TS2068 Technical Manual

This is the second edition of the manual published by Time Designs Magazine (now defunct). It is based on the original blue manual released by Timex Computer Corp. shortly before it folded. Aside from a page renumbering and some sections that were added in the second edition, there is not a lot of difference between the two.

This doc was captured using Adobe Acrobat 3.0, the only software I could find that did a half-decent job. There are still numerous errors where Acrobat got confused, but I'm too lazy to fix them.

Alvin 05/10/98
aralbrec@concentric. net

TIMEX SINCLAIR 2068
PERSONAL COLOR COMPUTER

TECHNICAL REFERENCE MANUAL

Prepared by

V. C. Corcoran
and
M. H. Branigin

TIMEX COMPUTER CORPORATION
Waterbury CT 06720

Second Edition Printing
Published Exclusively by:
TIME DESIGNS MAGAZINE CO.
COLTON, OREGON 97017

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.

TARLE OF CONTENTS

TABLE OF CONTENTS

(continued)

TABLE OF CONTENTS

(continued)

LIST OF FIGURES

LIST OF FIGURES
(continued)

LIST OF TABLES

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
RA!! - 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.

FIGURE 1.1-1

TS 2068 SYSTEM BLOCK DIAGRAM

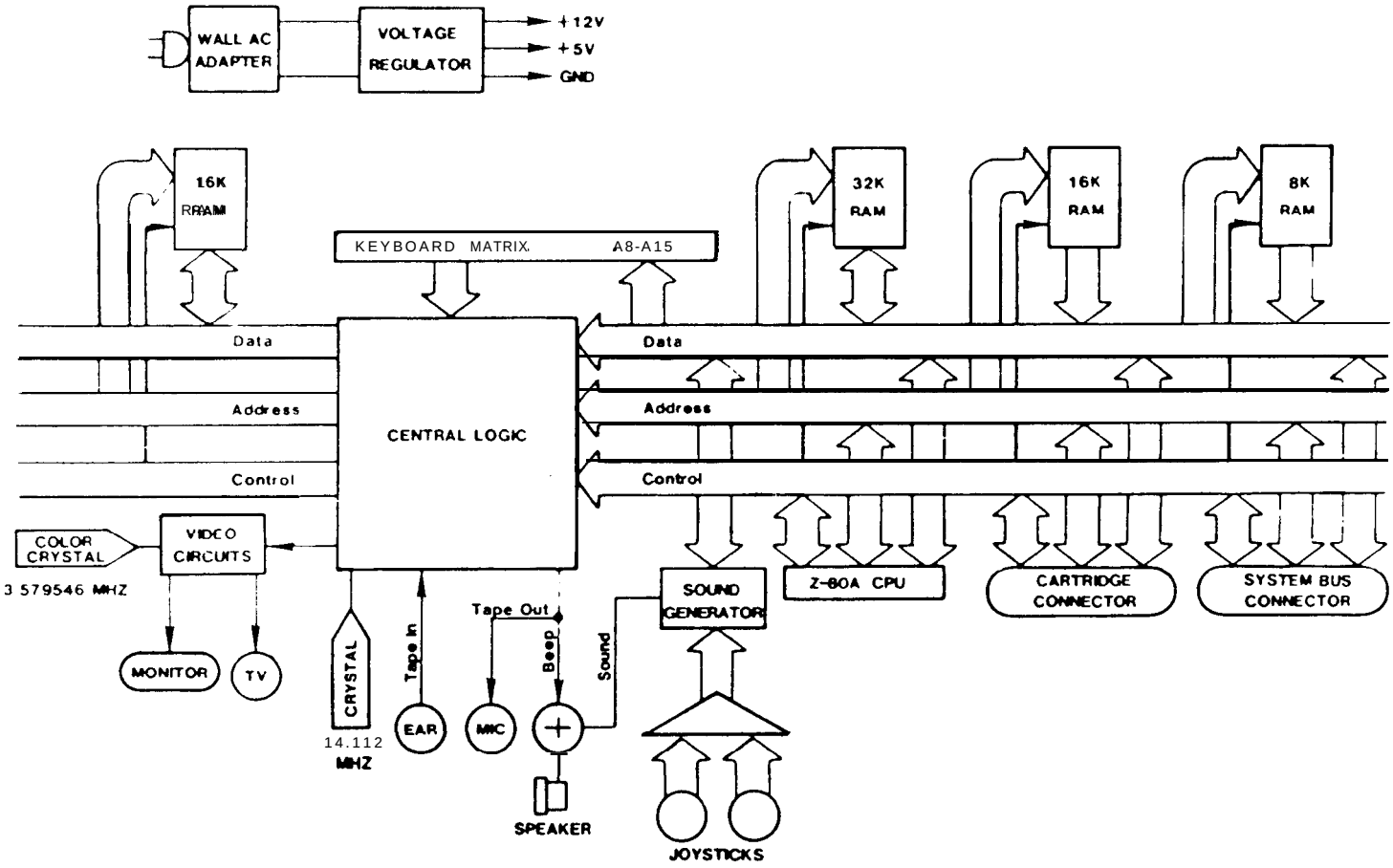
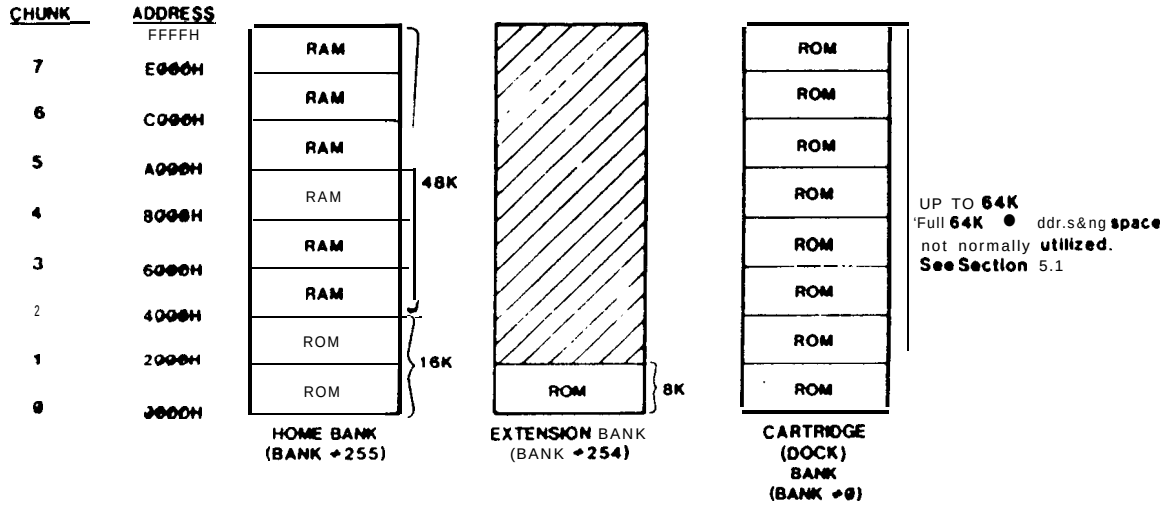


