to 6 pixels. The services include opening/closing the second display file (used by the machine stack, OS RAM routines and BASIC structures), PRINT position control, attribute control, clear screen and scroll screen. Services are displayed on the screen. For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and non-zero values for other conditions. The interface from BASIC is by means of GDE'ing the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chmck 7 (8000H/807344).

Appication Services

Name: SETMODE (SETMODE from BASIC – parameter to VIMODE)

Input: MODE (normal, 40 or 80 column mode)
From machine code: Register 6
From BASIC: In VIMODE

Description:
Sets specified video mode, opening or closing the second display file where needed. This involves determining if there is enough memory to open the second display file and if so, moving the BASIC structures and machine language variables area up and the UDG area down to make space for the machine stack and OS RAM routines at the top of memory. The second display file is cleared to screen. The affected system and internal variables are updated or initialized (see Usage Section). When returning to mode 0 from 80 Column mode, the structures are returned to their normal locations.

Output: BC = 0 Successful
        BC = 1 Invalid parameters (not equal to 0 or 8)
        BC = 2 Not enough memory

Name: CLSCHR' (CLSCHR from BASIC – parameters to CLSCTL)

Input: Line Count (1-24)
        Starting Line Number (0-23)
From machine code: Line Count in Register B
                        Starting Line in Register C
From BASIC: Starting Line Number in CLSCTL
                        Line Count in CLSCTL + 1

Description:
Clears background color (PAPER) the designated number of lines, beginning with the Starting Line Number. Line 0 with a count of 24 clears the entire screen. Upon return, the cursor position is at the beginning of the first line cleared.

Output: BC = 0 for successful completion
        BC = 1 Invalid parameters
                        (Line Number + Line Count < 1 or >24)

177
**NAME:** SETCUR (SETCUR from BASIC - parameters to LINCCL)

**Input:** Line Number (0-23)
Column Number (0-79) or (0-84)

From Machine Code: Line Number In Register B
Column Number In Register C

From BASIC: Column Number In LINCCL
Line Number In LINCCL + 1

**Description:**
Converts the requested position to internal format, determines display file address, and stores the values for use by the next display character operation. Note that once established, the position is updated automatically when a character is displayed so that it is only necessary to set the position when sequential display is not desired.

**Output:**
- BC = 0 for successful completion
- BC = 1 for invalid parameters (Line Number > 23 ,
Column Number > Line Length-1)

**NAME:** SETATT (SETATB from BASIC - parameter to ATTCTL)

**Input:** Attribute Byte -
bit 7 - BRIGHT
bit 6 - BRIGHT
bit 5 - PER
bit 4 - PER
bit 3 - BRIGHT
bit 2 - PER
bit 1 - N
bit 0 - R

From Machine Code: Register A
From BASIC: In ATTCTL

**Description:**
The specified LUM color (0-7) is used to set the video mode hardware and to update VIMOD. The complementary PER color is fixed by the LUM selection. BRIGHT and PER are fixed at zero by the hardware. Note that in 80 column mode the entire screen has the same attributes.

**Output:**
- BC = 0 Successful

**NAME:** WCHR (WCHR from BASIC - parameter to DATAB)

**Input:** Character code for character to be displayed
- 20m TO TFM - Std. TEXT Character Set
- 80m TO TFM - Std. Graphics Set
- 80m TO TFM - User-Defined Graphics Set

From Machine Code: Register A
From BASIC: In DATAB

**Description:**
Displays character at current cursor position, applying current mask. Moves cursor position on to next sequential position. If character would start a new line after SOFL (see Usage section) and the scroll count (variable SCRCCL) decrements to zero, the character will not be displayed and return will be made with BC = 3 (screen full). If the scroll count does not decrement to zero, the screen will be scrolled up one line using the information in SCRCCL and the new line started at the vacated line.

Note that only the first 4 bits of each byte in the User Defined Graphics area will be transferred to the display file.

**Output:**
- BC = 0 for successful completion
- BC = 1 invalid character code
- BC = 3 for screen full

---

178
NAME: SETMSK (SETMSB from BASIC - parameters to MSCTL)

Input: Mask Byte - bit 0 - COVER
- bit 1 - INVERSE
From Machine Code: Register A
From BASIC: MSCTL

Description:
The specified mask is stored for application to all subsequent display character operations. COVER = 1 implies new character
combined with old using an XOR operation; INVERSE = 1 implies
character is inverted.
Output: BC = 0 for successful completion

NAME: WRSTRB (WRITRE from BASIC - String Identifier to PARAMS)

Input: Character Code String

Free machine code: Address of string in HL
Count in BC
From BASIC: String Variable Identifier in,
System Variable PARAM = 23747 (8CCH)

Description:
Displays the characters from the string, beginning at the current
cursor location and continuing sequentially until the count
expires, or "Screen Full" is detected (see WRCHD description and
Usage Section on Automatic Scrolling). For the Screen Full
condition the remaining count is stored in the internal variable
STRICT for access by the user.

NOTE: Characters within the string which are outside of the
supported range (32 through 127 or 20H through 4FH) will be
ignored. E.g., BASIC Token codes and control codes
embedded in an INPUT string will not be displayed or
decoded.

From BASIC, PUKD the code for the string variable identifier into
PARAMS prior to invoking WRSTRB, e.g.
0005 LET s%"---------------"
0010 PUKD 23747.CODE "s"
0015 IF USR (WRSTRB) THEN "-----"
(continues)

Output: BC = 0 Successful
BC = 2 BASIC - String not found
BC = 3 Screen Full - Remaining Count in STRICT
(HL = Current Address in String)

NAME: SCROLL (SCRLB from BASIC - parameters to SCRCTL)

Input: Line Count (1-23)
Starting Line Number (1-23)

Free Machine Code: Line Count In B
Starting Line In C
From BASIC: Starting Line In SCRCTL
Line Count In SCRCTL + 1

Description:
Scrolls the designated number of lines up 1 position, starting at
the specified line number and inserts a blank line at the bottom
of the scrolled area. Line 1 with a count of 23 will scroll the
entire screen up 1 line. Upon return, the cursor position is at
the beginning of the inserted blank line.

Note: See Usage Section on "automatic" scrolling.

Output: BC = 0 Successful
BC = 1 Invalid Parameters
(Line Number + Line Count < 1 or > 24)

Name: GTCHAR (GETCHAR from BASIC - parameters to GETCTL)

Input: Line/Column position as for SETCUR
From Machine Code: Line Number in B
Column Number in C
From BASIC: Column Number in GETCTL
Line Number in GETCTL + 1

Description:
Returns in register C of the BC register pair the character code
for the character at the designated screen position. If no match
against the character set (including the standard and
user-defined graphics) is found, zero is returned. (Character
code or zero also returned in B.)

Note: Positions "printed" using the DVER technique will return
zero if they do not match against any single character.

Output: BC = 0 for no find
BC = 1 invalid parameters
BC = character code (20h-44h)

Name: GETATT (GETATT from BASIC - parameters to GETCTL)

Input: As for GTCHAR

Description:
Returns in register C of the BC register pair the attribute byte
for the character at the designated screen position. Note that
in BCG column mode the entire screen has common attributes. The
value returned will describe the current selection.

Output: BC = 1 for invalid parameters
BC = attribute byte (as for SETATT)

Name: GETCUR (GETCUR from BASIC)

Input: None

Description:
Returns in the BC register pair and in the BASIC parameter
location LEOCL, the current print position (where the next
character would be displayed).

Output: B = Line number (0-15)
C = Column number (0-79) or (0-94)
BASIC: LEOCL = Column number
LEOCL + 1 = Line number

Note: If the last character was printed at Col.79 (94) of Line
15 (last position on the screen), then Col.0/Line 0
will be returned.

NAME:

Memory Usage:
This package of machine code routines includes the following
internal variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
</table>
| SCRCTL | 2    | Scroll Control
        |      | LSH = Starting line number |
        |      | MSB = Number of lines to be scrolled |
| BDTLN  | 1    | Bottom Line - Line number (0-23) after which
        |      | test for scroll will be made |
| SCRCLT | 1    | Scroll Count - Number of times = 1 that
        |      | automatic scroll will be done. When
        |      | decremented to zero, data
        |      | will not be displayed and a
        |      | condition code will be returned
        |      | to user |

180
Initial values set via SETMCD (SETMDB) are as follows:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCRCTL</td>
<td>1701M</td>
</tr>
<tr>
<td>BCTLN</td>
<td>17M</td>
</tr>
<tr>
<td>SCRCLT</td>
<td>1M</td>
</tr>
<tr>
<td>EMTL</td>
<td>(Internal to Module)</td>
</tr>
<tr>
<td>GRCTL</td>
<td>(Internal to Module)</td>
</tr>
<tr>
<td>LILINE</td>
<td>50M</td>
</tr>
<tr>
<td>CURPCS</td>
<td>1851M</td>
</tr>
<tr>
<td>DFADDR</td>
<td>4000M</td>
</tr>
<tr>
<td>MASKB</td>
<td>0M</td>
</tr>
<tr>
<td>ATTBYT</td>
<td>38M</td>
</tr>
<tr>
<td>GTINDEX</td>
<td>80M</td>
</tr>
<tr>
<td>STRGYT</td>
<td>0M</td>
</tr>
<tr>
<td>MARGIN</td>
<td>1M</td>
</tr>
<tr>
<td>DFBIT</td>
<td>7M</td>
</tr>
</tbody>
</table>

The following are the variables used for passing parameters in BASIC. The * indicates those initialized by SETMDB:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>*DATABASE</td>
<td>1</td>
<td>0M</td>
</tr>
<tr>
<td>*LINCCL</td>
<td>2</td>
<td>0M</td>
</tr>
<tr>
<td>*CLRCCL</td>
<td>2</td>
<td>1800M</td>
</tr>
<tr>
<td>*ATTBYT</td>
<td>1</td>
<td>38M</td>
</tr>
<tr>
<td>*MASKB</td>
<td>1</td>
<td>0M</td>
</tr>
<tr>
<td>*GETBYT</td>
<td>2</td>
<td>0M</td>
</tr>
<tr>
<td>*VIMODE</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

In addition, VIDMCD, PARAMS and other system variables must be available to these routines. At minimum, chunks 0-3 and chunk 7 must be enabled in the Home Bank.

Location:

This package can be incorporated into a machine code program within the memory range from chunk 4 through 7, taking into consideration the remapping of certain structures when the second display file is open. NOTE: Machine code above RAMTOP is not executed.

The routine SETMCD (SETMDB) cannot be executed from a cartridge because of the necessity to enable the ROM Extension which disables the Dock Bank.
Registers:

Other than as documented for output values, no claims are made as to preservation of any register contents except for the IV Register which must always contain the value $8384h$ for access to the standard system variables.

Automatic Scrolling:

As initialized, test for scrolling will be made when the print position goes to the next line following the bottom line on screen (DOTLN=25=4th line). Condition Code 3 (Screen Full) will be returned since SCRLT will decrement to zero. If SCRLT is set to some larger value, then the parameters in SCRLT will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a POKE or setting the variables DOTLN and SCRLT to the desired values, automatic scrolling can be done using smaller sections of the screen. When working from BASIC it is recommended that DOTLN be set to Line 20 (15h) and SCRLT be set to 20 (18h) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Not that once SCRLT expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the 'Screen Full' condition.

By setting the SCRLT variable and invoking the SCROLL (SCLRL) routine any portion of the screen may be scrolled at any time.

Margin Control:

In 80-Column Mode, there are actually 85 character positions per line. The variable MARGIN determines the offset of the beginning of the 80 column line from the left side of the screen and has valid offset values of 0, 1 or 2. An offset value of 1 centers the 80 column line on the screen! 0 begins at the extreme left side; 2 terminates at the extreme right side. The default value is 1. When MARGIN is set to 0, the variable LINLEN can be set to 85 to permit access to the 5 extra print positions. Whenever MARGIN and/or LINLEN are modified, a 'Set Cursor' operation should be done to ensure the integrity of the print position.

NOTE: Since the different MARGIN values result in different pixel positions for the columns, care must be taken in mixing line length and margin values on the same line.

ADDITIONAL NOTES:

1. All screen operations done by the system 8DH (PRINT, LIST, Edit line 1/2, CLS, scrolling, etc.) relate only to the main Display File and work with standard 8-pixel wide characters. This means that every alternate 8 pixels across the screen will be affected. You will want to execute the Clear Screen function in this module to guarantee that no data in the second display file interferes with the use of a system screen service, e.g., prior to doing a LIST.

2. The CPY command will print only every other grouping of 8 pixels across the screen to the Z80 Printer.

3. During tape operations, the border will not change while in 80-column mode since this is fixed by the software to conform to the paper used.

4. The SAVE filename SCREENS will save only the main Display File data. The second can be saved by a SAVE filename CODE 24516,8144. Be careful that you have the computer in 80-column mode when you load this data or you will overwrite the DS RAM routines and "break" the system! (The count for saving the Display file is for the data portion only since the Attribute File area from 78000-7FFFh (10070-11487) is not used by the video area HW in 80-Column mode.)
SUBTL "GET" ROUTINES

1. MORE FROM BASIC ENTRY TO GET CHARACTER
2. (RETURNS CODE FOR CHARACTER AT
3. POSITION SPECIFIED IN GYTCYL)
4. SET CONTROL AMB.
5. ENTRY WITH POSITION IN DC
6. TEST PARAMETERS
7. CALL YFPMAP
8. RETURN SET POSITION
9. CONVERT TO INTERNAL FORMAT
10. CALL CROW
11. SET UP ADDRESS AND DEF POS.
12. CALL CS, CODE
13. CALL OC,YTCYL
14. SET UP ADDRESS AND DEF POS.
15. CALL OC,YTCYL
16. SET UP ADDRESS AND DEF POS.
17. SET UP ADDRESS AND DEF POS.
18. SET UP ADDRESS AND DEF POS.
19. SET UP ADDRESS AND DEF POS.
20. SET UP ADDRESS AND DEF POS.
21. SET UP ADDRESS AND DEF POS.
22. SET UP ADDRESS AND DEF POS.
23. SET UP ADDRESS AND DEF POS.
24. SET UP ADDRESS AND DEF POS.
25. SET UP ADDRESS AND DEF POS.
26. SET UP ADDRESS AND DEF POS.
27. SET UP ADDRESS AND DEF POS.
28. SET UP ADDRESS AND DEF POS.
29. SET UP ADDRESS AND DEF POS.
30. SET UP ADDRESS AND DEF POS.
31. SET UP ADDRESS AND DEF POS.
32. SET UP ADDRESS AND DEF POS.
33. SET UP ADDRESS AND DEF POS.
34. SET UP ADDRESS AND DEF POS.
35. SET UP ADDRESS AND DEF POS.
36. SET UP ADDRESS AND DEF POS.
37. SET UP ADDRESS AND DEF POS.
38. SET UP ADDRESS AND DEF POS.
39. SET UP ADDRESS AND DEF POS.
40. SET UP ADDRESS AND DEF POS.
41. SET UP ADDRESS AND DEF POS.
42. SET UP ADDRESS AND DEF POS.
43. SET UP ADDRESS AND DEF POS.
44. SET UP ADDRESS AND DEF POS.
45. SET UP ADDRESS AND DEF POS.
46. SET UP ADDRESS AND DEF POS.
47. SET UP ADDRESS AND DEF POS.
48. SET UP ADDRESS AND DEF POS.
49. SET UP ADDRESS AND DEF POS.
50. SET UP ADDRESS AND DEF POS.
51. SET UP ADDRESS AND DEF POS.
52. SET UP ADDRESS AND DEF POS.
53. SET UP ADDRESS AND DEF POS.
54. SET UP ADDRESS AND DEF POS.
55. SET UP ADDRESS AND DEF POS.
56. SET UP ADDRESS AND DEF POS.
57. SET UP ADDRESS AND DEF POS.
58. SET UP ADDRESS AND DEF POS.
59. SET UP ADDRESS AND DEF POS.
60. SET UP ADDRESS AND DEF POS.
61. SET UP ADDRESS AND DEF POS.
62. SET UP ADDRESS AND DEF POS.
63. SET UP ADDRESS AND DEF POS.
64. SET UP ADDRESS AND DEF POS.
65. SET UP ADDRESS AND DEF POS.
66. SET UP ADDRESS AND DEF POS.
67. SET UP ADDRESS AND DEF POS.
68. SET UP ADDRESS AND DEF POS.
69. SET UP ADDRESS AND DEF POS.
70. SET UP ADDRESS AND DEF POS.
71. SET UP ADDRESS AND DEF POS.
72. SET UP ADDRESS AND DEF POS.
73. SET UP ADDRESS AND DEF POS.
74. SET UP ADDRESS AND DEF POS.
75. SET UP ADDRESS AND DEF POS.
76. SET UP ADDRESS AND DEF POS.
77. SET UP ADDRESS AND DEF POS.
78. SET UP ADDRESS AND DEF POS.
79. SET UP ADDRESS AND DEF POS.
80. SET UP ADDRESS AND DEF POS.
81. SET UP ADDRESS AND DEF POS.
82. SET UP ADDRESS AND DEF POS.
83. SET UP ADDRESS AND DEF POS.
84. SET UP ADDRESS AND DEF POS.
85. SET UP ADDRESS AND DEF POS.
86. SET UP ADDRESS AND DEF POS.
87. SET UP ADDRESS AND DEF POS.
88. SET UP ADDRESS AND DEF POS.
89. SET UP ADDRESS AND DEF POS.
90. SET UP ADDRESS AND DEF POS.
91. SET UP ADDRESS AND DEF POS.
92. SET UP ADDRESS AND DEF POS.
93. SET UP ADDRESS AND DEF POS.
94. SET UP ADDRESS AND DEF POS.
95. SET UP ADDRESS AND DEF POS.
96. SET UP ADDRESS AND DEF POS.
97. SET UP ADDRESS AND DEF POS.
98. SET UP ADDRESS AND DEF POS.
99. SET UP ADDRESS AND DEF POS.
100. SET UP ADDRESS AND DEF POS.
101. SET UP ADDRESS AND DEF POS.
102. SET UP ADDRESS AND DEF POS.
103. SET UP ADDRESS AND DEF POS.
104. SET UP ADDRESS AND DEF POS.
105. SET UP ADDRESS AND DEF POS.
106. SET UP ADDRESS AND DEF POS.
107. SET UP ADDRESS AND DEF POS.
108. SET UP ADDRESS AND DEF POS.
109. SET UP ADDRESS AND DEF POS.
110. SET UP ADDRESS AND DEF POS.
111. SET UP ADDRESS AND DEF POS.
112. SET UP ADDRESS AND DEF POS.
113. SET UP ADDRESS AND DEF POS.
114. SET UP ADDRESS AND DEF POS.
115. SET UP ADDRESS AND DEF POS.
116. SET UP ADDRESS AND DEF POS.
117. SET UP ADDRESS AND DEF POS.
118. SET UP ADDRESS AND DEF POS.
119. SET UP ADDRESS AND DEF POS.
120. SET UP ADDRESS AND DEF POS.
121. SET UP ADDRESS AND DEF POS.
122. SET UP ADDRESS AND DEF POS.
123. SET UP ADDRESS AND DEF POS.
124. SET UP ADDRESS AND DEF POS.
125. SET UP ADDRESS AND DEF POS.
126. SET UP ADDRESS AND DEF POS.
127. SET UP ADDRESS AND DEF POS.
128. SET UP ADDRESS AND DEF POS.
129. SET UP ADDRESS AND DEF POS.
130. SET UP ADDRESS AND DEF POS.
131. SET UP ADDRESS AND DEF POS.
132. SET UP ADDRESS AND DEF POS.
133. SET UP ADDRESS AND DEF POS.
134. SET UP ADDRESS AND DEF POS.
135. SET UP ADDRESS AND DEF POS.
136. SET UP ADDRESS AND DEF POS.
137. SET UP ADDRESS AND DEF POS.
138. SET UP ADDRESS AND DEF POS.
139. SET UP ADDRESS AND DEF POS.
140. SET UP ADDRESS AND DEF POS.
141. SET UP ADDRESS AND DEF POS.
142. SET UP ADDRESS AND DEF POS.
143. SET UP ADDRESS AND DEF POS.
144. SET UP ADDRESS AND DEF POS.
145. SET UP ADDRESS AND DEF POS.
146. SET UP ADDRESS AND DEF POS.
147. SET UP ADDRESS AND DEF POS.
148. SET UP ADDRESS AND DEF POS.
149. SET UP ADDRESS AND DEF POS.
150. SET UP ADDRESS AND DEF POS.
151. SET UP ADDRESS AND DEF POS.
152. SET UP ADDRESS AND DEF POS.
153. SET UP ADDRESS AND DEF POS.
154. SET UP ADDRESS AND DEF POS.
155. SET UP ADDRESS AND DEF POS.
156. SET UP ADDRESS AND DEF POS.
157. SET UP ADDRESS AND DEF POS.
158. SET UP ADDRESS AND DEF POS.
159. SET UP ADDRESS AND DEF POS.
160. SET UP ADDRESS AND DEF POS.
161. SET UP ADDRESS AND DEF POS.
162. SET UP ADDRESS AND DEF POS.
163. SET UP ADDRESS AND DEF POS.
164. SET UP ADDRESS AND DEF POS.
165. SET UP ADDRESS AND DEF POS.
166. SET UP ADDRESS AND DEF POS.
167. SET UP ADDRESS AND DEF POS.
168. SET UP ADDRESS AND DEF POS.
169. SET UP ADDRESS AND DEF POS.
170. SET UP ADDRESS AND DEF POS.
171. SET UP ADDRESS AND DEF POS.
172. SET UP ADDRESS AND DEF POS.
173. SET UP ADDRESS AND DEF POS.
174. SET UP ADDRESS AND DEF POS.
175. SET UP ADDRESS AND DEF POS.
176. SET UP ADDRESS AND DEF POS.
177. SET UP ADDRESS AND DEF POS.
178. SET UP ADDRESS AND DEF POS.
179. SET UP ADDRESS AND DEF POS.
180. SET UP ADDRESS AND DEF POS.
181. SET UP ADDRESS AND DEF POS.
182. SET UP ADDRESS AND DEF POS.
183. SET UP ADDRESS AND DEF POS.
184. SET UP ADDRESS AND DEF POS.
185. SET UP ADDRESS AND DEF POS.
186. SET UP ADDRESS AND DEF POS.
187. SET UP ADDRESS AND DEF POS.
188. SET UP ADDRESS AND DEF POS.
OC6E 78 0D3E 1 T9L 010 07CH 0F POP AP 1 RESTORE ORIG. OP-AT
OC6E 79 0D4F 2 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 7A 0D49 3 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 7B 0D39 4 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 7C 0D29 5 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 7D 0D19 6 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 7E 0D09 7 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 7F 0D3E 8 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 80 0D49 9 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 81 0D39 A T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 82 0D29 B T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 83 0D19 C T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 84 0D3E D T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 85 0D49 E T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 86 0D39 F T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 87 0D29 0 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 88 0D19 1 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 89 0D3E 2 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 8A 0D49 3 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 8B 0D39 4 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 8C 0D29 5 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 8D 0D19 6 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 8E 0D3E 7 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 8F 0D49 8 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 90 0D39 9 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 91 0D29 A T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 92 0D19 B T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 93 0D3E C T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 94 0D49 D T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 95 0D39 E T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 96 0D29 F T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 97 0D19 0 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 98 0D3E 1 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 99 0D49 2 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 9A 0D39 3 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 9B 0D29 4 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 9C 0D19 5 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 9D 0D3E 6 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 9E 0D49 7 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6E 9F 0D39 8 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 00 0D49 9 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 01 0D39 A T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 02 0D29 B T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 03 0D19 C T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 04 0D3E D T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 05 0D49 E T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 06 0D39 F T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 07 0D29 0 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
OC6F 08 0D19 1 T9L 010 010 C0 0F DPOP IT 1 RESTORE ORIG.
INCLUDE CHARSET
SUBTL CHAR,SET FOR 48/80 COL./MOUSE