FIGURE 1.1-2

TS 2068 STANDARD MEMORY CONFIGURATION



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

  - o keyboard
  - o video screen
  - o 2040 32-col. dot matrix printer
  - o cassette tape
  - o joysticks
  - o software generated sound (BEEP)
  - o 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)

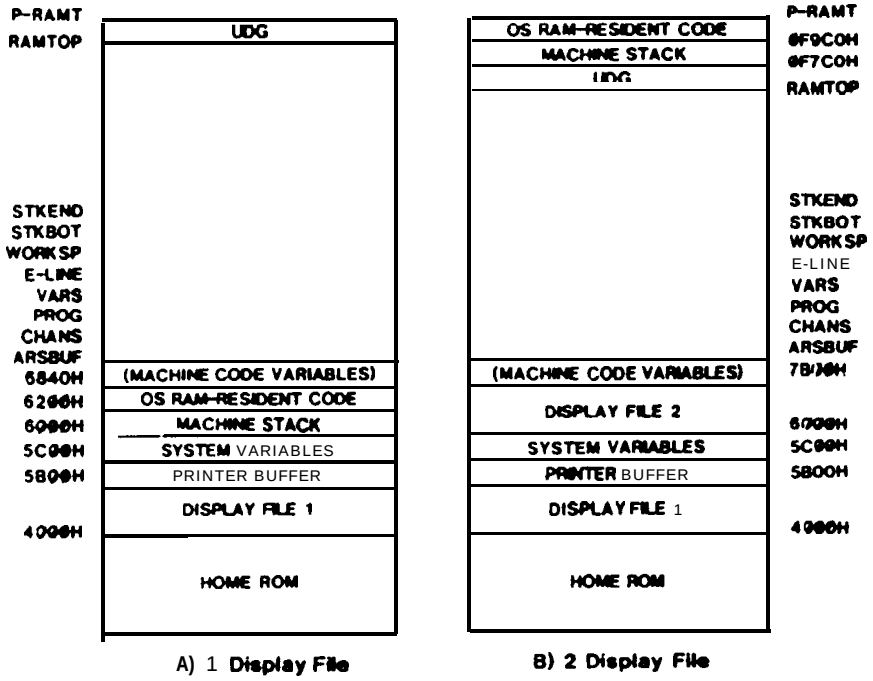
In addition, portions of the Home Bank RAM are used for the machine stack, the BASIC system variables, the Printer Buffer and the Display Files. Figure 1.1-3 shows the standard mapping of the Home Bank RAM and the mapping necessary when the second display file is to be used with the BASIC interpreter 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