ENTRY WITH SCEXT/CUR+IN
INTERNAL FORMAT
CALL CALCPFS
STORE UPDATED POSITION AND RETURN
ROUTINE TO CALCULATE POSITION IN DP
UPDATES DET POSITION IN VARIABLE DPST
RETURN OF ADDRESS IN ML, PRESERVES
LINE/COL.POSITION IN DC

GET DISPLAY FILE ADDRESSES
TEST WHICH DP
SAVE COL/NO.DC/LEN=113
300 COL/POS.+406-64=COL/POS.
000 COL/RES. IN DP2
IN DP2
CLEAR CARRY
COL/POS. IS POSITION IN DISPLAY FILE
GET ADJUSTMENT VALUE
DATA CONTAINS OFFSET INTO LINE
ML CONTAINS DP ADDRESS
6+COL/POS. FOR 80 COLS,
STARTS AT BIT 7
STARTS AT BIT 1/3 OR 3
CURSES ML/OF ADDRESS
GET DISPLAY FILE ADDRESS
FOR START OF LINE IN 6

LO A,8,SP
SUB A
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
AND 6,
AND 8,
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Code 20</td>
</tr>
<tr>
<td>0001</td>
<td>Code 21</td>
</tr>
<tr>
<td>0002</td>
<td>Code 22</td>
</tr>
<tr>
<td>0003</td>
<td>Code 23</td>
</tr>
<tr>
<td>0004</td>
<td>Code 24</td>
</tr>
<tr>
<td>0005</td>
<td>Code 25</td>
</tr>
<tr>
<td>0006</td>
<td>Code 26</td>
</tr>
<tr>
<td>0007</td>
<td>Code 27</td>
</tr>
<tr>
<td>0008</td>
<td>Code 28</td>
</tr>
<tr>
<td>0009</td>
<td>Code 29</td>
</tr>
<tr>
<td>000A</td>
<td>Code 30</td>
</tr>
<tr>
<td>000B</td>
<td>Code 31</td>
</tr>
<tr>
<td>000C</td>
<td>Code 32</td>
</tr>
<tr>
<td>000D</td>
<td>Code 33</td>
</tr>
<tr>
<td>000E</td>
<td>Code 34</td>
</tr>
<tr>
<td>000F</td>
<td>Code 35</td>
</tr>
</tbody>
</table>

*Note: The above codes are for a specific system or hardware setup.*
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1237</td>
<td>code 20</td>
<td>0</td>
</tr>
<tr>
<td>1239</td>
<td>defb 01111000</td>
<td></td>
</tr>
<tr>
<td>1239</td>
<td>defb 00010100</td>
<td></td>
</tr>
<tr>
<td>1240</td>
<td>defb 00101000</td>
<td></td>
</tr>
<tr>
<td>1240</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1241</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1241</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1242</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1243</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1244</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1245</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1246</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1247</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1248</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1249</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1250</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1251</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1252</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1253</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1254</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1255</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1256</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1257</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1258</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1259</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1260</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1261</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1262</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1263</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1264</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1265</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1266</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1267</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1268</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1269</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1270</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1271</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1272</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1273</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1274</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1275</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1276</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1277</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1278</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1279</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1280</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1281</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1282</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1283</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1284</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1285</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1286</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1287</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1288</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1289</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1290</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1291</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1292</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1293</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1294</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1295</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1296</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1297</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1298</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1299</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1300</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1301</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1302</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1303</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1304</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1305</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1306</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1307</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1308</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1309</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1310</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1311</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1312</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1313</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1314</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1315</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1316</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1317</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1318</td>
<td>defb 00000000</td>
<td></td>
</tr>
<tr>
<td>1319</td>
<td>defb 00000000</td>
<td></td>
</tr>
</tbody>
</table>

195
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0716</td>
<td>00</td>
</tr>
<tr>
<td>071C</td>
<td>00</td>
</tr>
<tr>
<td>071D</td>
<td>PC</td>
</tr>
<tr>
<td>071E</td>
<td>00</td>
</tr>
<tr>
<td>071F</td>
<td>00</td>
</tr>
<tr>
<td>0720</td>
<td>00</td>
</tr>
<tr>
<td>0721</td>
<td>00</td>
</tr>
<tr>
<td>0722</td>
<td>00</td>
</tr>
<tr>
<td>0723</td>
<td>00</td>
</tr>
<tr>
<td>0724</td>
<td>00</td>
</tr>
<tr>
<td>0725</td>
<td>00</td>
</tr>
<tr>
<td>0726</td>
<td>00</td>
</tr>
<tr>
<td>0727</td>
<td>00</td>
</tr>
<tr>
<td>0728</td>
<td>00</td>
</tr>
<tr>
<td>0729</td>
<td>00</td>
</tr>
<tr>
<td>072A</td>
<td>00</td>
</tr>
<tr>
<td>072B</td>
<td>00</td>
</tr>
<tr>
<td>072C</td>
<td>00</td>
</tr>
<tr>
<td>072D</td>
<td>00</td>
</tr>
<tr>
<td>072E</td>
<td>00</td>
</tr>
<tr>
<td>072F</td>
<td>00</td>
</tr>
<tr>
<td>0730</td>
<td>00</td>
</tr>
<tr>
<td>0731</td>
<td>00</td>
</tr>
<tr>
<td>0732</td>
<td>00</td>
</tr>
<tr>
<td>0733</td>
<td>00</td>
</tr>
<tr>
<td>0734</td>
<td>00</td>
</tr>
<tr>
<td>0735</td>
<td>00</td>
</tr>
<tr>
<td>0736</td>
<td>00</td>
</tr>
<tr>
<td>0737</td>
<td>00</td>
</tr>
<tr>
<td>0738</td>
<td>00</td>
</tr>
<tr>
<td>0739</td>
<td>00</td>
</tr>
<tr>
<td>073A</td>
<td>00</td>
</tr>
<tr>
<td>073B</td>
<td>00</td>
</tr>
<tr>
<td>073C</td>
<td>00</td>
</tr>
<tr>
<td>073D</td>
<td>00</td>
</tr>
<tr>
<td>073E</td>
<td>00</td>
</tr>
<tr>
<td>073F</td>
<td>00</td>
</tr>
<tr>
<td>0740</td>
<td>00</td>
</tr>
<tr>
<td>0741</td>
<td>00</td>
</tr>
<tr>
<td>0742</td>
<td>00</td>
</tr>
<tr>
<td>0743</td>
<td>00</td>
</tr>
<tr>
<td>0744</td>
<td>00</td>
</tr>
<tr>
<td>0745</td>
<td>00</td>
</tr>
<tr>
<td>0746</td>
<td>00</td>
</tr>
<tr>
<td>0747</td>
<td>00</td>
</tr>
<tr>
<td>0748</td>
<td>00</td>
</tr>
<tr>
<td>0749</td>
<td>00</td>
</tr>
<tr>
<td>074A</td>
<td>00</td>
</tr>
<tr>
<td>074B</td>
<td>00</td>
</tr>
<tr>
<td>074C</td>
<td>00</td>
</tr>
<tr>
<td>074D</td>
<td>00</td>
</tr>
<tr>
<td>074E</td>
<td>00</td>
</tr>
<tr>
<td>074F</td>
<td>00</td>
</tr>
<tr>
<td>0750</td>
<td>00</td>
</tr>
<tr>
<td>0751</td>
<td>00</td>
</tr>
<tr>
<td>0752</td>
<td>00</td>
</tr>
<tr>
<td>0753</td>
<td>00</td>
</tr>
<tr>
<td>0754</td>
<td>00</td>
</tr>
<tr>
<td>0755</td>
<td>00</td>
</tr>
<tr>
<td>0756</td>
<td>00</td>
</tr>
<tr>
<td>0757</td>
<td>00</td>
</tr>
<tr>
<td>0758</td>
<td>00</td>
</tr>
<tr>
<td>0759</td>
<td>00</td>
</tr>
<tr>
<td>075A</td>
<td>00</td>
</tr>
<tr>
<td>075B</td>
<td>00</td>
</tr>
<tr>
<td>075C</td>
<td>00</td>
</tr>
<tr>
<td>075D</td>
<td>00</td>
</tr>
<tr>
<td>075E</td>
<td>00</td>
</tr>
<tr>
<td>075F</td>
<td>00</td>
</tr>
<tr>
<td>0760</td>
<td>00</td>
</tr>
<tr>
<td>0761</td>
<td>00</td>
</tr>
<tr>
<td>0762</td>
<td>00</td>
</tr>
<tr>
<td>0763</td>
<td>00</td>
</tr>
<tr>
<td>0764</td>
<td>00</td>
</tr>
<tr>
<td>0765</td>
<td>00</td>
</tr>
<tr>
<td>0766</td>
<td>00</td>
</tr>
<tr>
<td>0767</td>
<td>00</td>
</tr>
<tr>
<td>0768</td>
<td>00</td>
</tr>
<tr>
<td>0769</td>
<td>00</td>
</tr>
<tr>
<td>076A</td>
<td>00</td>
</tr>
<tr>
<td>076B</td>
<td>00</td>
</tr>
<tr>
<td>076C</td>
<td>00</td>
</tr>
<tr>
<td>076D</td>
<td>00</td>
</tr>
<tr>
<td>076E</td>
<td>00</td>
</tr>
<tr>
<td>076F</td>
<td>00</td>
</tr>
<tr>
<td>0770</td>
<td>00</td>
</tr>
<tr>
<td>0771</td>
<td>00</td>
</tr>
<tr>
<td>0772</td>
<td>00</td>
</tr>
<tr>
<td>0773</td>
<td>00</td>
</tr>
<tr>
<td>0774</td>
<td>00</td>
</tr>
<tr>
<td>0775</td>
<td>00</td>
</tr>
<tr>
<td>0776</td>
<td>00</td>
</tr>
<tr>
<td>0777</td>
<td>00</td>
</tr>
<tr>
<td>0778</td>
<td>00</td>
</tr>
<tr>
<td>0779</td>
<td>00</td>
</tr>
<tr>
<td>077A</td>
<td>00</td>
</tr>
<tr>
<td>077B</td>
<td>00</td>
</tr>
<tr>
<td>077C</td>
<td>00</td>
</tr>
<tr>
<td>077D</td>
<td>00</td>
</tr>
<tr>
<td>077E</td>
<td>00</td>
</tr>
<tr>
<td>077F</td>
<td>00</td>
</tr>
<tr>
<td>0780</td>
<td>00</td>
</tr>
<tr>
<td>0781</td>
<td>00</td>
</tr>
<tr>
<td>0782</td>
<td>00</td>
</tr>
<tr>
<td>0783</td>
<td>00</td>
</tr>
<tr>
<td>0784</td>
<td>00</td>
</tr>
<tr>
<td>0785</td>
<td>00</td>
</tr>
<tr>
<td>0786</td>
<td>00</td>
</tr>
<tr>
<td>0787</td>
<td>00</td>
</tr>
<tr>
<td>0788</td>
<td>00</td>
</tr>
<tr>
<td>0789</td>
<td>00</td>
</tr>
<tr>
<td>078A</td>
<td>00</td>
</tr>
<tr>
<td>078B</td>
<td>00</td>
</tr>
<tr>
<td>078C</td>
<td>00</td>
</tr>
<tr>
<td>078D</td>
<td>00</td>
</tr>
<tr>
<td>078E</td>
<td>00</td>
</tr>
<tr>
<td>078F</td>
<td>00</td>
</tr>
<tr>
<td>0790</td>
<td>00</td>
</tr>
<tr>
<td>0791</td>
<td>00</td>
</tr>
<tr>
<td>0792</td>
<td>00</td>
</tr>
<tr>
<td>0793</td>
<td>00</td>
</tr>
<tr>
<td>0794</td>
<td>00</td>
</tr>
<tr>
<td>0795</td>
<td>00</td>
</tr>
<tr>
<td>0796</td>
<td>00</td>
</tr>
<tr>
<td>0797</td>
<td>00</td>
</tr>
<tr>
<td>0798</td>
<td>00</td>
</tr>
<tr>
<td>0799</td>
<td>00</td>
</tr>
<tr>
<td>079A</td>
<td>00</td>
</tr>
<tr>
<td>079B</td>
<td>00</td>
</tr>
<tr>
<td>079C</td>
<td>00</td>
</tr>
<tr>
<td>079D</td>
<td>00</td>
</tr>
<tr>
<td>079E</td>
<td>00</td>
</tr>
<tr>
<td>079F</td>
<td>00</td>
</tr>
<tr>
<td>07A0</td>
<td>00</td>
</tr>
<tr>
<td>07A1</td>
<td>00</td>
</tr>
<tr>
<td>07A2</td>
<td>00</td>
</tr>
<tr>
<td>07A3</td>
<td>00</td>
</tr>
<tr>
<td>07A4</td>
<td>00</td>
</tr>
<tr>
<td>07A5</td>
<td>00</td>
</tr>
<tr>
<td>07A6</td>
<td>00</td>
</tr>
<tr>
<td>07A7</td>
<td>00</td>
</tr>
<tr>
<td>07A8</td>
<td>00</td>
</tr>
<tr>
<td>07A9</td>
<td>00</td>
</tr>
<tr>
<td>07AA</td>
<td>00</td>
</tr>
<tr>
<td>07AB</td>
<td>00</td>
</tr>
<tr>
<td>07AC</td>
<td>00</td>
</tr>
<tr>
<td>07AD</td>
<td>00</td>
</tr>
<tr>
<td>07AE</td>
<td>00</td>
</tr>
<tr>
<td>07AF</td>
<td>00</td>
</tr>
<tr>
<td>07B0</td>
<td>00</td>
</tr>
<tr>
<td>07B1</td>
<td>00</td>
</tr>
<tr>
<td>07B2</td>
<td>00</td>
</tr>
<tr>
<td>07B3</td>
<td>00</td>
</tr>
<tr>
<td>07B4</td>
<td>00</td>
</tr>
<tr>
<td>07B5</td>
<td>00</td>
</tr>
<tr>
<td>07B6</td>
<td>00</td>
</tr>
<tr>
<td>07B7</td>
<td>00</td>
</tr>
<tr>
<td>07B8</td>
<td>00</td>
</tr>
<tr>
<td>07B9</td>
<td>00</td>
</tr>
<tr>
<td>07BA</td>
<td>00</td>
</tr>
<tr>
<td>07BB</td>
<td>00</td>
</tr>
<tr>
<td>07BC</td>
<td>00</td>
</tr>
<tr>
<td>07BD</td>
<td>00</td>
</tr>
<tr>
<td>07BE</td>
<td>00</td>
</tr>
<tr>
<td>07BF</td>
<td>00</td>
</tr>
<tr>
<td>07C0</td>
<td>00</td>
</tr>
<tr>
<td>07C1</td>
<td>00</td>
</tr>
<tr>
<td>07C2</td>
<td>00</td>
</tr>
<tr>
<td>07C3</td>
<td>00</td>
</tr>
<tr>
<td>07C4</td>
<td>00</td>
</tr>
<tr>
<td>07C5</td>
<td>00</td>
</tr>
<tr>
<td>07C6</td>
<td>00</td>
</tr>
<tr>
<td>07C7</td>
<td>00</td>
</tr>
<tr>
<td>07C8</td>
<td>00</td>
</tr>
<tr>
<td>07C9</td>
<td>00</td>
</tr>
<tr>
<td>07CA</td>
<td>00</td>
</tr>
<tr>
<td>07CB</td>
<td>00</td>
</tr>
<tr>
<td>07CC</td>
<td>00</td>
</tr>
<tr>
<td>07CD</td>
<td>00</td>
</tr>
<tr>
<td>07CE</td>
<td>00</td>
</tr>
<tr>
<td>07CF</td>
<td>00</td>
</tr>
<tr>
<td>07D0</td>
<td>00</td>
</tr>
<tr>
<td>07D1</td>
<td>00</td>
</tr>
<tr>
<td>07D2</td>
<td>00</td>
</tr>
<tr>
<td>07D3</td>
<td>00</td>
</tr>
<tr>
<td>07D4</td>
<td>00</td>
</tr>
<tr>
<td>07D5</td>
<td>00</td>
</tr>
<tr>
<td>07D6</td>
<td>00</td>
</tr>
<tr>
<td>07D7</td>
<td>00</td>
</tr>
<tr>
<td>07D8</td>
<td>00</td>
</tr>
<tr>
<td>07D9</td>
<td>00</td>
</tr>
<tr>
<td>07DA</td>
<td>00</td>
</tr>
<tr>
<td>07DB</td>
<td>00</td>
</tr>
<tr>
<td>07DC</td>
<td>00</td>
</tr>
<tr>
<td>07DD</td>
<td>00</td>
</tr>
<tr>
<td>07DE</td>
<td>00</td>
</tr>
<tr>
<td>07DF</td>
<td>00</td>
</tr>
<tr>
<td>07E0</td>
<td>00</td>
</tr>
<tr>
<td>07E1</td>
<td>00</td>
</tr>
<tr>
<td>07E2</td>
<td>00</td>
</tr>
<tr>
<td>07E3</td>
<td>00</td>
</tr>
<tr>
<td>07E4</td>
<td>00</td>
</tr>
<tr>
<td>07E5</td>
<td>00</td>
</tr>
<tr>
<td>07E6</td>
<td>00</td>
</tr>
<tr>
<td>07E7</td>
<td>00</td>
</tr>
<tr>
<td>07E8</td>
<td>00</td>
</tr>
<tr>
<td>07E9</td>
<td>00</td>
</tr>
<tr>
<td>07EA</td>
<td>00</td>
</tr>
<tr>
<td>07EB</td>
<td>00</td>
</tr>
<tr>
<td>07EC</td>
<td>00</td>
</tr>
<tr>
<td>07ED</td>
<td>00</td>
</tr>
<tr>
<td>07EE</td>
<td>00</td>
</tr>
<tr>
<td>07EF</td>
<td>00</td>
</tr>
<tr>
<td>07F0</td>
<td>00</td>
</tr>
<tr>
<td>07F1</td>
<td>00</td>
</tr>
<tr>
<td>07F2</td>
<td>00</td>
</tr>
<tr>
<td>07F3</td>
<td>00</td>
</tr>
<tr>
<td>07F4</td>
<td>00</td>
</tr>
<tr>
<td>07F5</td>
<td>00</td>
</tr>
<tr>
<td>07F6</td>
<td>00</td>
</tr>
<tr>
<td>07F7</td>
<td>00</td>
</tr>
<tr>
<td>07F8</td>
<td>00</td>
</tr>
<tr>
<td>07F9</td>
<td>00</td>
</tr>
<tr>
<td>07FA</td>
<td>00</td>
</tr>
<tr>
<td>07FB</td>
<td>00</td>
</tr>
<tr>
<td>07FC</td>
<td>00</td>
</tr>
<tr>
<td>07FD</td>
<td>00</td>
</tr>
<tr>
<td>07FE</td>
<td>00</td>
</tr>
<tr>
<td>07FF</td>
<td>00</td>
</tr>
</tbody>
</table>