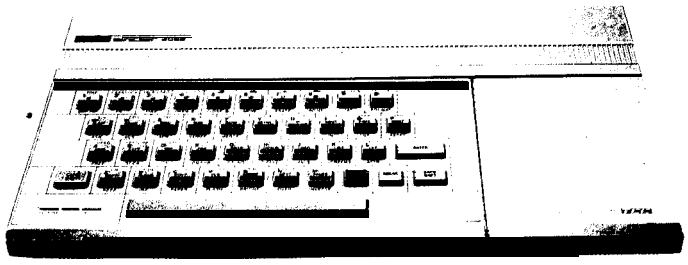
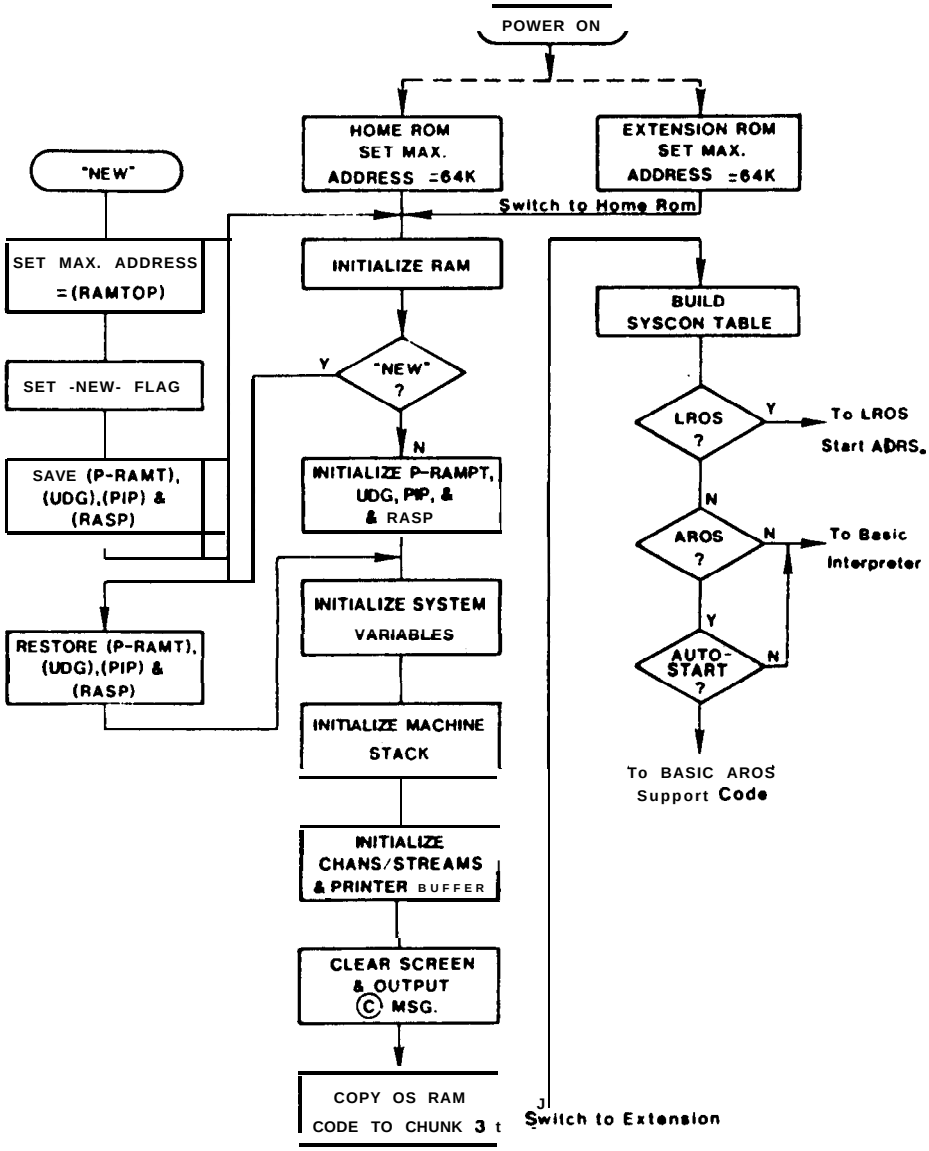