*1 code 60  pound sign*
APPENDIX C-3

40 COLUMN MODE

Name: 40 Column Mode Support

Description:

This component provides support for the application programmer for using the 40-column mode feature of the TI 2048. 40-column mode is implemented by modifying the character width from 8 to 6 pixels. The services include position control, clear screen and scroll screen services and display of characters. For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and designated non-zero values for other conditions. The interface from BASIC is by means of DEFINE'ing the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chunk 7 (66000H/87344).

Attributes and other display controls such as Inverse are taken from the standard TI2048 System Variables (see Usage Section.)

This component is designed to permit use in normal video mode (Display File 1 only) or, in conjunction with ASC046 - Dual Screen Mode Support, to permit use of Display File 1 and/or Display File 2. The value of the System Variable VICMODE is used to determine which display file is the target of the requested service.

APPLICATION SERVICES

Name: INIT40 (INIT48 from BASIC)

Input: None

Description:

Initializes the internal variables to their default values for 40-column mode (see Usage Section).
Name: CLSCH (CLSCHR from BASIC - parameters to CLSCTL)

Input: Line Count (1-24)
Starting Line Number (0-23)

From Machine Code: Line Count In Register B
Starting Line In Register C

From BASIC: Starting Line Number in CLSCTL
Line Count in CLSCTL + 1

Description:
Clears to background color (PAPER) the designated number of lines, beginning with the Starting Line Number. Line 0 with a count of 24 clears the entire screen. Upon return, the cursor position is at the beginning of the first line cleared.

Output: BC = 0 for successful completion
BC = 1 for invalid parameters
(Line Number = Line Count < 1 or > 24)

Name: SETCUB (SETCUB from BASIC - parameters to LINCOL)

Input: Line Number (0-23)
Column Number (0-39) or (0-61)

From Machine Code: Line Number In Register B
Column Number In Register C

From BASIC: Column Number in LINCOL
Line Number in LINCOL + 1

Description:
Converts the requested position to internal format, determines display file address and stores the values for use by the next display character operation. Note that once established, the position is updated automatically when a character is displayed so that it is only necessary to set the position when sequential display is not desired.

Output: BC = 0 for successful completion
BC = 1 for invalid parameters (Line Number > 23 ; Column Number > Line Length-1)

Name: WRCBH (WRCBH from BASIC - parameter to DTDAB)

Input: Character code for character to be displayed
20h TO 7Fh - Std. T52088 Character Set
80h TO BFh - Std. Graphics Set
90h TO FFh - User-Defined Graphics Set

From Machine Code: Register A

From BASIC: in DTDAB

Description:
Displays character at current cursor position, applying current attributes and mask. Moves cursor position on to next sequential position. If character would start a new line after BOTLM (see Usage section) and the scroll count (variable SCRLCT) decrements to zero, the character will not be displayed and return will be made with BC = 3 (screen full). If the scroll count does not decrement to zero, the screen will be scrolled up one line using the information in SCRLCTL and the new line started at the vacated lines.

Note that only the first 6 bits of each byte in the User Defined Graphics area will be transferred to the display file.

Output: BC = 0 for successful completion
BC = 1 for invalid character code
BC = 3 for screen full

206
NAME: WSTRBC (WSTRBC from BASIC - String Identifier in PARAMS)

Input: Character Code String

From machine code: Address of string in RL
Count in BC

From BASIC: String Variable Identifier in
System Variable PARAMS = 23747 (SEE CM)

Description:

Displays the characters from the string, beginning at the current
cursor location and continuing sequentially until the count
expires, or "Screen Full" is detected (see WRCMD description and
Usage Section on Automatic Scrolling). For the Screen Full
condition, the remaining count is stored in the internal variable
STRICT for access by the user.

NOTE: Characters within the string which are outside of the
supported range (81 through local 120=64H) will be
ignored. E.g., BASIC Token codes and control codes
embedded in an INPUT string will not be displayed or
decoded.

From BASIC, PEEK the code for the string variable identifier into
PARAMS prior to invoking WSTRBC, e.g.

0000 LET a=string
0010 PEEK 23747;G001=0
0015 IF USR (WSTRBC)>0 THEN -----
(Continues)

Output:
BC = 0 Successful
BC = 2 BASIC - String not found
BC = 3 Screen Full - Remaining Count in STRICT
(ML=Current Address in String)

NAME: SCRL (SCRLB from BASIC - parameters to SCRCTL)

Input: Line Count (1-23)
Starting Line Number (1-23)

From machine code: Line Count in B
Starting Line in C
From BASIC: Starting Line in SCRCTL
Line Count in SCRCTL + 1

Description:

Scrolls the designated number of lines up 1 position, starting at
the specified line number and inserts a blank line at the bottom
of the scrolled area. Line 1 with a count of 23 will scroll the
entire screen up 1 line. Upon return, the cursor position is at
the beginning of the inserted blank line.

Note: See Usage Section on "automatic" scrolling.

Output: BC = 0 Successful
BC = 1 Invalid Parameters
(0 < Line Number < 24)

207
Name: GTCCHAR (GTCCHAR from BASIC - parameters to GTCCTL)

Input: Line/Column position as for SETCUR

Description:
Returns in register C of the BC register pair the character code for the character at the designated screen position. If no match against the character set (including the standard and user-defined graphics) is found, zero is returned. Character code or zero also returned in A.

Note: Positions "printed" using the DVER technique will return zero if they do not match against any single character.

Output: BC = 0 for no data
        BC = 1 invalid parameters
        BC = character code (00-6FH)

Name: GTCATT (GTCATT from BASIC - parameters to GTCCTL)

Input: As for GTCCHAR

Description:
Returns in register C of the BC register pair the attribute byte for the character at the designated screen position. Note that in 40 column mode the 4-pixel character width may cross attribute byte boundaries in the display file (e.g. the character may have 2 pixels in one byte and 4 in the next). The attribute bytes for these locations may be different. The value of the attribute byte controlling the starting location of the character will be returned.

Output: BC = 1 for invalid parameters
        BC = attribute byte
        Bit 7 - FLASH
        Bit 6 - BRIGHT
        Bit 5
        Bit 4 - PAPER
        Bit 3 /
        Bit 2
        Bit 1 - INK
        Bit 0 /

Name: GETCUR (GETCUR from BASIC)

Input: None

Description:
Returns in the BC register pair and in the BASIC parameter location LINCCL, the current print position (where the next character would be displayed).

Output: E = Line number (0-23)
        C = Column number (0-39) or (0-41)
        BASIC LINCCL - Column Number
        BASIC LINCCL + 1 - Line Number

NOTE: If the last character was printed at Col.39 (41) of Line 23 (last position on the screen), then Col.0/Line 23 will be returned.

---------
**WASSA**

**Memory Usage:**

This package of machine code routines includes the following internal variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
</table>
| SCRCTL | 2    | Scroll Control
|        |      | LSB = Starting line number
|        |      | MSB = Number of lines to be scrolled                                       |
| BOTLN  | 1    | Bottom Line - Line number (0-23) after which test for scroll will be made.  |
| SCRLCT | 1    | Scroll Count - Number of times + 1 that automatic scroll will be done. When decremented to zero, data will not be displayed and a condition code will be returned to user. |
| CRTBL  | 2    | Character Table (Base Address-100H)                                          |
| GRTEL  | 2    | Std. Graphics Character Table (Base-100H)                                   |
| LINLEN | 1    | Line Length - (40 or 42 when in 40-Col. Mode)                               |
| CURPOS | 2    | Current Position (Internal Format)
|        |      | LSB = Column Position
|        |      | MSB = Line Position                                                          |
| DFADDR | 2    | Current Display File Address                                                |
| MASKB  | 1    | Working Byte - (P FLAG Shifted Right 1)
|        |      | (bit 0 = OVERRIDE)
|        |      | (bit 2 = INVERSE)
|        |      | (bit 4 =INK Complement of PAPER)
|        |      | (bit 6 = PAPER Complement of INK)                                           |
| ATRBYT | 1    | Working Byte - (Copy of ATTR P)                                             |
| STINDX | 1    | "Get" Index - Used by STCHAR                                                 |
| STRGCT | 2    | String Count - Contains remaining byte count when BC=3 (Screen Full) is returned from the Write String service WRSTRG (W$STRB). |
| MARGIN | 1    | Margin - Margin Adjust (0/1)                                                |
| DFBIT  | 1    | Display File Bit - Current Bit Position                                     |
| ATTMSK | 1    | Working Byte - (Copy of MASK P)                                             |

209
Initial values set via INIT40 (INIT4E) are as follows:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCRCTL</td>
<td>1701M</td>
</tr>
<tr>
<td>BOTLM</td>
<td>17M</td>
</tr>
<tr>
<td>SCRCT</td>
<td>1M</td>
</tr>
<tr>
<td>CMRBL</td>
<td>(Internal to Module)</td>
</tr>
<tr>
<td>GRBL</td>
<td>(Internal to Module)</td>
</tr>
<tr>
<td>L2WLEN</td>
<td>28M</td>
</tr>
<tr>
<td>CLRSQS</td>
<td>1029M</td>
</tr>
<tr>
<td>DFADDR</td>
<td>4000M</td>
</tr>
<tr>
<td>GTINDX</td>
<td>80M</td>
</tr>
<tr>
<td>STRGCT</td>
<td>0M</td>
</tr>
<tr>
<td>MARGIN</td>
<td>1M</td>
</tr>
<tr>
<td>DBFST</td>
<td>7M</td>
</tr>
</tbody>
</table>

The following are the variables used for passing parameters in BASIC and their values as initialized by INIT4E:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAB</td>
<td>1</td>
<td>0M</td>
</tr>
<tr>
<td>LINCOL</td>
<td>2</td>
<td>0M</td>
</tr>
<tr>
<td>COLCTR</td>
<td>2</td>
<td>1800M</td>
</tr>
<tr>
<td>GETCTR</td>
<td>2</td>
<td>0M</td>
</tr>
</tbody>
</table>

In addition, VIDMOD, PARAMS and other system variables must be available to these routines. At minimum, chunks 0-3 and chunk 7 must be enabled in the Home Bank.

Location:

This package can be incorporated into a machine code program within the memory range from chunk 4 through 7, taking into consideration the remapping of certain structures when the second display file is open (Dual Screen Mode only). NOTE: Machine code above RAMTOP is not moved.

Registers:

Other than as documented for output values, no claims are made as to preservation of any register contents except for the 1Y Register which must always contain the value 5C3AH for access to the standard system variables.

Automatic Scrolling:

As initialized, text for scrolling will be made when the print position goes to the next line following the bottom line on screen (BOTLM+13=24th line). Condition Code 1 (Screen Full) will be returned since SCRCTL will decrement to zero. If SCRCTL is set to some larger values, then the parameters in SCRCTL will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a PCEI or setting the variables BOTLM and SCRCTL to the desired values, automatic scrolling can be done using smaller sections of the screen. When working from BASIC it is recommended that BOTLM be set to line 21 (15H) and SCRCTL=1 be set to 21 (15H) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Note that once SCRCTL expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the "Screen Full" condition.

By setting the SCRCTL variable and invoking the SCRLL (SCRLLS) routine any portion of the screen may be scrolled at any time.
Margin Control:

In 40-Column Mode, there are actually 42 character positions per line. The variable MARGIN determines the offset of the beginning of the 40 column line from the left side of the screen and has valid offset values of 0 or 1. An offset value of 1 centers the 40 column line on the screen; 0 begins at the extreme left side. The default value is 1. When MARGIN is set to 0, the variable LINLEN can be set to 42 to permit access to the 2 extra print positions. Whenever MARGIN and/or LINLEN are modified, a "Set Cursor" operation should be done to insure the integrity of the print position.

NOTE: Since the different MARGIN values result in different pixel positions for the columns, care must be taken in mixing line length and margin values on the same line.

Attribute Control:

Attribute and masking (Inverse/Over) control information will be taken from the system variables ATT P, MSL P and ? PLS as defined below. These variables contain the "permanent" attribute controls set via the BASIC command PAPER, INK, BRIGHT, FLASH, INVERSE, and OVER, or by directly writing to the specified locations.

In 40-Column Mode, the 6-pixel character width results in characters crossing attribute byte boundaries in the display file. Every four columns across a line are controlled by three attribute bytes. This constraint must be taken into consideration when mixing attributes within a line. Inverse and Over are applied to individual characters and are therefore not subject to the above limitation.

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATT P</td>
<td>23693</td>
<td>Bit 7 - FLASH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - BRIGHT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - PAPER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3/</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 - INK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0/</td>
</tr>
<tr>
<td>MSL P</td>
<td>23694</td>
<td>SAME FORMAT AS ATT P.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>USED FOR &quot;TRANSFARPT&quot; DISPLAYS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For each bit that is set to 1, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>corresponding information will be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>taken from the current screen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>position instead of free</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATTP.</td>
</tr>
<tr>
<td>PLS</td>
<td>23697</td>
<td>BIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7 - PAPER+COMPLEMENT OF INK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - INVERSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - OVER (Same Characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INK'd with INK)</td>
</tr>
</tbody>
</table>

ADDITIONAL NOTES:

1. All screen operations done by the system ROM (PRINT, LIST, Edit line [FO, etc.] work with the standard 8-pixel wide characters.
APPENDIX C-4

DUAL SCREEN MODE

Name: Dual Screen Mode Support

Description:
This component provides support to the application programmer for using the dual-screen capability of the TS 2063. The services include opening/closing the second display file (moving the machine stack, DS RAM routines and BASIC structures), position controls, clear screen and scroll screen services, and display of characters. In addition, services are provided to control which display file is active at the screen and which is the target of the current screen operation, as well as a Copy Service and an Exchange Service to transfer screen data/attributes within or between the two display files. "Get" services are provided to return the current display position, the character code for a specified display position, or the attribute byte for a specified display position.