FIGURE 1.1-4

SYSTEM INITIALIZATION



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 pluq 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 280 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

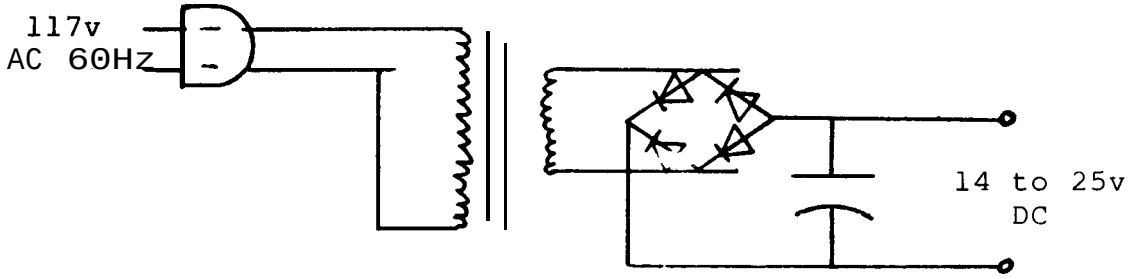
2.1 Description of Major Hardware Functions

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

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

2.1.1 AC Adapter

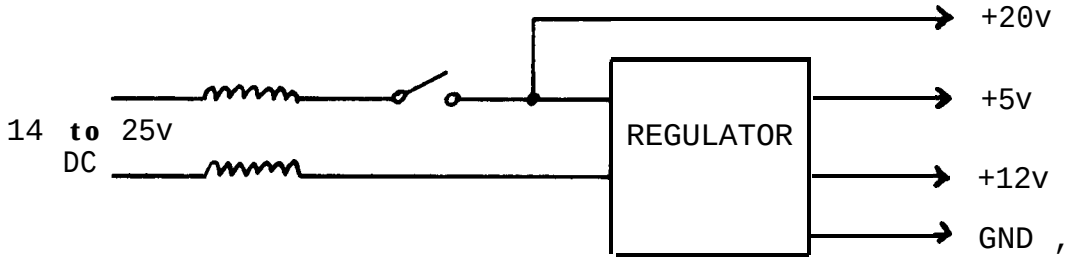
The AC Adapter transforms 117V AC (Nominal) to filtered DC via a step down transformer, full-wave bridge rectifier, and filter capacitor to supply from 14 to 25 volts at 1 amp over the AC voltage variation range of 105 to 130 V AC. Transformer isolation exceeds 1500 volts.



2.1.2 Voltage Regulation

Unregulated DC from the AC Adapter is supplied for regulation through a bi-filar torroidal 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:

SUPPLY	VOLTAGE RANGE	CURRENT RANGE
5V	4.75 - 5.25V	200ma - 1.0 A
12v	11.5 - 12.5V	20ma - 100ma



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

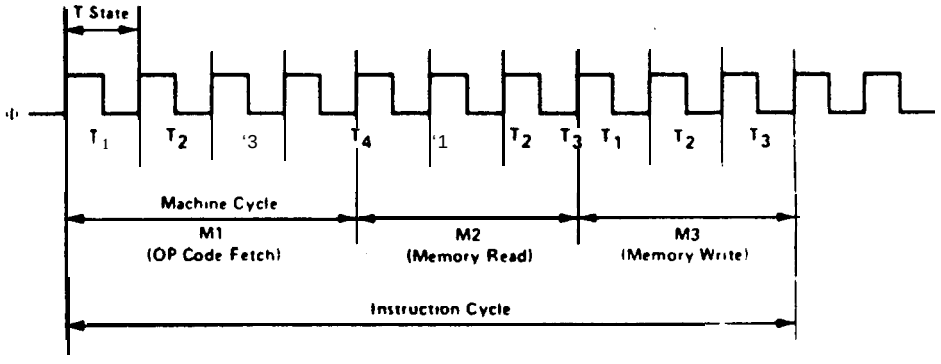
The Z-80 CPU executes instructions by proceeding through a sequence of operations that include:

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



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, DO - 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-l.