For ease of use from BASIC, status is returned in the BC registers pair, usually zero for successful completion and designated non-zero values for other conditions. The interface from BASIC is by means of PCK'ing the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chunk 7 (1000h-371h).

Attributes and other display controls such as Inverse are taken from the standard TS2063 System Variables (see Usage Section.) The system variable VIMODE is used to determine which display file is the target of the requested service.

---

**Standard Services**

---

**Note:** SETMODE (SETMDE from BASIC - parameter to VIMODE)

**Input:**
- MODE: 0 = Normal (Display File 1 only)
- 128+(0h) = Display File 2 (DF2) Open
- 1 = Display File 1 Active
- 2 = Display File 2 Active
- 3 = Screen Operations to DF1
- 4 = Screen Operations to DF2

From Machine Code: Mode Parameter in A
From BASIC: Mode Parameter in VIMODE

223
Description:

Mode values of 0, 328 or 1 will cause an update of the video mode hardware and the clearing or closing of the second display file as needed. This involves determining if there is enough free RAM to free the second display file and if so, moving the BASIC structures and machine language variables area up and the UDG (User-Defined Graphics) area down to make space for the machine stack and OS RAM routines at the top of memory. The second display file is cleared to zeros. The affected system and internal variables are updated or initialized (see Usage Section). When returning to Mode 0 the structures are returned to their normal locations. In these modes the designated Display File is active at the screen and is the target of all screen operations.

The mode values of 4 and 5 do not affect the hardware, but are used to redirect screen operations to the desired display file. This permits building of a display file prior to activating it at the screen.

The screen position is set to "move" (Line 0/Col.0) whenever SETMODE (SEIMOD) is executed.

NOTE: All screen operations of the TS 2008 System BASIC Interpreter including program entry, system messages, LIST, PRINT, PLOT, DRAW, etc. work only in Display File 1.

NAME: CLRSRC (CLRSRC from BASIC - parameters to CLSCRL)

Input: Line Count (1-24)
Starting Line Number (0-23)

From Machine Code: Line Count In Register B
Starting Line In Register C

From BASIC: Starting Line Number In CLSCRL
Line Count In CLSCRL + 1

Description:

Clears background color (PAPERS) the designated number of lines, beginning with the Starting Line Number. Line 0 with a count of 24 clears the entire screen. Upon return, the cursor position is at the beginning of the first line cleared.

Output: BC = 0 for successful completion
BC = 1 invalid parameters (Line Number + Line Count (1 or 24))

NAME: SETCUE (SETCUE from BASIC - parameters to LINCUE)

Input: Line Number (0-23)
Column Number (0-31)

From Machine Code: Line Number In Register B
Column Number In Register C

From BASIC: Column Number In LINCUE
Line Number In LINCUE + 1

Description:

Converts the requested position to internal format, determines display file address, and stores the values for use by the next display character operation. Note that once established, the position is updated automatically when a character is displayed so that it is only necessary to set the position when sequential display is not desired.

NOTE: The screen position used and maintained by these service routines is independent of that used by the System ROM (S POW).

Output: BC = 0 for successful completion
BC = 1 for invalid parameters (Line Number > 23 ,
Column Number > Line Length-1)
NAME: UCHRB (UCHAR from BASIC - parameter to DATAB)

INPUT: Character code for character to be displayed

2DH TO 7FH - Std. 752068 Character Set
B0H TO B9H - Std. Graphics Set
90H TO A7H - User-Defined Graphics Set

From Machine Code: Register A
From BASIC: in DATAB

DESCRIPTION:
Displays character at current cursor position, applying current attributes and mask. Moves cursor position to next sequential position. If character would start a new line after BOLN (see usage section) and the scroll count (variable SCRCTL) decrements to zero, the character will not be displayed and return will be made with RC = 3 (screen full). If the scroll count does not decrement to zero, the screen will be scrolled up one line using the information in SCRCTL and the new line started at the vacated line.

OUTPUT:
RC = 0 for successful completion
RC = 1 for invalid character code
RC = 3 for screen full

NAME: USTRB (USTRB from BASIC - string identifier in PARAMS)

INPUT: Character Code String

From machine code: Address of string in ML
Count in BC
From BASIC: String variable identifier in System Variable PARAMS = 23747 (SCCM)

DESCRIPTION:
Displays the characters from the string, beginning at the current cursor location and continuing sequentially until the count reaches, or "Screen Full" is detected (see UCHR description and Usage Section on Automatic Scrolling). For the Screen Full condition, the remaining count is stored in the internal variable SCRCTL for access by the user.

NOTE: Characters within the string which are outside of the supported range (32 through 16D (20H-40H) will be ignored, e.g., BASIC 'Asc' codes and control codes embedded in an INPUT string will not be displayed or decoded.

From BASIC, PDE the code for the string variable identifier into PARAMS prior to invoking USTRB, e.g.,

0003 LET astr---string--------
0010 PDE 23747;CODE =a
0015 IF USTR (USTRB) = 3 THEN -----(continue)

OUTPUT:
RC = 0 Successful
RC = 2 BASIC - String not found
RC = 3 Screen full - Remaining Count in SCRCTL
(ML=Current address in string)

NAME: SCROLL (SCRLB from BASIC - parameters to SCRCTL)

INPUT: Line Count (1-23)
Starting Line number (1-23)

From Machine Code: Line Count in B
Starting Line in C
From BASIC: Line Count in SCRCTL
Line Count in SCRCTL + 1

225
Description:

Scrolls the designated number of lines up 1 position, starting at
the specified line number and inserts a blank line at the bottom
of the scrolled area. Line 1 with a count of 25 will scroll the
entire screen up 1 line. Upon return, the cursor position is at
the beginning of the inserted blank line.

Note: See usage section on ‘automatic’ scrolling.

Output: BC = 0 Successful
BC = 1 Invalid Parameters
(Line Number + Line Count < 1 or > 24)

Name: GTCHAR (GETCHB from BASIC - parameters to GETCTL)

Input: Line/Column position as for SETCU

From Machine Code: Line Number in B
Column Number in C

From BASIC: Line Number in GETCTL
Line Number in GETCTL + 1

Description:

Returns in register C of the BC register pair the character code
for the character at the designated screen position. If no match
against the character set (including the standard and
user-defined graphics) is found, zero is returned. (Character
code or zero also returned in A.)

Note: Positions "printed" using the OVER technique will return
zero if they do not match against any single character.

Output: BC = 0 for no find
BC = 1 Invalid parameters
BC = character code (20h-4Fh)

Name: GETATT (GETATB from BASIC - parameters to GETCTL)

Input: As for GTCHAR

Description:

Returns in register C of the BC register pair the attribute byte
for the character at the designated screen position.

Output: BC = 1 for invalid parameters
BC = attribute byte
Bit 7 = PLSEM
Bit 6 = BRIGHT
Bit 5
Bit 4 = PAPER
Bit 3 /
Bit 2
Bit 1 = INK
Bit 0 /

Name: GETCU (GETCUB from BASIC)

Input: None

Description:

Returns in the BC register pair and in the BASIC parameter
location LINCOL, the current print position (where the next
character would be displayed).

Output: E = Line number (0-23)
C = Column number (0-31)

BASIC: LINCOL = Column number
LINCOL + 1 = Line number

Note: If the last character was printed at Col.31 of Line
23 (last position on the screen), then Col.0/Line 23
will be returned.
Name: COPYSC (CPYSE from BASIC - string identifier in PARAMS)

Input:
Source Line (0-23)
Destination Line (0-23)
Source Display File (1 or 2)
Destination Display File (1 or 2)
Line Count (1-24)

Free Machine Code:
Source Line in B
Destination Line in C
Source DF in D
Dest DF in E
Line Count in A

Free BASIC: Parameters in string variable, separated by commas; String Identifier in System Variable PARAMS - 23747 (ECCM)

Description:
Copies the designated portion of the source display file to the designated portion of the destination display file. The source and destination may be in the same display file. During a multi-line copy, source and destination operands are accessed in a top to bottom fashion; therefore, in certain cases of operand overlap, the operation may be destructive, i.e., the source data may be modified before it is copied. To get the desired result, it may be necessary to copy individual lines.

Free BASIC, the following example would copy lines 0 thru 20 from DP1 to lines 0 thru 12 in DP2:

LET as=#0,0,1,2,13* (Starting Line)
DPDestination Line
1=Source DF
2=Dest DF
13=Line Count

POKE 23747,CODE "as* (String Identifier to PARAMS)
LET cc=USR (CPYSE)
IF cc<>0 THEN..............

Output: BC=0 Successful
BC=1 Invalid Parameters (CPYSE) or 21 Line No. 0-231 Line Count 0-24
BC=02 BASIC - String Not Found

Name: EXCHSC (EXCHSC from BASIC)

Input:
Source "1" Display File (1 or 2)
Source "2" Line (0-23)
Source "2" Display File (1 or 2)
Source "2" Line (0-23)
Line Count (1-24)

Free Machine Code:
Source "1" DF in B
Source "1" Line in C
Source "2" DF in D
Source "2" Line in E
Line Count in A

Free BASIC: Parameters in string variable, separated by commas, in the order Source "1" Line, Source "2" Line, Source "2" DF, Source "2" Line Count, String Identifier in System Variable PARAMS - 23747 (ECCM)

Description:
Exchanges the specified number of lines between Source "1" and Source "2". Both sources may be in the same Display File. The exchange is done by ORDERing the two sources together three times. The operation is done a line at a time, accessing the two sources in a top to bottom fashion, e.g., to exchange lines 0-12 in DP1 with lines 16-20 in DP2 would first exchange line 0 (DP1) with line 16 (DP2), then line 1 with line 17 and proceed in this fashion through exchange of line 11 with line 20.

Free BASIC, the following example would exchange lines 0 thru 4 in DP1 with lines 0-12 in DP2:

LET as=#0,0,1,2,13* (Starting Line)
1=Source "1" Line
2=Source "2" Line
3=Source "1" DF
2=Source "2" DF
5=Line Count

POKE 23747,CODE "as* (String Identifier to PARAMS)
LET cc=USR (EXCHSC)
IF cc<>0 THEN..............

Output: BC=0 Successful
BC=1 Invalid Parameters (EXCHSC) or 21 Line No. 0-231 Line Count 0-24
BC=02 BASIC - String Not Found
USBSEI

Memory Usage

This package of machine code routines includes the following internal variables:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCRTCL</td>
<td>2</td>
<td>Scroll Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LSB = Starting line number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSB = Number of lines to be scrolled</td>
</tr>
<tr>
<td>BDTHW</td>
<td>1</td>
<td>Bottom Line - Line number (0-23) after which test for scroll will be made.</td>
</tr>
<tr>
<td>SCLCTY</td>
<td>1</td>
<td>Scroll Count - Number of times +</td>
</tr>
<tr>
<td>CNTL</td>
<td>2</td>
<td>Character Table (Base Address-100H)</td>
</tr>
<tr>
<td>GETBL</td>
<td>2</td>
<td>Std.Graphics Character Table (Base-100H)</td>
</tr>
<tr>
<td>LINLEN</td>
<td>1</td>
<td>Line Length</td>
</tr>
<tr>
<td>CURPOS</td>
<td>2</td>
<td>Current Position (Internal Format)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LSB = Column Position</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSB = Line Position</td>
</tr>
<tr>
<td>DPADDR</td>
<td>2</td>
<td>Current Display File Address</td>
</tr>
<tr>
<td>MASKS</td>
<td>1</td>
<td>Working Byte - (P.FLAG Shifted Right 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(bit 0 = OVER)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(bit 2 = INVERSE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(bit 4 = INK Complement of PAPER)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(bit 6 = PAPER Complement of INK)</td>
</tr>
<tr>
<td>ATTRBT</td>
<td>1</td>
<td>&quot;Get&quot; Byte - (Copy of ATTR #)</td>
</tr>
<tr>
<td>GTMN0</td>
<td>1</td>
<td>&quot;Get&quot; Index - Used by GTCMAR</td>
</tr>
<tr>
<td>STRCTY</td>
<td>2</td>
<td>String Count - Contains remaining byte count when EOM (Screen Full) is returns from the write String service WSTRG (WSTPE).</td>
</tr>
<tr>
<td>ATTRBS</td>
<td>1</td>
<td>Working Byte - (Copy of MASK #)</td>
</tr>
</tbody>
</table>

Initial values set via SETMODE (SETMOD) when the second display file is first opened are as follows:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCRTCL</td>
<td>1701H</td>
</tr>
<tr>
<td>BDTHW</td>
<td>17H</td>
</tr>
<tr>
<td>SCLCTY</td>
<td>1H</td>
</tr>
<tr>
<td>CNTL</td>
<td>3000H</td>
</tr>
<tr>
<td>GETBL</td>
<td>(Internal to Module)</td>
</tr>
<tr>
<td>LINLEN</td>
<td>20H</td>
</tr>
<tr>
<td>CURPOS</td>
<td>182H</td>
</tr>
<tr>
<td>DPADDR</td>
<td>4000H/6000H</td>
</tr>
<tr>
<td>GTMN0</td>
<td>80H</td>
</tr>
<tr>
<td>STRCTY</td>
<td>0H</td>
</tr>
</tbody>
</table>

The following are the variables used for passing parameters in BASIC. The * indicates these initialized by SETMOD when the second display file is first opened:

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>*DATAR</td>
<td>2</td>
<td>0H</td>
</tr>
<tr>
<td>*LIMCCL</td>
<td>2</td>
<td>0H</td>
</tr>
<tr>
<td>*CLRTCL</td>
<td>2</td>
<td>1800H</td>
</tr>
<tr>
<td>*GTCRTL</td>
<td>2</td>
<td>0H</td>
</tr>
<tr>
<td>*VIMODE</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

In addition, VIDMOD, PARAM, and other system variables must be available to these routines. At minimum, chunks 0-3 and chunk 7 must be enabled in the Menu Bank.

Location:

This package can be incorporated into a machine code program within the memory range free chunk 4 through 7, taking into consideration the remapping of certain structures when the second display file is open.

NOTE: Machine code above RAFTOP is not used.
Registers:

Other than as documented for output values, no claims are made as to preservation of any register contents except for the IF register which must always contain the value $0138$ for access to the standard system variables.

Automatic Scrolling:

As initialized, test for scrolling will be made when the print position goes to the next line following the bottom line on screen (BOTLN=23+24th line). Condition Code 3 (Screen Full) will be returned since SCRLTY will decrement to zero. If SCRLTY is set to some larger value, then the parameters in SCRLTY will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a PGR or setting the variable BOTLN and SCRLTY to the desired values, automatic scrolling can be done using smaller sections of the screen. When working with BASIC it is recommended that BOTLN be set to Line 21 (ISH) and SCRLTY be set to 21 (ISH) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Note that once SCRLTY expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the "Screen Full" condition.

By setting the SCRLTY variable and invoking the SCROLL (SCRLS) routine any portion of the screen may be scrolled at any time.

Attribute Control:

Attribute and making (In reverse/Over) control information will be taken from the system variables ATTR_P, MASK_P and P_FLAG as defined below. These variables contain the "permanent" attribute controls set via the BASIC commands PAPER, INK, BRIGHT, FLASH, INVERSE, and OVER, or by directly writing to the specified locations.

<table>
<thead>
<tr>
<th>NAME</th>
<th>ADDRESS</th>
<th>CONTENTS</th>
</tr>
</thead>
</table>
| ATTR_P | 23693   | Bit 7 - FLASH 6 - BRIGHT 5 - PAPER 3 - INK 1 - INK 0 - 
|        |         | SAME FORMAT AS ATTR_P. USED FOR "TRANSPARENT" CISPLOT. For each bit that is set to 1, the corresponding information will be taken from the current screen position instead of from ATTR_P. |
| MASK_P | 23694   | SAME FORMAT AS ATTR_P. USED FOR "TRANSPARENT" CISPLOT. For each bit that is set to 1, the corresponding information will be taken from the current screen position instead of from ATTR_P. |
| P_FLAG | 23697   | Bit 7 - PAPER=COMPLEMENT OF INK 5 - INK=COMPLEMENT OF PAPER 3 - INVERSE 1 - OVER (New Characters ARM'D with Old) |

Using System ROM Screen Services:

The following technique can be used to build screens using the BASIC commands such as PRINT, PLOT, CIRCLE, and DRAW or other System ROM routines and get them into the second display file:

1. Set Mode 80 (scene 2nd DP)
2. Build Screen in DP1 using System or ADL Services
3. Copy CPS to DP2 using COPYSC (COPYSB)
4. Set Mode 1 (DP2 to Screen)
5. Set Mode 6 (DP2 at Screen/Operations to DP1)
6. Build Screen in DP1 using System or ADL Services
7. Set Mode 80 (DP1 to Screen)
8. Go to Step 3

The switching of the screen from one display file to the other is transparent to the viewer where the files contain the same data.
WARNING

When the second display file is active on the screen, any system messages such as SCROLL? error reports, or use of the INPUT command will not be visible and the system may appear to be "hung". Indeterminate pressing keys may be filled on the "invisible" Edit Line with "garbage" thus further aggravating the situation. The PNG FACILITIY can be used to intercept some of these situations and set the slave mode to use Display File 1 at the screen in order that the message can be seen and responded to. Otherwise, it is necessary to key in and execute from the FIDLL without being able to see it.
236

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATT8Y</td>
<td>127# 597 1567 14128</td>
</tr>
<tr>
<td>ATT8X</td>
<td>1249# 1369 12499</td>
</tr>
<tr>
<td>ATT90</td>
<td>259# 259 7258</td>
</tr>
<tr>
<td>BTLA</td>
<td>1493# 14056</td>
</tr>
<tr>
<td>CALCS</td>
<td>429 566 677 1081 1004 1141 1164</td>
</tr>
<tr>
<td>CALCPS</td>
<td>869 974 1209 13776</td>
</tr>
<tr>
<td>CHNGY</td>
<td>52# 772</td>
</tr>
<tr>
<td>CHSSET</td>
<td>109# 742 747</td>
</tr>
<tr>
<td>CNTR</td>
<td>1470 281 590 7458</td>
</tr>
<tr>
<td>CM_400</td>
<td>61# 835 940 10768 561 561 8730</td>
</tr>
<tr>
<td>CLINT</td>
<td>40# 727</td>
</tr>
<tr>
<td>CLSCCL</td>
<td>104# 281 7288</td>
</tr>
<tr>
<td>CLSCPS</td>
<td>407 4008</td>
</tr>
<tr>
<td>CLSICD</td>
<td>441 493 4976</td>
</tr>
<tr>
<td>CLSICS</td>
<td>516# 567</td>
</tr>
<tr>
<td>CLSICJ</td>
<td>573# 576</td>
</tr>
<tr>
<td>CLSICL</td>
<td>524# 543</td>
</tr>
<tr>
<td>CLSICM</td>
<td>533# 533</td>
</tr>
<tr>
<td>CLSICY</td>
<td>510 5706</td>
</tr>
<tr>
<td>CLSICZ</td>
<td>50# 1076</td>
</tr>
<tr>
<td>CLSICD</td>
<td>60 6335</td>
</tr>
<tr>
<td>CLSCS</td>
<td>40 1076</td>
</tr>
<tr>
<td>CONVHF</td>
<td>337 988 675 680 12908</td>
</tr>
<tr>
<td>CONVPS</td>
<td>64# 9286</td>
</tr>
<tr>
<td>CONVPS</td>
<td>67 870 8780</td>
</tr>
<tr>
<td>CONVPS</td>
<td>67 870 8786</td>
</tr>
<tr>
<td>CONVPS</td>
<td>955 945 7476</td>
</tr>
<tr>
<td>CONVPS</td>
<td>970 8749</td>
</tr>
<tr>
<td>CONVPS</td>
<td>106 8249</td>
</tr>
<tr>
<td>CONVPS</td>
<td>944# 1069</td>
</tr>
<tr>
<td>CONVPS</td>
<td>9416 1069</td>
</tr>
<tr>
<td>CPYC</td>
<td>79# 1025</td>
</tr>
<tr>
<td>CPYD</td>
<td>79# 956</td>
</tr>
<tr>
<td>CPYD</td>
<td>79# 1025</td>
</tr>
<tr>
<td>CPOPS</td>
<td>1025</td>
</tr>
</tbody>
</table>
APPENDIX C-5

SPRITE GRAPHICS SUPPORT

Name: Sprite Graphics Support

Description: This component provides support for sprite graphics. A sprite is a graphical object which can be created and manipulated by a set of services to be described below. A sprite can be up to 256 x 256 characters in size, however at most 32 x 32 can be displayed. Each sprite is identified by a sprite-id, a number greater or equal to zero, and has the following properties:

SPRINTALO - a pointer to a width x height character bitmap, where width and height are user defined. A 1 indicates foreground color, a 0 indicates background color.

SPRINTLOC - the row, col location on the screen of the upper left corner of the sprite.

SPRINTC - the screen attribute (foreground color). A value of 0 indicates black, 1 indicates blue, etc.

SPRINTB - the width and height of the sprite (in characters).

The sprite graphics package assumes that the system variables area of memory is not used by an application. The sprite support services can be invoked from machine code and BASIC. There is a BASIC interface routine, which takes as its parameter a BASIC string variable. The BASIC interface and sprite services code is loaded at SPCODE, where SPCODE = 00000H. The name of this variable is specified by adding in the starting character at address SPCODE-$. For example, if space SPRCDE-$, CODE "C" is executed, the variable CS will be used to pass commands to the interface routine. This must be done before using any sprite services. The interface routine is invoked by LET variable = USE SPCODE. The status code returned by the last sprite service executed is assigned to variable.

The BASIC interface routine causes the report "$A: Invalid argument" to be produced whenever an illegal command or invalid argument to a command is found. If too few or too many arguments are given for some command, the report "Q: Parameter error" is produced. If the command string variable cannot be found, the report "$C: Nonsense in BASIC" is produced. If there is not enough memory for the number of sprites specified, the report "$A: Out of memory" is produced (see Initialize). There exists a variable called COMMANO, at address SPCODE-9, which contains the number of the last sprite command to be executed. Thus if an error occurs in a sprite command, this variable can be tested to find out which command in the command string caused the error. The first command in the command string is the command zero.

The machine code routines all return status codes in the BC register pair. If the value in $C is 0, then no error occurred. In this case 0 contains a value indicating further status information. If an error occurred, C will contain an odd value (thus bit 0 is set).

The BASIC interface string variable contains sprite graphics commands, separated by spaces. The commands have the following syntax:

"<command letter> <parameter list>"

where <command letter> is a single upper or lower case letter identifying the command:

A = Init_Sprites
B = Initscreen
C = Create_Sprite
D = Set_AttrWrap_Mode
E = Put_Sprite
F = Erase_Sprite
G = Novel_Sprite
H = Change_Sprite_Location
I = Change_Sprite_Attribute
J = Overlay
K = Horizontal_Scroll
L = Vertical_Scroll

and where <parameter list> is a list of numbers separated by commas. The string assigned to C$ can be a string expression, i.e., "<substr>"SSTR(123)+"substr>". The semantics of these commands will be described in detail below.
APPLICATION SERVICES

Name: Init_Sprites

Machine code interface:

Inputs: A = max_sprites; 0..255

Outputs: BC = 0000h - OK
         = 0001h - not enough memory

Entry: SPRIVC = SPRICE + 210h

BASIC interface:

Inputs: LET C# = "max_sprites"

Description: This service initializes the data structures used by the sprites services. Memory immediately below the sprites code is used for global variables and memory immediately below this is used to store sprite definition. Each sprite takes up 6 bytes of data (not including its definition, which is stored elsewhere). Thus the total amount of memory used by the sprites package is:

A bytes + max_sprites * 20 byte globals area + 276 byte code area

If there is not enough memory between RAMTOP and the global variables area for the number of sprites specified, 0001h is returned. Thus, RAMTOP should be set to

SPRICE - 29 = 8 max_sprites

before calling Init_Sprites. This service must be called before any of the others.

BASIC example: "112" initializes memory for 32 sprites.

Name: Init_Screen

Machine code interface:

Inputs: A = screen_height; 0..24
         D = background_color; 0..7
         E = border_color; 0..7

Outputs: BC = 0000h - OK
         = 0001h - illegal screen ht
         = 0005h - illegal color

Entry: SPRIVC = 2D0h

BASIC interface:

Inputs: LET EQ = "(screen ht)<background_color>><border_color>"

Description: This service clears the top screen ht lines of the screen and sets their color to the background_color. The bottom 24-screen ht lines are cleared and set to the border_color. The top screen ht lines are used for displaying sprites. The remaining lines can be used for text display. The border is set to the border_color.

BASIC example: "S12.5;1" clears the top 12 lines and sets their color to cyan, clears the bottom two lines and sets their color to blue, and sets the border color to blue.

248
Name: Create_Sprite

Machine code interface:
Inputs: A = sprite-id, 0..max-sprites - 1  
B = width, 0..255  
C = height, 0..255  
D = color, 0..7  
ML = definition address, 0..64K-1  
Outputs: BC = 0000h - OK (sprite created)  
         = 0003h - illegal sprite-id  
Entry: SPRSVC + 58H

BASIC interface:
Inputs: LET C8 =  
         "C(sprite-id),<width>,<height>,<color>,<color>"

Description: This service creates a sprite with the 
specified width, height, color, and definition address. 
Initially, the position of the sprite is (0, 0). If there 
was already a sprite with the specified id, it is destroyed. 
In all further operations the new sprite is identified by 
its sprite-id.

BASIC example: "C0,2,2,2,32768" creates sprite 0 which is 2 
* 2 characters in size, is red in color, and whose 
definition is located at 32768. The initial position of the 
sprite is (0, 0).

Name: Set_Autoauto_Mode

Machine code interface:
Inputs: A = mode: 0 - off, not 0 - on  
Outputs: BC = 0000h - OK  
Entry: SPRSVC + B0H

BASIC interface:
Inputs: LET C8 = "<mode>"

Description: This service turns the Autoauto mode on and 
off, when the Autoauto mode is on, all changes to a 
sprite's location are made so that the new location is on 
the screen (wraparound).

BASIC example: "WI" causes Autoauto mode to be turned on.

Name: Put_Sprite

Machine code interface:
Inputs: A = sprite-id, 0..max-sprites - 1  
D = row, 0..255  
E = column, 0..255  
Outputs: BC = 0000h - OK; nothing overwritten  
         = 0100h - OK; something overwritten  
         = 0003h - illegal sprite-id  
         = row  
         = column  
Entry: SPRSVC + B8H

BASIC interface:
Inputs: LET C8 = "P(sprite-id),<row>,<column>"

Description: This service writes a sprite on the screen. 
The row and column described above define the location at 
which the upper left corner of the sprite is written. If 
the Autoauto mode is set, the row and column values are 
modified to make sure that the sprite location is a legal 
screen location. The new row and column values are 
returned. If the Autoauto mode is not set, the position is 
not changed. Moreover, only those parts of the sprite 
corresponding to legal screen positions will be displayed. 
If writing the sprite causes something on the screen to be 
overwritten, the appropriate code is returned.

BASIC example: "P0.11.0" puts sprite 0 at row 11, column 0. 
Anything already at that location is overwritten. If 
anything is overwritten, the interface routine returns 10h.
Name: Erase_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..max-sprites = 1
Outputs: BC = 0000h - OK (sprite erased)
         0003h = illegal sprite-id
Entry: SPRESV = 198h

BASIC interface:

Inputs: LET C8 = \$c(sprite-id)\$

Description: This service erases a sprite. The screen
becomes blank where the sprite used to be. The sprite still
exists and can be written elsewhere.

BASIC example: \$70\$ erases sprite 0. The screen locations
taken up by the sprite become blank.

Name: Move_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..max-sprites = 1
         0 = relative vertical motion, -128..127
         E = relative horizontal motion, -128..127
Outputs: BC = 0000h - OK, nothing overwritten
         0100h - OK, something overwritten
         0003h - illegal sprite-id
         D = row
         E = column
Entry: SPRESV = 200h

BASIC interface:

Inputs: LET C8 = \$c(sprite-id),<vert>,<hor>\$

Description: This service moves a sprite. The sprite is
erased at its current location and written at a new location
defined by:

new row = old row + relative vertical motion
new col = old col + relative horizontal motion

The location of the sprite is updated to reflect the motion.
As with the \$FF\$ command, if the sprite overwrites anything,
10h is returned. If Autoresize mode is set, then the location
is automatically wrapped to fit onto the screen (i.e., E4, C0)
becomes (0, 0). The new row and column values are
returned. If Autoresize mode is not set, only those parts of
the sprite which correspond to legal screen locations will
be displayed.

BASIC example: \$M0,2,0\$ changes the position of the sprite
from (11, 0) to (13, 0). This operation is equivalent to
\$EO 00,13,0\$.  

Name: Change_Sprite_Location

Machine code interface:

Inputs: A = sprite-id, 0..max-sprites = 1
         0 = new row, 0..255
         E = new column, 0..255
Outputs: BC = 0000h - OK (location changed)
         0003h - illegal sprite-id
Entry: SPRESV = 237h

BASIC interface:

Inputs: LET C8 = \$c(sprite-id),<new-row>,<new-col>\$

Description: This service updates the position property of
a sprite in the same fashion as for Put_Sprite. The sprite
is not updated on the screen until a Move_Sprite,
Put_Sprite, or Erase_Sprite is executed.

BASIC example: \$L0,13,0\$ will set the location of sprite 0
to (13, 0).

---

250
Name: Change_Sprite_Attribute

Machine code interface:

Inputs: A = sprite-id, 0..max-sprites - 1
       D = color, 0..7

Outputs: BC = 0000h - OK (color changed)
         0003h - illegal sprite-id
         000Fh - illegal color

Entry: SPRISVC = 240h

BASIC interface:

Inputs: LET CS = "<sprite-id>,<color>"

Description: This service updates the color property of a
sprite. The sprite on the screen is not updated until
a Put_Sprite, Move_Sprite, or Clear_Sprite is executed.
BASIC example: "A0,2" causes the color property of sprite 0
to be set to red.

Name: Overlap?

Machine code interface:

Inputs: D = sprite-id-1, 0..max-sprites - 1
       E = sprite-id-2, 0..max-sprites - 1

Outputs: BC = 0000h - OK, no overlap
         0200h - OK, overlap
         0003h - illegal sprite-id

Entry: SPRISVC = 240h

BASIC interface:

Inputs: LET CS = "<sprite-id-1>,<sprite-id-2>"

Description: This service is used to detect if two sprites
overlap.
BASIC example: "D0,1" causes the interface routine to
return TBL if sprites C and 1 overlap on the screen.

Name: Vertical_Scroll

Machine code interface:

Inputs: A = direction: positive number = down,
       negative number = up

Outputs: BC = 0000h - OK

Entry: SPRISVC = 550h

BASIC interface:

Inputs: LET CS = "<direction>"

Description: This service is used to scroll the entire
screen in the vertical direction by one row. The direction
of scroll is up if direction is less than 0, otherwise the
scroll is down. The position property of all sprites is
updated by one row to reflect the effect of the scroll. If
AutoScroll mode is not set, then 1 is added to the row value.
If AutoScroll mode is set, then 1 is added to the row value,
and this new value is wrapped to fit on the screen.
BASIC example: "V1" causes the screen to scrolled down one
row.

251
Name: Horizontal_Scroll

Machine code interface:

Inputs: \( X = \text{direction: positive number} \rightarrow \text{right}; \) 
\( \text{negative number} \rightarrow \text{left} \)

Outputs: \( BC = 0000h \rightarrow DB \)

Entry: SPRVC + 3800h

BASIC interface:

Inputs: LET CS = "(direction)"

Description: This service is used to scroll the entire screen in the horizontal direction by one column. The direction of scroll is to the left if direction is less than 0, otherwise the scroll is to the right. The position property of all sprites is updated by one column to reflect the effect of the scroll. If Autosnap mode is not set, then 1 is added to the column value. If Autosnap mode is set, then 1 is added to the column value, and this new value is wrapped to fit on the screen.

BASIC example: "MI" causes the screen to scrolled right one column.
Register Values |
| Status Code |
| Low Level |
| I/O Enables |

Note: The interface between the I/O handler module and the Sprite Services module is the same as the interface between an assembly language program and the Sprite Services module.