2.1.3.4 Op Code Fetch

The timing during an Ml cycle (OP Code Fetch) is shown in Figure 2.1.3-2. At the beginning of the Ml 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 $\overline{RD}$ 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 $\overline{MREQ}$ and $\overline{RD}$ signals used as in the fetch cycle. During a write cycle the $\overline{WR}$ 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 $\overline{WR}$ signal goes active. Figure 2.1.3-3 illustrates.

FIGURE 2.1.3-2

INSTRUCTION OP CODE FETCH

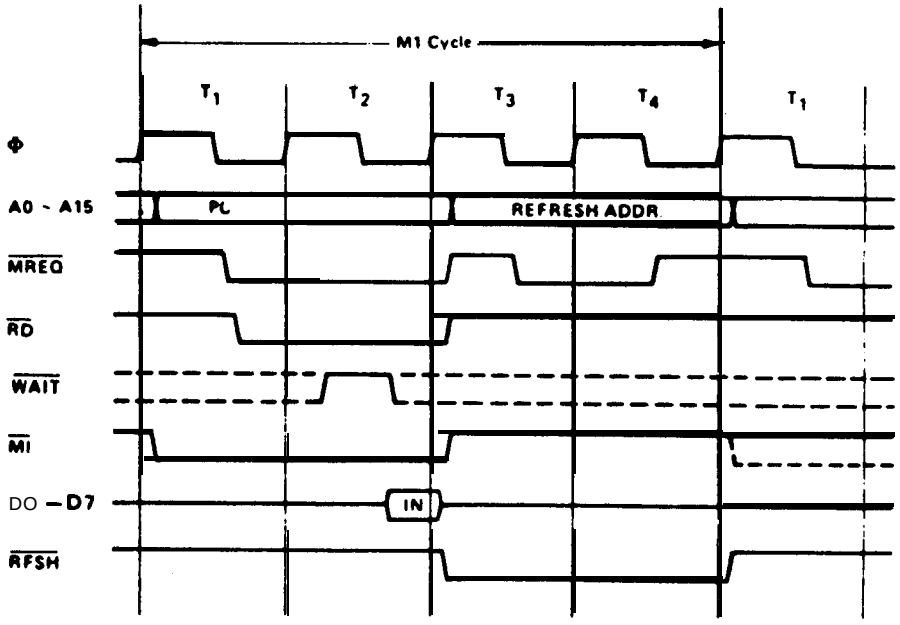
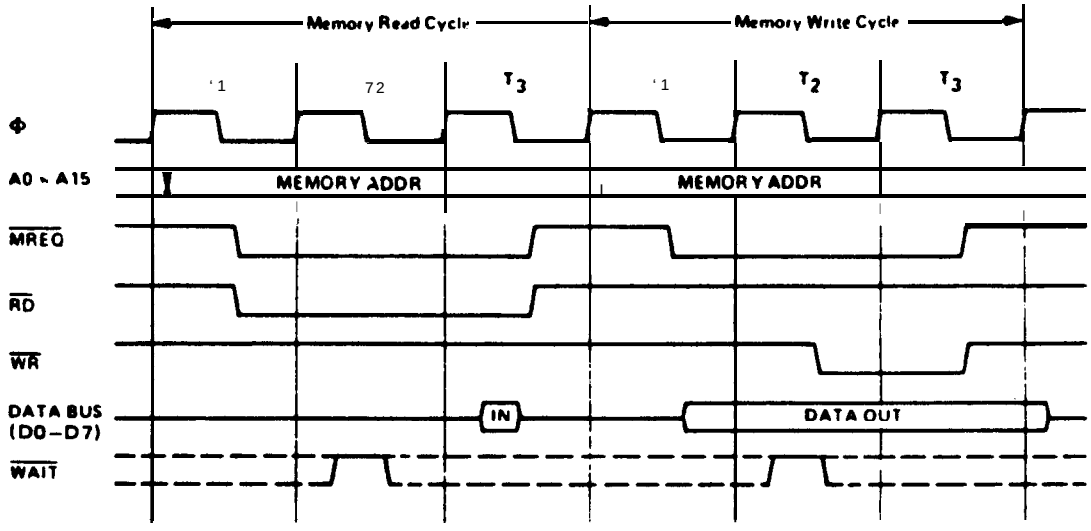