Following is the layout in memory of the code and data used by the Sprite Services component of the application development library:

```
<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>0000</td>
</tr>
<tr>
<td>0009</td>
<td>0009</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
</tr>
<tr>
<td>0011</td>
<td>0011</td>
</tr>
<tr>
<td>0012</td>
<td>0012</td>
</tr>
<tr>
<td>0013</td>
<td>0013</td>
</tr>
<tr>
<td>0014</td>
<td>0014</td>
</tr>
<tr>
<td>0015</td>
<td>0015</td>
</tr>
<tr>
<td>0016</td>
<td>0016</td>
</tr>
<tr>
<td>0017</td>
<td>0017</td>
</tr>
<tr>
<td>0018</td>
<td>0018</td>
</tr>
<tr>
<td>0019</td>
<td>0019</td>
</tr>
<tr>
<td>0020</td>
<td>0020</td>
</tr>
<tr>
<td>0021</td>
<td>0021</td>
</tr>
<tr>
<td>0022</td>
<td>0022</td>
</tr>
<tr>
<td>0023</td>
<td>0023</td>
</tr>
<tr>
<td>0024</td>
<td>0024</td>
</tr>
<tr>
<td>0025</td>
<td>0025</td>
</tr>
<tr>
<td>0026</td>
<td>0026</td>
</tr>
<tr>
<td>0027</td>
<td>0027</td>
</tr>
<tr>
<td>0028</td>
<td>0028</td>
</tr>
<tr>
<td>0029</td>
<td>0029</td>
</tr>
<tr>
<td>0030</td>
<td>0030</td>
</tr>
<tr>
<td>0031</td>
<td>0031</td>
</tr>
<tr>
<td>0032</td>
<td>0032</td>
</tr>
<tr>
<td>0033</td>
<td>0033</td>
</tr>
<tr>
<td>0034</td>
<td>0034</td>
</tr>
<tr>
<td>0035</td>
<td>0035</td>
</tr>
<tr>
<td>0036</td>
<td>0036</td>
</tr>
<tr>
<td>0037</td>
<td>0037</td>
</tr>
<tr>
<td>0038</td>
<td>0038</td>
</tr>
<tr>
<td>0039</td>
<td>0039</td>
</tr>
<tr>
<td>0040</td>
<td>0040</td>
</tr>
<tr>
<td>0041</td>
<td>0041</td>
</tr>
<tr>
<td>0042</td>
<td>0042</td>
</tr>
<tr>
<td>0043</td>
<td>0043</td>
</tr>
<tr>
<td>0044</td>
<td>0044</td>
</tr>
<tr>
<td>0045</td>
<td>0045</td>
</tr>
<tr>
<td>0046</td>
<td>0046</td>
</tr>
<tr>
<td>0047</td>
<td>0047</td>
</tr>
<tr>
<td>0048</td>
<td>0048</td>
</tr>
<tr>
<td>0049</td>
<td>0049</td>
</tr>
<tr>
<td>0050</td>
<td>0050</td>
</tr>
<tr>
<td>0051</td>
<td>0051</td>
</tr>
<tr>
<td>0052</td>
<td>0052</td>
</tr>
<tr>
<td>0053</td>
<td>0053</td>
</tr>
<tr>
<td>0054</td>
<td>0054</td>
</tr>
<tr>
<td>0055</td>
<td>0055</td>
</tr>
<tr>
<td>0056</td>
<td>0056</td>
</tr>
<tr>
<td>0057</td>
<td>0057</td>
</tr>
<tr>
<td>0058</td>
<td>0058</td>
</tr>
<tr>
<td>0059</td>
<td>0059</td>
</tr>
<tr>
<td>0060</td>
<td>0060</td>
</tr>
<tr>
<td>0061</td>
<td>0061</td>
</tr>
<tr>
<td>0062</td>
<td>0062</td>
</tr>
<tr>
<td>0063</td>
<td>0063</td>
</tr>
<tr>
<td>0064</td>
<td>0064</td>
</tr>
<tr>
<td>0065</td>
<td>0065</td>
</tr>
<tr>
<td>0066</td>
<td>0066</td>
</tr>
<tr>
<td>0067</td>
<td>0067</td>
</tr>
<tr>
<td>0068</td>
<td>0068</td>
</tr>
<tr>
<td>0069</td>
<td>0069</td>
</tr>
<tr>
<td>0070</td>
<td>0070</td>
</tr>
<tr>
<td>0071</td>
<td>0071</td>
</tr>
<tr>
<td>0072</td>
<td>0072</td>
</tr>
<tr>
<td>0073</td>
<td>0073</td>
</tr>
<tr>
<td>0074</td>
<td>0074</td>
</tr>
<tr>
<td>0075</td>
<td>0075</td>
</tr>
<tr>
<td>0076</td>
<td>0076</td>
</tr>
<tr>
<td>0077</td>
<td>0077</td>
</tr>
<tr>
<td>0078</td>
<td>0078</td>
</tr>
<tr>
<td>0079</td>
<td>0079</td>
</tr>
<tr>
<td>0080</td>
<td>0080</td>
</tr>
<tr>
<td>0081</td>
<td>0081</td>
</tr>
<tr>
<td>0082</td>
<td>0082</td>
</tr>
<tr>
<td>0083</td>
<td>0083</td>
</tr>
<tr>
<td>0084</td>
<td>0084</td>
</tr>
<tr>
<td>0085</td>
<td>0085</td>
</tr>
<tr>
<td>0086</td>
<td>0086</td>
</tr>
<tr>
<td>0087</td>
<td>0087</td>
</tr>
<tr>
<td>0088</td>
<td>0088</td>
</tr>
<tr>
<td>0089</td>
<td>0089</td>
</tr>
<tr>
<td>0090</td>
<td>0090</td>
</tr>
<tr>
<td>0091</td>
<td>0091</td>
</tr>
<tr>
<td>0092</td>
<td>0092</td>
</tr>
<tr>
<td>0093</td>
<td>0093</td>
</tr>
<tr>
<td>0094</td>
<td>0094</td>
</tr>
<tr>
<td>0095</td>
<td>0095</td>
</tr>
<tr>
<td>0096</td>
<td>0096</td>
</tr>
<tr>
<td>0097</td>
<td>0097</td>
</tr>
<tr>
<td>0098</td>
<td>0098</td>
</tr>
<tr>
<td>0099</td>
<td>0099</td>
</tr>
<tr>
<td>0100</td>
<td>0100</td>
</tr>
<tr>
<td>0101</td>
<td>0101</td>
</tr>
<tr>
<td>0102</td>
<td>0102</td>
</tr>
<tr>
<td>0103</td>
<td>0103</td>
</tr>
<tr>
<td>0104</td>
<td>0104</td>
</tr>
<tr>
<td>0105</td>
<td>0105</td>
</tr>
<tr>
<td>0106</td>
<td>0106</td>
</tr>
<tr>
<td>0107</td>
<td>0107</td>
</tr>
<tr>
<td>0108</td>
<td>0108</td>
</tr>
<tr>
<td>0109</td>
<td>0109</td>
</tr>
<tr>
<td>0110</td>
<td>0110</td>
</tr>
<tr>
<td>0111</td>
<td>0111</td>
</tr>
<tr>
<td>0112</td>
<td>0112</td>
</tr>
<tr>
<td>0113</td>
<td>0113</td>
</tr>
<tr>
<td>0114</td>
<td>0114</td>
</tr>
<tr>
<td>0115</td>
<td>0115</td>
</tr>
<tr>
<td>0116</td>
<td>0116</td>
</tr>
<tr>
<td>0117</td>
<td>0117</td>
</tr>
<tr>
<td>0118</td>
<td>0118</td>
</tr>
<tr>
<td>0119</td>
<td>0119</td>
</tr>
<tr>
<td>0120</td>
<td>0120</td>
</tr>
<tr>
<td>0121</td>
<td>0121</td>
</tr>
<tr>
<td>0122</td>
<td>0122</td>
</tr>
<tr>
<td>0123</td>
<td>0123</td>
</tr>
<tr>
<td>0124</td>
<td>0124</td>
</tr>
<tr>
<td>0125</td>
<td>0125</td>
</tr>
<tr>
<td>0126</td>
<td>0126</td>
</tr>
<tr>
<td>0127</td>
<td>0127</td>
</tr>
<tr>
<td>0128</td>
<td>0128</td>
</tr>
<tr>
<td>0129</td>
<td>0129</td>
</tr>
<tr>
<td>0130</td>
<td>0130</td>
</tr>
<tr>
<td>0131</td>
<td>0131</td>
</tr>
<tr>
<td>0132</td>
<td>0132</td>
</tr>
<tr>
<td>0133</td>
<td>0133</td>
</tr>
<tr>
<td>0134</td>
<td>0134</td>
</tr>
<tr>
<td>0135</td>
<td>0135</td>
</tr>
<tr>
<td>0136</td>
<td>0136</td>
</tr>
<tr>
<td>0137</td>
<td>0137</td>
</tr>
<tr>
<td>0138</td>
<td>0138</td>
</tr>
<tr>
<td>0139</td>
<td>0139</td>
</tr>
<tr>
<td>0140</td>
<td>0140</td>
</tr>
<tr>
<td>0141</td>
<td>0141</td>
</tr>
<tr>
<td>0142</td>
<td>0142</td>
</tr>
<tr>
<td>0143</td>
<td>0143</td>
</tr>
<tr>
<td>0144</td>
<td>0144</td>
</tr>
<tr>
<td>0145</td>
<td>0145</td>
</tr>
<tr>
<td>0146</td>
<td>0146</td>
</tr>
<tr>
<td>0147</td>
<td>0147</td>
</tr>
<tr>
<td>0148</td>
<td>0148</td>
</tr>
<tr>
<td>0149</td>
<td>0149</td>
</tr>
<tr>
<td>0150</td>
<td>0150</td>
</tr>
<tr>
<td>0151</td>
<td>0151</td>
</tr>
<tr>
<td>0152</td>
<td>0152</td>
</tr>
<tr>
<td>0153</td>
<td>0153</td>
</tr>
<tr>
<td>0154</td>
<td>0154</td>
</tr>
<tr>
<td>0155</td>
<td>0155</td>
</tr>
</tbody>
</table>
```

253


```assembly
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.256
```
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>reset</code></td>
<td>Returns to the initial state</td>
</tr>
<tr>
<td><code>clear</code></td>
<td>Clears the display</td>
</tr>
<tr>
<td><code>scroll</code></td>
<td>Scrolls the display</td>
</tr>
<tr>
<td><code>increment</code></td>
<td>Increments the display</td>
</tr>
<tr>
<td><code>decrement</code></td>
<td>Decrements the display</td>
</tr>
<tr>
<td><code>add</code></td>
<td>Adds to the display</td>
</tr>
<tr>
<td><code>subtract</code></td>
<td>Subtracts from the display</td>
</tr>
</tbody>
</table>

**Example Code:**

```assembly
; Initial state
reset:
    ret

clear:
    clear:
    ret

scroll:
    scroll:
    ret

increment:
    increment:
    ret

decrement:
    decrement:
    ret

add:
    add:
    ret

subtract:
    subtract:
    ret
```

**Description:**
- The `reset` function returns the display to its initial state.
- `clear` clears the display.
- `scroll` scrolls the display.
- `increment` increments the display by one.
- `decrement` decrements the display by one.
- `add` adds a value to the display.
- `subtract` subtracts a value from the display.

**Further Information:**
- The code above is a simplified example demonstrating how to control a display through assembly language.
- Each function is designed to manipulate the display in a specific way, allowing for basic operations like adding, subtracting, and clearing.
- The use of labels like `reset` and `clear` helps in easily identifying and calling each function within a larger program.
E186    2F    574    le    a, b    Store to convert to 2's complement
E185    47    575    le    b, a
E184    2F    576    le    a, c
E183    2A    577    cucl    b, a
E182    6A    578    rop    b, a
E181    03    579    set    rop
E180    06    580    set    rop, (100)
E179    07    581    le    a, (ls), d
E178    2F    582    le    d, a
E177    2A    583    cucl    d, b
E176    23    584    le    d, (ls), b
E175    01    585    rop    d
E174    23    586    le    d, (ls), b
E173    01    587    rop    d
E172    06    588    rop    d
E171    06    589    rop    d
E170    06    590    rop    d
E169    06    591    rop    d
E168    06    592    ret
E167    06    593

|----------------------------------------------------------------------------------------------------------------------------------|
E166    3E    594    cmea    le    a, b, d2
E165    00    595    rop    b, c
E164    00    596    rop    b, c
E163    4F    597    rop    b, c
E162    01    598    rop    b, c
E161    00    599    rop    b, c
E160    00    600    ret
E159    00    601

|----------------------------------------------------------------------------------------------------------------------------------|
E158    58    602    COMP
E157    0E    603    Input: current string pointer in HL.
E156    06    604    b = parser state.
E155    07    605    Output: b = parser state.
E154    06    606    Global: read memo.
E153    06    607    Global: written: memo.
E152    06    608    Procedures: call (1).
E151    06    609    Procedures: called by: endint
E150    06    610    Description: this routine recognizes a comma. If state is 4B, then comma is skipped and state becomes 61. Otherwise
E149    06    611    state becomes 61_error.
E148    06    612

|----------------------------------------------------------------------------------------------------------------------------------|
E147    06    613    COMP
E146    00    614    Output: b = parser state.
E145    06    615    Global: read memo.
E144    06    616    Global: written: memo.
E143    06    617    Procedures: call (1).
E142    06    618    Procedures: called by: endint
E141    06    619    Description: this routine recognizes a space. If state is 60, then the space is skipped and state remains 60. Otherwise
E140    06    620    state becomes 64_error.
E139    06    621

|----------------------------------------------------------------------------------------------------------------------------------|
E138    06    622    COMP
E137    00    623    Output: b = parser state.
E136    00    624    Output: b = parser state.
E135    00    625    Input: current string pointer in HL.
E134    00    626    b = parser state.
E133    00    627    Output: b = parser state.
E132    00    628    Global: read memo.
E131    00    629    Global: written: memo.
E130    00    630    Procedures: call (1).
E129    00    631    Procedures: called by: endint
E128    00    632    Description: this routine recognizes a space. If state is 66, then the space is skipped and state remains 66. Otherwise
E127    00    633    state becomes 66_error.
E126    00    634

|----------------------------------------------------------------------------------------------------------------------------------|
E125    00    635    COMP
E124    00    636    Output: b = parser state.
E123    00    637    Output: b = parser state.
E122    00    638    Input: current string pointer in HL.
E121    00    639    b = parser state.
E120    00    640    Output: b = parser state.
E119    00    641    Global: read memo.
E118    00    642    Global: written: memo.
E117    00    643    Procedures: call (1).
E116    00    644    Procedures: called by: endint
E115    00    645    Description: this routine recognizes a space. If state is 6F, then the space is skipped and state remains 6F. Otherwise
E114    00    646    state becomes 6F_error.
E113    00    647

|----------------------------------------------------------------------------------------------------------------------------------|
E112    00    648    COMP
E111    00    649    Output: b = parser state.
E110    00    649    Output: b = parser state.
E109    00    649    Input: current string pointer in HL.
E108    00    649    b = parser state.
E107    00    649    Output: b = parser state.
E106    00    649    Global: read memo.
E105    00    649    Global: written: memo.
E104    00    649    Procedures: call (1).
E103    00    649    Procedures: called by: endint
E102    00    649    Description: this routine recognizes a space. If state is 75, then the space is skipped and state remains 75. Otherwise
E101    00    649    state becomes 75_error.
E100    00    649

|----------------------------------------------------------------------------------------------------------------------------------|
E99    00    649    COMP
E98    00    649    Output: b = parser state.
E97    00    649    Output: b = parser state.
E96    00    649    Input: current string pointer in HL.
E95    00    649    b = parser state.
E94    00    649    Output: b = parser state.
E93    00    649    Global: read memo.
E92    00    649    Global: written: memo.
E91    00    649    Procedures: call (1).
E90    00    649    Procedures: called by: endint
E89    00    649    Description: this routine recognizes a space. If state is 7B, then the space is skipped and state remains 7B. Otherwise
E88    00    649
include sprites\t

// Global constants

// Global variables

// Data structures

// Parameter block

// Sprite data offsets

// Input

// Output

// Description:

// Interface routine puts the values of the sprite in the
// proper block into the appropriate registers. The sprite
// exists in the stack. When the service returns, the
// status code is checked and, if an error occurred, an
// appropriate report is printed.
description: This is the main routine of this module. It translates the values in the parse block into calls to sprite services. The interface to the sprite services are described in a table. For each service the is an entry containing the correct number of parameters, the address of the interface routine appropriate to that number of parameters, and the address of the service. The number of parse is the table entry is checked against the value in the parse block. If they do not agree report "G" is produced.

If they do agree, the interface routine indicated in the table entry is called. If the part of the status code in C is non-zero, then an error occurs in executing the sprite service, and a appropriate report is produced.

+-----------------------------------------------------------------+
| 138  | i_f_handler  | 1a  | 16 (svc_code) | I check num.parse against the | 1 value in i_f_table |
| 133  |              | 1d  | b  | a  | 0  | 1e  | i = $0 | svc_code |
| 132  |              | 1d  | c  | a  | 0  | 1f  | i = HI | i_f_table |
| 131  |              | 1d  | d  | 0  | 1g  | i = dl | i_f_table |
| 130  |              | 1e  | e  | (num.parse) | I add to i_f_table |
| 129  |              | 1f  | f  | (HI) | I check num.parse correct, then jump |
| 128  | ret          | 1g  | g  | 0  | 1h  | I/ f routine addr |
| 127  |              | 1i  | h  | (HI) | I/ f routine addr |
| 126  |              | 1j  | j  | (HI) | I/ f routine addr |
| 125  |              | 1k  | k  | (HI) | I/ f routine addr |
| 124  |              | 1l  | l  | (HI) | I/ f routine addr |
| 123  |              | 1m  | m  | (HI) | I/ f routine addr |
| 122  |              | 1n  | n  | (HI) | I/ f routine addr |
| 121  |              | 1o  | o  | (HI) | I/ f routine addr |
| 120  |              | 1p  | p  | (HI) | I/ f routine addr |
| 119  |              | 1q  | q  | (HI) | I/ f routine addr |
| 118  |              | 1r  | r  | (HI) | I/ f routine addr |
| 117  |              | 1s  | s  | (HI) | I/ f routine addr |
| 116  |              | 1t  | t  | (HI) | I/ f routine addr |
| 115  |              | 1u  | u  | (HI) | I/ f routine addr |
| 114  |              | 1v  | v  | (HI) | I/ f routine addr |
| 113  |              | 1w  | w  | (HI) | I/ f routine addr |
| 112  |              | 1x  | x  | (HI) | I/ f routine addr |
| 111  |              | 1y  | y  | (HI) | I/ f routine addr |
| 110  |              | 1z  | z  | (HI) | I/ f routine addr |
+-----------------------------------------------------------------+
This table contains the interface definitions for the sprite services. The first item is the number of parameters to be passed to the service. The second item is the interface routine for doing this. The third item is the address of the service.

<table>
<thead>
<tr>
<th>Service Code</th>
<th>Parameters</th>
<th>Interface</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>S105</td>
<td>1</td>
<td>Lf_table</td>
<td>00</td>
</tr>
<tr>
<td>S106</td>
<td></td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>S107</td>
<td></td>
<td></td>
<td>02</td>
</tr>
<tr>
<td>S108</td>
<td></td>
<td></td>
<td>03</td>
</tr>
<tr>
<td>S109</td>
<td></td>
<td></td>
<td>04</td>
</tr>
<tr>
<td>S110</td>
<td></td>
<td></td>
<td>05</td>
</tr>
<tr>
<td>S111</td>
<td></td>
<td></td>
<td>06</td>
</tr>
<tr>
<td>S112</td>
<td></td>
<td></td>
<td>07</td>
</tr>
<tr>
<td>S113</td>
<td></td>
<td></td>
<td>08</td>
</tr>
<tr>
<td>S114</td>
<td></td>
<td></td>
<td>09</td>
</tr>
<tr>
<td>S115</td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>S116</td>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>S117</td>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>S118</td>
<td></td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>S119</td>
<td></td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>S120</td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>S121</td>
<td></td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>S122</td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>S123</td>
<td></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>S124</td>
<td></td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>S125</td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>S126</td>
<td></td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>S127</td>
<td></td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>S128</td>
<td></td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>S129</td>
<td></td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>S130</td>
<td></td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>S131</td>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>S132</td>
<td></td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>S133</td>
<td></td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>S134</td>
<td></td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>S135</td>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>S136</td>
<td></td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>S137</td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>S138</td>
<td></td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>S139</td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>S140</td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>S141</td>
<td></td>
<td></td>
<td>36</td>
</tr>
<tr>
<td>S142</td>
<td></td>
<td></td>
<td>37</td>
</tr>
<tr>
<td>S143</td>
<td></td>
<td></td>
<td>38</td>
</tr>
<tr>
<td>S144</td>
<td></td>
<td></td>
<td>39</td>
</tr>
<tr>
<td>S145</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>S146</td>
<td></td>
<td></td>
<td>41</td>
</tr>
<tr>
<td>S147</td>
<td></td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>S148</td>
<td></td>
<td></td>
<td>43</td>
</tr>
<tr>
<td>S149</td>
<td></td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>S150</td>
<td></td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>S151</td>
<td></td>
<td></td>
<td>46</td>
</tr>
<tr>
<td>S152</td>
<td></td>
<td></td>
<td>47</td>
</tr>
<tr>
<td>S153</td>
<td></td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>S154</td>
<td></td>
<td></td>
<td>49</td>
</tr>
<tr>
<td>S155</td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>S156</td>
<td></td>
<td></td>
<td>51</td>
</tr>
<tr>
<td>S157</td>
<td></td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>S158</td>
<td></td>
<td></td>
<td>53</td>
</tr>
<tr>
<td>S159</td>
<td></td>
<td></td>
<td>54</td>
</tr>
<tr>
<td>S160</td>
<td></td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>S161</td>
<td></td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>S162</td>
<td></td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>S163</td>
<td></td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>S164</td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>S165</td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>S166</td>
<td></td>
<td></td>
<td>61</td>
</tr>
<tr>
<td>S167</td>
<td></td>
<td></td>
<td>62</td>
</tr>
<tr>
<td>S168</td>
<td></td>
<td></td>
<td>63</td>
</tr>
<tr>
<td>S169</td>
<td></td>
<td></td>
<td>64</td>
</tr>
<tr>
<td>S170</td>
<td></td>
<td></td>
<td>65</td>
</tr>
<tr>
<td>S171</td>
<td></td>
<td></td>
<td>66</td>
</tr>
<tr>
<td>S172</td>
<td></td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>S173</td>
<td></td>
<td></td>
<td>68</td>
</tr>
<tr>
<td>S174</td>
<td></td>
<td></td>
<td>69</td>
</tr>
<tr>
<td>S175</td>
<td></td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>S176</td>
<td></td>
<td></td>
<td>71</td>
</tr>
</tbody>
</table>