FIGURE 2.1.3-3

MEMORY READ OR WRITE CYCLES

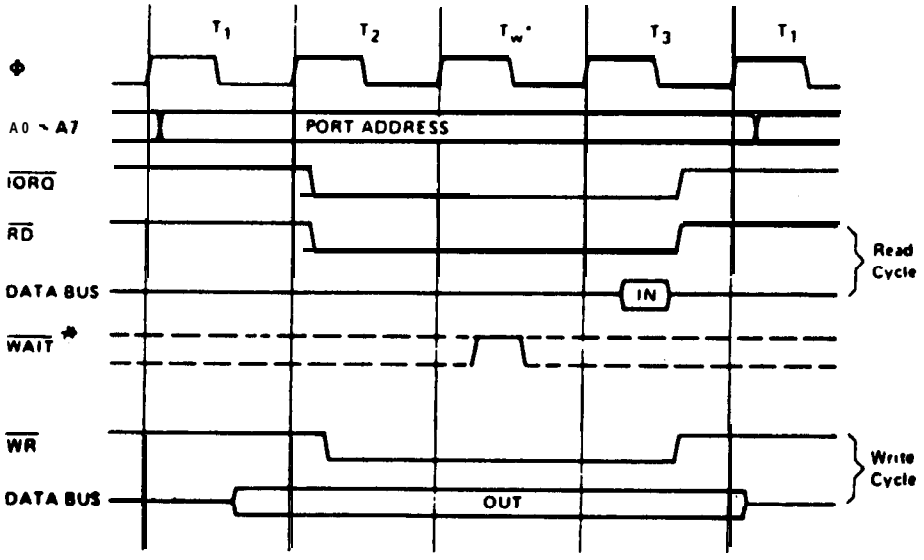


2.1.3.6 I/O READ/WRITE

During I/O operations $\overline{IORQ}$ and $\overline{RD}$ or $\overline{WR}$ 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 $\overline{RD}$ and $\overline{WR}$ signals are used to enable data from the addressed port onto the data bus and to, on the rising edge of $\overline{WR}$, clock data to the I/O port, respectively. Note that external I/O may stretch the activation period of the $\overline{WAIT}$ line to extend the I/O cycles.