Description: this is the interface routine for sprite services with one parameter.
L,F2  14  a, (in)
     14  d, (in+2)
     ret

L,F3  14  a, (in)
     14  d, (in+2)
     14  d, (in+6)
     ret

L,F4  14  a, (in)
     14  b, (in+2)
     14  c, (in+4)
     d, (in+6)
     ret

L,F5  14  a, (in)
     14  b, (in+2)
     14  c, (in+4)
     d, (in+6)
     e, (in+8)
     ret
Global variables

sprite_eew         sprite_eew         sprite_eew
flags_eew         flags_eew         flags_eew
max_sprites       max_sprites       max_sprites
screen_nt         screen_nt         screen_nt
aflags_eew        aflags_eew        aflags_eew
backcolor_eew     backcolor_eew     backcolor_eew
pare_eew          pare_eew          pare_eew
scratchhead_eew   scratchhead_eew   scratchhead_eew

Parameter block

pix_code_eew      pix_code_eew      pix_code_eew
max_pare_eew      max_pare_eew      max_pare_eew
pare0_eew         pare0_eew         pare0_eew
pare1_eew         pare1_eew         pare1_eew
pare2_eew         pare2_eew         pare2_eew
pare3_eew         pare3_eew         pare3_eew
pare4_eew         pare4_eew         pare4_eew

sprite data offsets

00000000

init_sprites

 procedures 1  I_P_HANDLER

 Description:  this routines allocates memory for sprite data.  (max_sprites)

 in:  # number-sprites, the number of sprites for which memory is
      to be allocated.

 out:  BC = 0000 to Cc
       0051 = not enough memory

 Global area:

 variables:
      screen_nt

 Procedure Call:

 I init_sprites 1  I_P_HANDLER

 push bc

 hl, screen_nt

 if (HL) max_sprites

 push bc

 hl, flags

 if (HL) 0

 iclear global flags

 push bc

 (max_sprites), 8

 iclear, where the sprite data should be

 push bc

 bc = #number-sprites

 push bc

 hl, pare0

 push bc

 hl, pare1

 push bc

 hl, pare2

 push bc

 hl, pare3

 push bc

 hl, pare4

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0

 push bc

 bc, 0
The document contains code snippets related to sprite handling in a computer program. The code includes procedures for creating sprites, initializing sprite records, and handling sprite properties. The comments within the code are in multiple languages, indicating that the code is written in a multilingual programming environment. The code is structured in a way that suggests it is part of a larger program dealing with graphical or game-related operations.
check_id push hi
push hl
ld n, sprite_id
ld hl, max_sprites
iml (hl)
pop hl
ret
check_id out of range

set_outp push hl
ld n, hl, flags
pop a
jr in off
ld r, outpflg (nl) if n = 0, then jump
jr as start
jr as exit
off ret
as exit push hl
ld a, 0
ret

display_char push hl
pop bc
push bc
call SET_PRINT_POS
ld a, $20
call WRITE_CHAR
ld hl, $0000
add hl, bc
ld bc, hl
display_char push hl
**VIsible**

- Input: & = row
- C = column

- Output: CP set if SC is a legal screen location, otherwise CP is reset

- Globals: screen.ht

- Procedures: none.

- Procedures Called by: put.sprite

**Put sprite**

- Input: & = sprite-id
- D = row
- E = column

- Output: SC = 0000 - ok, nothing overwritten
- SC = 0100 - ok, something overwritten
- = 0099 - illegal sprite-id

- Globals: sprite data structure

- Procedures Called: check.id

- Procedures Called by: new.sprite

**Description:** This routine writes the specified sprite to the screen.

- If overwrite mode is set, then the specified row and column are erased to fit onto the screen size.

- If overwrite mode is not set, then only the portion of the sprite which is visible on the screen will be written.

- Procedures Called: select.sprite
The image contains a block of code, possibly from a programming language, but it is not clearly readable. The text is fragmented and not formatted in a way that is easily interpretable. It seems to be a segment of a program, possibly related to a graphical user interface or some form of data manipulation.

The text is not coherent enough to be transcribed accurately. It contains various lines of code, possibly part of a larger program, but the exact nature of the program cannot be determined from the image alone.
This table contains the destination address for each row for scrolling left:

<table>
<thead>
<tr>
<th>Increment</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x3A94</td>
<td>0x3A9B</td>
</tr>
<tr>
<td>0x3A9B</td>
<td>0x3A9C</td>
</tr>
<tr>
<td>0x3A9C</td>
<td>0x3A9D</td>
</tr>
<tr>
<td>0x3A9D</td>
<td>0x3A9E</td>
</tr>
<tr>
<td>0x3A9E</td>
<td>0x3A9F</td>
</tr>
<tr>
<td>0x3A9F</td>
<td>0x3A00</td>
</tr>
</tbody>
</table>

This table contains the destination address for each row for scrolling right:

<table>
<thead>
<tr>
<th>Increment</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0F95</td>
<td>0x0F96</td>
</tr>
<tr>
<td>0x0F96</td>
<td>0x0F97</td>
</tr>
<tr>
<td>0x0F97</td>
<td>0x0F98</td>
</tr>
<tr>
<td>0x0F98</td>
<td>0x0F99</td>
</tr>
<tr>
<td>0x0F99</td>
<td>0x0F9A</td>
</tr>
<tr>
<td>0x0F9A</td>
<td>0x0F9B</td>
</tr>
<tr>
<td>0x0F9B</td>
<td>0x0F9C</td>
</tr>
<tr>
<td>0x0F9C</td>
<td>0x0F9D</td>
</tr>
<tr>
<td>0x0F9D</td>
<td>0x0F9E</td>
</tr>
<tr>
<td>0x0F9E</td>
<td>0x0F9F</td>
</tr>
<tr>
<td>0x0F9F</td>
<td>0x0FA0</td>
</tr>
</tbody>
</table>

This table contains the attribute for each column:

<table>
<thead>
<tr>
<th>Column</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column 1</td>
<td>Red (0x0)</td>
</tr>
<tr>
<td>Column 2</td>
<td>Green (0x1)</td>
</tr>
<tr>
<td>Column 3</td>
<td>Blue (0x2)</td>
</tr>
<tr>
<td>Column 4</td>
<td>White (0x3)</td>
</tr>
<tr>
<td>Column 5</td>
<td>Black (0x4)</td>
</tr>
</tbody>
</table>

The function clear_first_column clears the first column of the display.
 ennesh

} call celsius I calculate position
} call stpen I store updated position and return
ret

} Global: float

} Global: float

} Description: This routine calculates the position in the display file corresponding to the specified row and column position.

} call loc
} get display file area.
i

} Label

} Inputs: B = row in internal format
} Outputs: ML = address in display file of start of row B.

} Global: float

} Global: float

} Procedures Called: celsius

} Description: This routine calculates the address of the start of row B in the display file. This is the first byte of the first scan line.

} get display file address


sten 1 store cursor position

lesen 1 load cursor position

update 1 update attribute byte for character

update2 1 update attribute byte for character

jump 1 just printed

 moth 1 any scan of char.in of

0907
0907 DE 03 0155
0908 11 0077
0909 C9

0907: sten

1 store cursor position

0912:

lesen

1 load cursor position

update

update2

jump

moth

0907: sten

1 store cursor position

0912:

lesen

1 load cursor position

update

update2

jump

moth

0907: sten

1 store cursor position

0912:

lesen

1 load cursor position

update

update2

jump

moth
616 |CALCATT
618 |Input PL = address of any scan of a character position
620 |Output NL = address of attribute byte
622 |Globals Need none.
624 |Globals Written none.
626 |Procedures Called none.
628 |Description: This routine calculates the address of the attribute byte
630 |corresponding to the character position identified with
632 |NL.
634 |calcatt
636 |in a h 1 word byte of address
638 |carriage return
640 |end 16h
642 |or $09
644 |PL 16h 1 test which of
646 |JR 80h
648 |AR 16h
650 |call 16h 1 cif
652 |ret
654 |
656 |LOATT
658 |Input: none.
660 |Output: none.
662 |Globals Need: ATTR_P
664 |radix
666 |Globals Written: attr
668 |attack
670 |Output: none.
672 |Procedures Called: none.
674 |Description: This routine sets the values of the variables used by this
676 |module for manipulating attributes with values from the
678 |variables used by the name DC.
680 |
682 |letstr
684 |push of 1 load internal attribute variables
686 |pop of 1 free system variables
688 |
690 |
692 |
694 |
696 |
698 |
700 |
702 |
704 |
706 |
708 |
710 |
712 |
714 |
716 |
718 |
720 |
722 |
724 |
726 |
728 |
730 |
732 |
734 |
736 |
738 |
740 |
742 |
744 |
746 |
748 |
750 |
752 |
754 |
756 |
758 |
760 |
762 |
764 |
766 |
768 |
770 |
772 |
774 |
776 |
778 |
780 |
782 |
784 |
786 |
788 |
790 |
792 |
794 |
796 |
798 |
800 |
802 |
804 |
806 |
808 |
810 |
812 |
814 |
816 |
818 |
820 |
822 |
824 |
826 |
828 |
830 |
832 |
834 |
836 |
838 |
840 |
842 |
844 |
846 |
848 |
850 |
852 |
854 |
856 |
858 |
860 |
862 |
864 |
866 |
868 |
870 |
872 |
874 |
876 |
878 |
880 |
882 |
884 |
886 |
888 |
890 |
892 |
894 |
896 |
898 |graphic
900 |
902 |
904 |
906 |
908 |
910 |
912 |
914 |
916 |
918 |
920 |
922 |
924 |
926 |
928 |
930 |
932 |
934 |
936 |
938 |
940 |
942 |
944 |
946 |
948 |
950 |
952 |
954 |
956 |
958 |
960 |
962 |
964 |
966 |
968 |
970 |
972 |
974 |
976 |
978 |
980 |
982 |
984 |
986 |
988 |
990 |
992 |286
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Graphics</td>
</tr>
<tr>
<td>01</td>
<td>Graphics</td>
</tr>
<tr>
<td>02</td>
<td>Graphics</td>
</tr>
<tr>
<td>03</td>
<td>Graphics</td>
</tr>
<tr>
<td>04</td>
<td>Graphics</td>
</tr>
<tr>
<td>05</td>
<td>Graphics</td>
</tr>
<tr>
<td>06</td>
<td>Graphics</td>
</tr>
<tr>
<td>07</td>
<td>Graphics</td>
</tr>
<tr>
<td>08</td>
<td>Graphics</td>
</tr>
<tr>
<td>09</td>
<td>Graphics</td>
</tr>
<tr>
<td>10</td>
<td>Graphics</td>
</tr>
<tr>
<td>11</td>
<td>Graphics</td>
</tr>
<tr>
<td>12</td>
<td>Graphics</td>
</tr>
<tr>
<td>13</td>
<td>Graphics</td>
</tr>
<tr>
<td>14</td>
<td>Graphics</td>
</tr>
<tr>
<td>15</td>
<td>Graphics</td>
</tr>
<tr>
<td>16</td>
<td>Graphics</td>
</tr>
<tr>
<td>17</td>
<td>Graphics</td>
</tr>
<tr>
<td>18</td>
<td>Graphics</td>
</tr>
<tr>
<td>19</td>
<td>Graphics</td>
</tr>
<tr>
<td>20</td>
<td>Graphics</td>
</tr>
<tr>
<td>21</td>
<td>Graphics</td>
</tr>
<tr>
<td>22</td>
<td>Graphics</td>
</tr>
<tr>
<td>23</td>
<td>Graphics</td>
</tr>
<tr>
<td>24</td>
<td>Graphics</td>
</tr>
<tr>
<td>25</td>
<td>Graphics</td>
</tr>
<tr>
<td>26</td>
<td>Graphics</td>
</tr>
<tr>
<td>27</td>
<td>Graphics</td>
</tr>
<tr>
<td>28</td>
<td>Graphics</td>
</tr>
<tr>
<td>29</td>
<td>Graphics</td>
</tr>
<tr>
<td>30</td>
<td>Graphics</td>
</tr>
<tr>
<td>31</td>
<td>Graphics</td>
</tr>
<tr>
<td>32</td>
<td>Graphics</td>
</tr>
<tr>
<td>33</td>
<td>Graphics</td>
</tr>
<tr>
<td>34</td>
<td>Graphics</td>
</tr>
<tr>
<td>35</td>
<td>Graphics</td>
</tr>
<tr>
<td>36</td>
<td>Graphics</td>
</tr>
<tr>
<td>37</td>
<td>Graphics</td>
</tr>
<tr>
<td>38</td>
<td>Graphics</td>
</tr>
<tr>
<td>39</td>
<td>Graphics</td>
</tr>
<tr>
<td>40</td>
<td>Graphics</td>
</tr>
<tr>
<td>41</td>
<td>Graphics</td>
</tr>
<tr>
<td>42</td>
<td>Graphics</td>
</tr>
<tr>
<td>43</td>
<td>Graphics</td>
</tr>
<tr>
<td>44</td>
<td>Graphics</td>
</tr>
<tr>
<td>45</td>
<td>Graphics</td>
</tr>
<tr>
<td>46</td>
<td>Graphics</td>
</tr>
<tr>
<td>47</td>
<td>Graphics</td>
</tr>
<tr>
<td>48</td>
<td>Graphics</td>
</tr>
<tr>
<td>49</td>
<td>Graphics</td>
</tr>
<tr>
<td>50</td>
<td>Graphics</td>
</tr>
<tr>
<td>51</td>
<td>Graphics</td>
</tr>
<tr>
<td>52</td>
<td>Graphics</td>
</tr>
<tr>
<td>53</td>
<td>Graphics</td>
</tr>
<tr>
<td>54</td>
<td>Graphics</td>
</tr>
<tr>
<td>55</td>
<td>Graphics</td>
</tr>
<tr>
<td>56</td>
<td>Graphics</td>
</tr>
<tr>
<td>57</td>
<td>Graphics</td>
</tr>
<tr>
<td>58</td>
<td>Graphics</td>
</tr>
<tr>
<td>59</td>
<td>Graphics</td>
</tr>
<tr>
<td>60</td>
<td>Graphics</td>
</tr>
<tr>
<td>61</td>
<td>Graphics</td>
</tr>
<tr>
<td>62</td>
<td>Graphics</td>
</tr>
<tr>
<td>63</td>
<td>Graphics</td>
</tr>
<tr>
<td>64</td>
<td>Graphics</td>
</tr>
<tr>
<td>65</td>
<td>Graphics</td>
</tr>
<tr>
<td>66</td>
<td>Graphics</td>
</tr>
<tr>
<td>67</td>
<td>Graphics</td>
</tr>
<tr>
<td>68</td>
<td>Graphics</td>
</tr>
<tr>
<td>69</td>
<td>Graphics</td>
</tr>
<tr>
<td>70</td>
<td>Graphics</td>
</tr>
<tr>
<td>71</td>
<td>Graphics</td>
</tr>
<tr>
<td>72</td>
<td>Graphics</td>
</tr>
<tr>
<td>73</td>
<td>Graphics</td>
</tr>
<tr>
<td>74</td>
<td>Graphics</td>
</tr>
<tr>
<td>75</td>
<td>Graphics</td>
</tr>
<tr>
<td>76</td>
<td>Graphics</td>
</tr>
<tr>
<td>77</td>
<td>Graphics</td>
</tr>
<tr>
<td>78</td>
<td>Graphics</td>
</tr>
<tr>
<td>79</td>
<td>Graphics</td>
</tr>
<tr>
<td>80</td>
<td>Graphics</td>
</tr>
<tr>
<td>81</td>
<td>Graphics</td>
</tr>
<tr>
<td>82</td>
<td>Graphics</td>
</tr>
<tr>
<td>83</td>
<td>Graphics</td>
</tr>
<tr>
<td>84</td>
<td>Graphics</td>
</tr>
<tr>
<td>85</td>
<td>Graphics</td>
</tr>
<tr>
<td>86</td>
<td>Graphics</td>
</tr>
<tr>
<td>87</td>
<td>Graphics</td>
</tr>
<tr>
<td>88</td>
<td>Graphics</td>
</tr>
<tr>
<td>89</td>
<td>Graphics</td>
</tr>
<tr>
<td>90</td>
<td>Graphics</td>
</tr>
<tr>
<td>91</td>
<td>Graphics</td>
</tr>
<tr>
<td>92</td>
<td>Graphics</td>
</tr>
<tr>
<td>93</td>
<td>Graphics</td>
</tr>
<tr>
<td>94</td>
<td>Graphics</td>
</tr>
<tr>
<td>95</td>
<td>Graphics</td>
</tr>
<tr>
<td>96</td>
<td>Graphics</td>
</tr>
<tr>
<td>97</td>
<td>Graphics</td>
</tr>
<tr>
<td>98</td>
<td>Graphics</td>
</tr>
<tr>
<td>99</td>
<td>Graphics</td>
</tr>
<tr>
<td>100</td>
<td>Graphics</td>
</tr>
<tr>
<td>101</td>
<td>Graphics</td>
</tr>
<tr>
<td>102</td>
<td>Graphics</td>
</tr>
<tr>
<td>103</td>
<td>Graphics</td>
</tr>
<tr>
<td>104</td>
<td>Graphics</td>
</tr>
<tr>
<td>105</td>
<td>Graphics</td>
</tr>
<tr>
<td>106</td>
<td>Graphics</td>
</tr>
<tr>
<td>107</td>
<td>Graphics</td>
</tr>
<tr>
<td>108</td>
<td>Graphics</td>
</tr>
<tr>
<td>109</td>
<td>Graphics</td>
</tr>
<tr>
<td>110</td>
<td>Graphics</td>
</tr>
<tr>
<td>111</td>
<td>Graphics</td>
</tr>
<tr>
<td>112</td>
<td>Graphics</td>
</tr>
<tr>
<td>113</td>
<td>Graphics</td>
</tr>
<tr>
<td>114</td>
<td>Graphics</td>
</tr>
<tr>
<td>115</td>
<td>Graphics</td>
</tr>
<tr>
<td>116</td>
<td>Graphics</td>
</tr>
<tr>
<td>117</td>
<td>Graphics</td>
</tr>
<tr>
<td>118</td>
<td>Graphics</td>
</tr>
<tr>
<td>119</td>
<td>Graphics</td>
</tr>
<tr>
<td>120</td>
<td>Graphics</td>
</tr>
<tr>
<td>121</td>
<td>Graphics</td>
</tr>
<tr>
<td>122</td>
<td>Graphics</td>
</tr>
<tr>
<td>123</td>
<td>Graphics</td>
</tr>
<tr>
<td>124</td>
<td>Graphics</td>
</tr>
<tr>
<td>125</td>
<td>Graphics</td>
</tr>
<tr>
<td>126</td>
<td>Graphics</td>
</tr>
<tr>
<td>127</td>
<td>Graphics</td>
</tr>
<tr>
<td>128</td>
<td>Graphics</td>
</tr>
<tr>
<td>129</td>
<td>Graphics</td>
</tr>
<tr>
<td>130</td>
<td>Graphics</td>
</tr>
<tr>
<td>131</td>
<td>Graphics</td>
</tr>
<tr>
<td>132</td>
<td>Graphics</td>
</tr>
<tr>
<td>133</td>
<td>Graphics</td>
</tr>
<tr>
<td>134</td>
<td>Graphics</td>
</tr>
<tr>
<td>135</td>
<td>Graphics</td>
</tr>
<tr>
<td>136</td>
<td>Graphics</td>
</tr>
<tr>
<td>137</td>
<td>Graphics</td>
</tr>
<tr>
<td>138</td>
<td>Graphics</td>
</tr>
<tr>
<td>139</td>
<td>Graphics</td>
</tr>
<tr>
<td>140</td>
<td>Graphics</td>
</tr>
<tr>
<td>141</td>
<td>Graphics</td>
</tr>
<tr>
<td>142</td>
<td>Graphics</td>
</tr>
<tr>
<td>143</td>
<td>Graphics</td>
</tr>
<tr>
<td>144</td>
<td>Graphics</td>
</tr>
<tr>
<td>145</td>
<td>Graphics</td>
</tr>
<tr>
<td>146</td>
<td>Graphics</td>
</tr>
<tr>
<td>147</td>
<td>Graphics</td>
</tr>
<tr>
<td>148</td>
<td>Graphics</td>
</tr>
<tr>
<td>149</td>
<td>Graphics</td>
</tr>
<tr>
<td>150</td>
<td>Graphics</td>
</tr>
<tr>
<td>151</td>
<td>Graphics</td>
</tr>
<tr>
<td>152</td>
<td>Graphics</td>
</tr>
<tr>
<td>153</td>
<td>Graphics</td>
</tr>
<tr>
<td>154</td>
<td>Graphics</td>
</tr>
<tr>
<td>155</td>
<td>Graphics</td>
</tr>
<tr>
<td>156</td>
<td>Graphics</td>
</tr>
<tr>
<td>157</td>
<td>Graphics</td>
</tr>
<tr>
<td>158</td>
<td>Graphics</td>
</tr>
<tr>
<td>159</td>
<td>Graphics</td>
</tr>
<tr>
<td>160</td>
<td>Graphics</td>
</tr>
<tr>
<td>161</td>
<td>Graphics</td>
</tr>
<tr>
<td>162</td>
<td>Graphics</td>
</tr>
<tr>
<td>163</td>
<td>Graphics</td>
</tr>
</tbody>
</table>
APPENDIX D