FIGURE 2.1.3-4

INPUT OR OUTPUT CYCLES



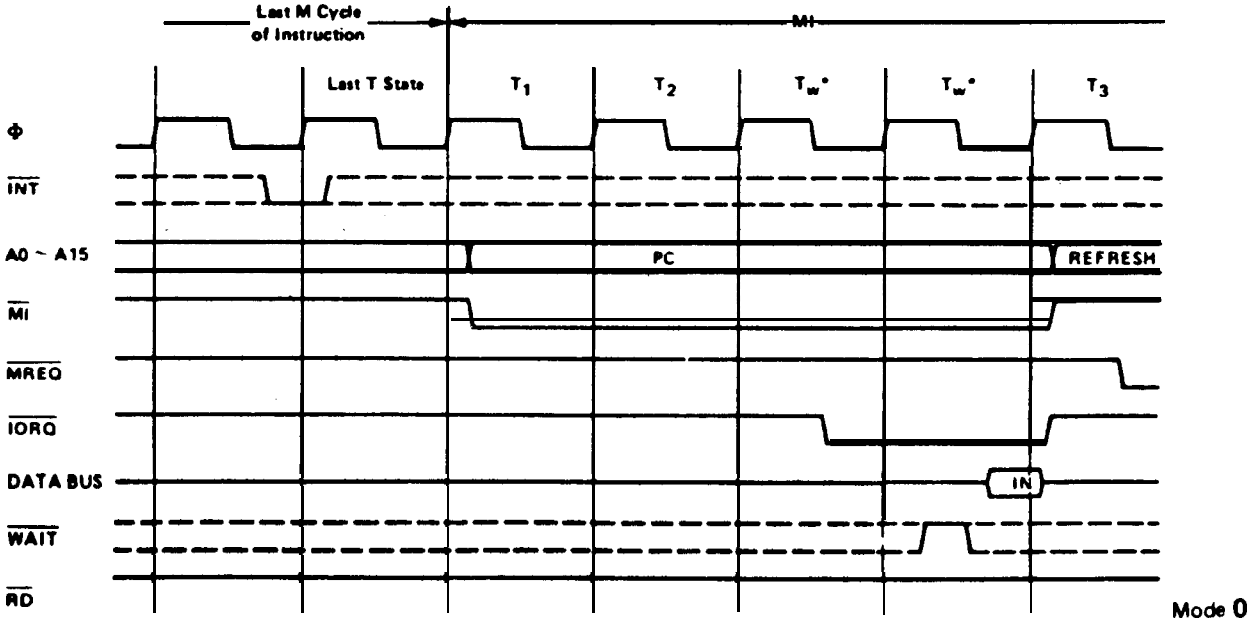
● Inserted by Z80 CPU

2.1.3.7 Maskable Interruption

When enabled by software, when $\overline{BUSRQ}$ is not active and when $\overline{INT}$ is active at the rising edge of the last clock of any instruction, a maskable interruption occurs during the subsequent Ml cycle, as illustrated in Figure 2.1.3-5.

FIGURE 2.1.3-5

INTERRUPT REQUEST / ACKNOWLEDGE CYCLE



In Interruption Mode 0, the interrupting I/O device places any instruction on the data bus during the $\overline{IORQ}$ activation and the CPU executes that instruction. The RESTART instruction is commonly used for this purpose. $\overline{RESET}$ 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 E-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 $\overline{\text{NMI}}$ 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 reponse 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 2-1

Z-80 CONTROL SIGNALS

	ACRONYM	DEFINITION
SYSTEM CONTROL	$\overline{M1}$	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. $\overline{M1}$ is also used with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.
	$\overline{MREQ}$	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.
	$\overline{IORQ}$	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 $\overline{M1}$ 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 $\overline{M1}$ time.
	$\overline{RD}$	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.
	$\overline{WR}$	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.
	$\overline{RI\text{-}SH}$	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.7 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.

TABLE 2-1

Z80 CONTROL SIGNALS
(continued)

	ACRONYM	DEFINITION
CPU CONTROL	$\overline{\text{HALT}}$	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.
	$\overline{\text{WAIT}}$	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.
	$\overline{\text{INT}}$	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 $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interruption, an acknowledge signal is sent out at the beginning of the next instruction cycle ($\overline{\text{IORQ}}$ at M1 time). There are three interruption modes selectable by the software.
	$\overline{\text{NMI}}$	Non-Maskable Interruption - Input, negative edge triggered. This signal has a higher priority than $\overline{\text{INT}}$ 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.

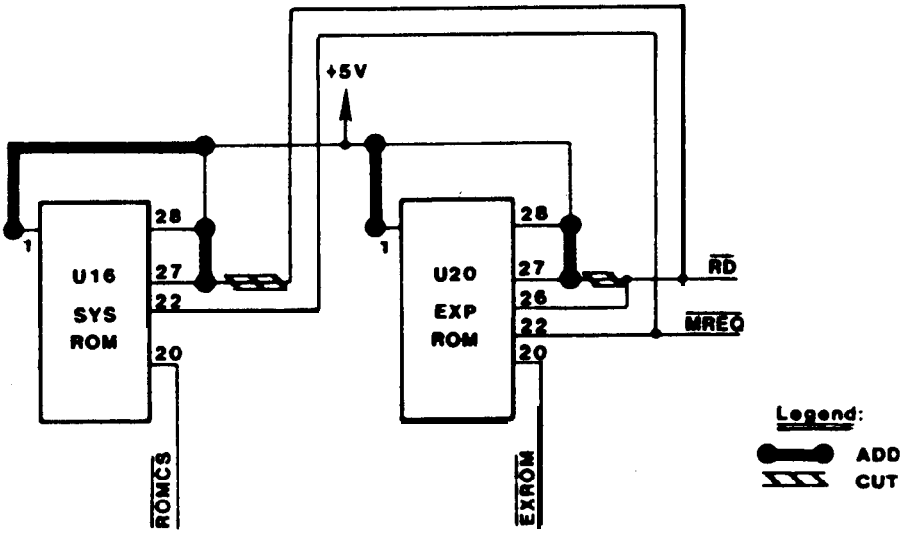
TABLE 2-1

Z80 CONTROL SIGNALS
(continued)

	ACRONYM	DEFINITION
	$\overline{\text{RESET}}$	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.
CPU BUS CONTROL	$\overline{\text{BUSRQ}}$	Bus Request - Input, active low. This signal is used to request the CPU address bus, data bus and tri-state output control signals to qo 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.
	$\overline{\text{BUSAK}}$	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 $\overline{\text{BUSRQ}}$.