TS2068 PCB Assembly and Schematic Diagram

The following Appendix contains the PCB Assembly Drawing, the PCB Parts List, and PCB Schematic Diagram (a "fold-out" page located just inside the back cover). The Table below contains some corrections to the Schematic Diagram.

***TS2068 PCB Schematic Diagram Corrections***

Page 34 of the Technical Manual shows pin 9 of the joystick ports grounded as it should be. The traces were left off the TS2068 PCB.

VR1: U3-33 goes to VR1/Q5
Q4: Connect base to R55/R54
Solder dots on horizontal lines below keyboard:
   U12-4 to U3-65 (WR)
       U12-5 to U3-66 (MREQ)
U5: U5-2 to U3-38 (A7R not A7RB)
P1: P1-4B +15V (not -15V)
U21:

![Schematic Diagram](image-url)
### APPENDIX D

TS2068 PARTS LIST

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>COMPONENT DESIGNATION</th>
<th>QTY PER ASSY</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P.C.B. (Fabrication and Artwork)</td>
<td></td>
<td></td>
<td>REV 3A</td>
</tr>
<tr>
<td>CAP. 0.1 uf, Ceramic, Axial</td>
<td>C2,7,9,16,24,30</td>
<td>23</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td>31,34,35,37,39,43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEMP Z5U</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>44,48,49,50,51,52</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>53,54,55,56,57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAP. 0.01 uf, Ceramic, Axial</td>
<td>C11,12,14,33,61</td>
<td>8</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td>62,68,69</td>
<td></td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 0.001 uf, Ceramic, Axial</td>
<td>C8,45,46,47</td>
<td>4</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 0.047 uf, Ceramic, Axial</td>
<td>C10,15,74,75</td>
<td>4</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 20pf Ceramic Axial</td>
<td>C23</td>
<td>1</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 39pf Ceramic Axial</td>
<td>C20</td>
<td>1</td>
<td>NPO</td>
</tr>
<tr>
<td>CAP. 43pf Ceramic Axial</td>
<td>C19</td>
<td>1</td>
<td>NPO</td>
</tr>
<tr>
<td>CAP. 56pf Ceramic Axial</td>
<td>C25</td>
<td>1</td>
<td>NPO</td>
</tr>
<tr>
<td>CAP. 75pf Ceramic Axial</td>
<td>C32</td>
<td>1</td>
<td>NPO</td>
</tr>
<tr>
<td>CAP. 120pf Ceramic Disc</td>
<td>C59,63,64,65,72</td>
<td>6</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td>73</td>
<td></td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 470uf, 25V AL Electrolytic Axial</td>
<td>C3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CAP. 1 uf, 16V MIN AL Electrolytic Axial</td>
<td>C21</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CAP. 47 uf, 16V MIN AL Electrolytic Axial or Radial</td>
<td>C41</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CAP. 1000 uf, 12V MIN AL Electrolytic Axial</td>
<td>C40</td>
<td>1</td>
<td>LOW ESR</td>
</tr>
<tr>
<td>CAP. 1000 pf, 50V MIN FILM MYLAR</td>
<td>C36</td>
<td>1</td>
<td>+/- 20%</td>
</tr>
<tr>
<td>CAP. 100 uf, 10V MIN AL Electrolytic Axial</td>
<td>C58,67</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CAP. 6-50 pf, TRIMMER</td>
<td>C5,18</td>
<td>2</td>
<td>NPO</td>
</tr>
<tr>
<td>CAP. 0.47 uf Ceramic Axial</td>
<td>C60</td>
<td>1</td>
<td>-20 +80% or GMV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 33 uf TANTALUM</td>
<td>C71</td>
<td>1</td>
<td>+/- 20%</td>
</tr>
</tbody>
</table>
### APPENDIX D

**TS2068 PARTS LIST**

(continued)

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>COMPONENT DESIGNATION</th>
<th>QTY</th>
<th>QTY PER ASSY</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAP. 68 pf Ceramic Axial</td>
<td>C70</td>
<td>1</td>
<td>-20 +80% or GMV</td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 24 pf Ceramic Axial</td>
<td>C29,27</td>
<td>2</td>
<td>-20 +80% or GMV</td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>CAP. 47 pf Ceramic Axial</td>
<td>C28</td>
<td>1</td>
<td>-20 +80% or GMV</td>
<td>TEMP Z5U</td>
</tr>
<tr>
<td>RES. 300 OHM, 1/4W, +/-5%, CF</td>
<td>R23</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 200 OHM, 1/4W, +/-5%, CF</td>
<td>R19,50,54,55</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 100 OHM, 1/4W, +/-5%, CF</td>
<td>R58</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 240 OHM, 1/4W, +/-5%, CF</td>
<td>R24,28,56,57</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 68 OHM, 1/4W, +/-5%, CF</td>
<td>R2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 680 OHM, 1/4W, +/-5%, CF</td>
<td>R13,68</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 390 OHM, 1/4W, +/-5%, CF</td>
<td>R74</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 1K OHM, 1/4W, +/-5%, CF</td>
<td>R11,33,34,35,36</td>
<td>8</td>
<td>38,42,62</td>
<td></td>
</tr>
<tr>
<td>RES. 1.5K OHM, 1/4W, +/-5%, CF</td>
<td>R41</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 1.8K OHM, 1/4W, +/-5%, CF</td>
<td>R29,30</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 620 OHM, 1/4W, +/-5%, CF</td>
<td>R52</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 2K OHM, 1/4W, +/-5%, CF</td>
<td>R22</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 3K OHM, 1/4W, +/-5%, CF</td>
<td>R32</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 2.2K OHM, 1/4W, +/-5%, CF</td>
<td>R61</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 110 OHM, 1/4W, +/-5%, CF</td>
<td>R53</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 510 OHM, 1/4W, +/-5%, CF</td>
<td>R69</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 5K1 OHM, 1/4W, +/-5%, CF</td>
<td>R31</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 10K OHM, 1/4W, +/-5%, CF</td>
<td>R16,40,60,70</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 13K OHM, 1/4W, +/-5%, CF</td>
<td>R26,27</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 20K OHM, 1/4W, +/-5%, CF</td>
<td>R44,45</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 62K OHM, 1/4W, +/-5%, CF</td>
<td>R9,73</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 100K OHM, 1/4W, +/-5%, CF</td>
<td>R15,49</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 220K OHM, 1/4W, +/-5%, CF</td>
<td>R43</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 75 OHM, 1/4W, +/-5%, CF</td>
<td>R46,67</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 1.10K OHM 1/4W, +/-1%, MF</td>
<td>R6</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 3.32K OHM 1/4W, +/-1%, MF</td>
<td>R5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 10K OHM, VARIABLE, LINEAR</td>
<td>VR1,2,3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 330 OHM, 0.5W, +/-5%, CF</td>
<td>R4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 56 OHM, 1/4W, +/-5%, CF</td>
<td>R65,71</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES. 0.110 OHM, 3W, +/-5%, CF</td>
<td>R1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wire Wound

| RES. 20 OHM, 1/4W, +/-5%, CF | R63 | 1 | | |
| RES. 82 OHM, 1/4W, +/-5%, CF | R64 | 1 | | |
| RES. 22 OHM, 1/4W, +/-5%, CF | R66 | 1 | | |
| RES. 680K OHM, 1/4W, +/-5%, CF | R14 | 1 | | |
| RES. 47K OHM, 1/4W, +/-5%, CF | R48 | 1 | | |
| RES. 390K OHM, 1/4W, +/-5%, CF | R72 | 1 | | |
| RES. 6.8K OHM, 1/4W, +/-5%, CF | R12 | 1 | | |
### APPENDIX D

#### TS2068 PARTS LIST

(continued)

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>COMPONENT DESIGNATION</th>
<th>QTY PER ASSY</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIODE 1N4148</td>
<td>CR4,5,6,7,8,9,10</td>
<td>1</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>11,12,13,14,15,16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17,18,19,20,21,22</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23,24,25,26,27,28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIODE, Schottky 1N5821 or equivalent</td>
<td>CR1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, UA 78S40 NPC, Switching Regulator</td>
<td>U1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, SCLD</td>
<td>U3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, LM1889N, Video Modulator</td>
<td>U4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, 74LS244N</td>
<td>U5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, TMS4416-15 (150NS) MOS Dynamic RAM</td>
<td>U6,7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IC, UA 78L12 Regulator</td>
<td>U8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, 74LS245</td>
<td>U9,15</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IC, 74LS157N</td>
<td>U10,11</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IC, TMS4416-20 (200NS) MOS Dynamic RAM</td>
<td>U12,13,17,18</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>IC, AY-3-8912, Sound Gen.and I/O Port</td>
<td>U14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, 23128 Mask ROM (16K X 8)</td>
<td>U16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, CPU Z80A</td>
<td>U19</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, 2364 Mask ROM (8K X 8)</td>
<td>U20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IC, 74LS00</td>
<td>U21</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TRAN. PNP D43C1</td>
<td>Q1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TRAN. PNP 2N2907</td>
<td>Q3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TRAN. PNP 2N3904</td>
<td>Q7,8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>TRAN. PNP 2N2222</td>
<td>Q5,4,2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>COMPONENT DESIGNATION</td>
<td>QTY PER ASSY</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>EMI Filter (Bifilter) 2.2mh</td>
<td>L1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Inductor 230 uh</td>
<td>L2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Inductor .33uh Axial</td>
<td>L3,4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Inductor .12uh</td>
<td>L6,7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Crystal Oscillator 14.112 MHz</td>
<td>Y1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Crystal Oscillator 3.579545 MHz</td>
<td>Y2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Switch SPDT, Rocker</td>
<td>SW2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Switch Channel Select, SPDT Slide</td>
<td>SW1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Video Jack Insulation Pad</td>
<td></td>
<td>1</td>
<td>Under J7</td>
</tr>
<tr>
<td>Jack, Right Angle RCA Video Jack</td>
<td>J7</td>
<td>1</td>
<td>Monitor</td>
</tr>
<tr>
<td>Jack, Mini Phone, EAR &amp; MIC</td>
<td>J2,3</td>
<td>2</td>
<td>Tape</td>
</tr>
<tr>
<td>Jack, COAX, DC Power, 2 1/2 MM Pin</td>
<td>J1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Jack, Phono</td>
<td>J8</td>
<td>1</td>
<td>Assembled to Shield, R.F.</td>
</tr>
<tr>
<td>Connector, Cartridge 2 X 18 Pin 0.1&quot; Space</td>
<td>J4</td>
<td>1</td>
<td>Key between Contact 4&amp;6</td>
</tr>
<tr>
<td>Connector, Flex Cable 14 Pin</td>
<td>J9</td>
<td>1</td>
<td>Keyboard</td>
</tr>
<tr>
<td>Connector, Joystick 9-Pin Male (D Type)</td>
<td>J5,6</td>
<td>2</td>
<td>Joysticks</td>
</tr>
<tr>
<td>Shield, R.F. Button</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Shield, R.F. Top</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Heat Sink</td>
<td>HS1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Heat Sink Insulation Pad</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>COMPONENT DESIGNATION</td>
<td>QTY PER ASSY</td>
<td>COMMENTS</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>----------</td>
</tr>
<tr>
<td>Socket, IC, 28 Pin</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Socket, IC, 40 Pin</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Speaker, 45 OHM, Mylar Cone</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Jumper Wire</td>
<td>W1, 2, 50</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Ferrite Bead</td>
<td>L5, 8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>PC Board Assembly, Daughter</td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX E

Expansion Buss Comparison of TS2068, Sinclair Spectrum and ZX81

TS 2068 | SPECTRUM | ZX-81
---|---|---
**BOTTOM** | **TOP** | **BOTTOM** | **TOP** | **BOTTOM** | **TOP**
---|---|---|---|---|---
GND | GND | A14 | A15 | 5V | D1
SPKR/TAPE | 2 | EAR | 3 | A12 | A13 | 9V | A15 | NMI
+15v | 3 | ATR | 4 | 5V | D7 | 9V | 2 | ROMCS
+5v | 4 | D7 | 5v | 3 | D7 | 9V | 4 | ROMCS
N.C. | 6 | N.C. | | | | | | |
PWR GND | 7 | D0 | Ov | 6 | D0 | | | |
PWR GND | 8 | D1 | Ov | 7 | D1 | | | |
CLK | 9 | D2 | CK | 8 | D2 | | | |
A0 | 10 | D6 | A0 | 9 | D6 | | | |
A1 | 11 | D6 | A1 | 10 | D6 | | | |
A2 | 12 | D5 | A2 | 11 | D5 | | | |
A3 | 13 | D4 | A3 | 12 | D4 | | | |
A16B | 14 | INT | ORQGE | 13 | INT | | | |
A14B | 15 | NMI | Ov | 14 | NMI | | | |
A13B | 16 | HALT | Video | 15 | HALT | | | |
A18 | 17 | MREQB | Y | 16 | MREQ | | | |
A16 | 18 | ORQG | V | 17 | ORQG | | | |
A14 | 19 | RD5 | U | 18 | RD | | | |
A9 | 20 | WRB | BUSQ | 19 | WR | | | |
A8 | 21 | BUSAK | RESET | 20 | -5V | | | |
A7 | 22 | WAIT | A7 | 21 | WAIT | | | |
A6 | 23 | BUSRQ | A6 | 22 | +12V | | | |
A5 | 24 | RESET | A5 | 23 | -12V | | | |
A4 | 25 | MT | A4 | 24 | MT | | | |
N.C. | 26 | RFSHB | ROMCS | 25 | RFSH | | | |
RGB Red | 27 | EXROM | BUSAK | 26 | A9 | | | |
RGB Grn | 28 | ROSCS | A9 | 27 | A10 | | | |
RGB Blu | 29 | DE | A8 | 28 | | | | |
10A5 | 30 | 31 | SOUND | | | | | |
VIDEO | 31 | 32 | GND | | | | | |
GND | 32 | GND | | | | | | |
APPENDIX F

August 1985
Bob Orrfelt

TS2068 MODIFICATIONS FOR EPROMS

There are a number of errors in the TS2068 Home ROM and the Extension ROM. The errors can be corrected by using EPROMs. The following modifications are necessary:

Non-component side of the pcb.
0. Remove ROMs.
1. Cut the trace between U20-26 and U20-27
2. Jumper pins 1 to 28 to 27 on each socket.

Component side of pcb.
3. Remove the two zero ohm resistors W1 and W2.
4. Cut the trace just above and to the left of hole A.
5. Add a jumper from hole A to the trace. This connects MREQ to U16 pin 22.
6. Add a jumper from hole C to hole B. This connects ROMCS to U16 pin 20.
7. Use a 27128 (16K) EPROM for U16.
8. Use a 2764 (8K) EPROM for U20.
October 1985
Bob Orrfelt

Proposed TS2068 Home ROM Corrections and Improvements

NMI fix.
868D 2B01 JR Z,0070H

DELETE delay timing.
0351 010100 LD BC,0001H
0354 0B DEC BC
0355 79 LD A,C
0356 80 OR A
0357 20FB JR NZ,0354H
0359 F1 POP AF
035A 18D2 JR 032EH

Optional turn on message.
(Last character add 80H)
11B8 Orrfelt........
1138 ..............
1148 .......

INT -5536 etc. errors.
33F1 F5 PUSH AF
33F2 3C INC A
33F3 83 OR D
33F4 B2 OR D
33F5 C2F435 JP NZ,35E4H
33F8 C3F35 JP 35FH
35E2 181A JR 35FEH
35E4 F1 POP AF
35E5 77 LD (HL),A
35E6 23 INC HL
35E7 73 LD (HL),E
35E8 23 INC HL
35E9 72 LD (HL),D
35EA 2B DEC HL
35EB 2B DEC HL
35EC 2B DEC HL
35ED D1 POP DE
35EE C9 RET
35EF F1 POP AF
35F0 2B DEC HL
35F1 3691 LD (HL),91H
35F3 23 INC HL
35F4 3680 LD (HL),80H
35F6 3C INC A
35F7 18ED JR 35E6H
35F9 00000000 blanks
35FD FF