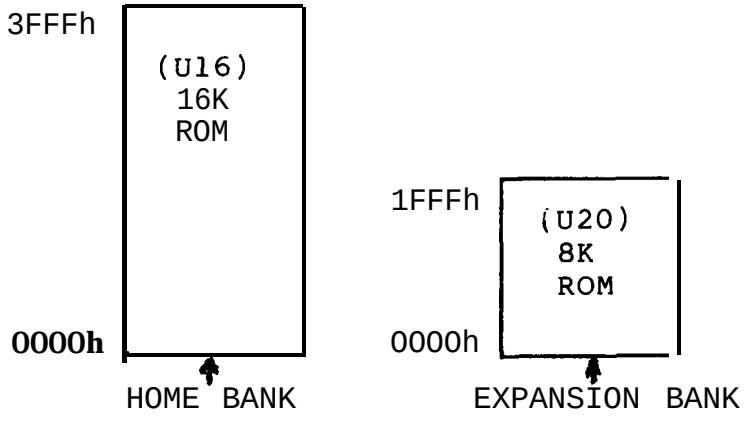
Figure 2.1.4-1

REWORK TO REPLACE ROM's with EPROM's



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-l) 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 programed 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 programed 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 000000000001 (divide by 1) and the highest period value is 111111111111 (divide by 4095).

FIGURE 2.1.6-l

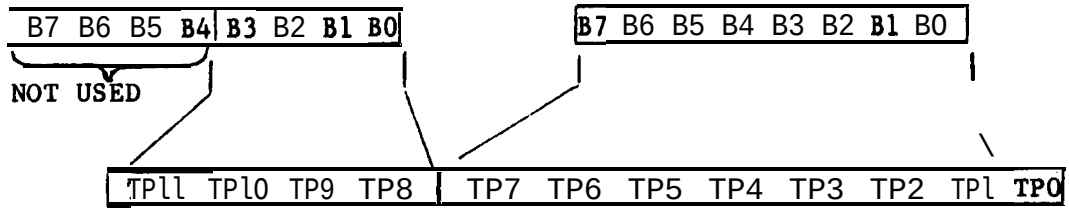
PSG REGISTER BLOCK DIAGRAM

REGISTER				BIT							
DEC	HEX	OCT		B7	B6	B5	B4	B3	B2	B1	B0
R0	R0	R0	Channel A	8 Bit Fine Tune							
R1	R1	R1	Tone Period	////////////////////				4 Bit Coarse Tune			
R2	R2	R2	Channel B	8 Bit Fine Tune							
R3	R3	R3	Tone Period	////////////////////				4 Bit Coarse Tune			
R4	R4	R4	Channel C	8 Bit Fine Tune							
R5	R5	R5	Tone Period	////////////////////				4 Bit Coarse Tune			
R6	R6	R6	Noise Period	//////////////				5 Bit Period Control			
				IN/OUT		NOISE			TONE		
R7	R7	R7	Enable	IOB	IOA	C	B	A	C	B	A
R8	R8	R10	Ch.A Amplitude	//////////			M	L3	L2	L1	L0
R9	R9	R11	Ch.B Amplitude	//////////			M	L3	L2	L1	L0
R10	RA	Rl2	Ch.C Amplitude	//////////			M	L3	L2	L1	L0
Rll	RB	Rl3	Envelope Period	8 Bit Fine Tune E							
R12	RC	R14		8 Bit Coarse Tune E							
R13	RD	R15	Envelope Shape/Cycle	////////////////////				CONT.	ATT.	ALT.	HOLD
R14	RE	R16	I/O Port A Data Store	8 Bit Parallel I/O on Port A							

FIGURE 2.1.6-2

12-BIT TONE PERIOD **(TP) TO TONE GENERATOR**

COARSE TUNE REGISTER	CHANNEL	FINE TUNE REGISTER
Rl	A	R0
R3	B	R2
R5	C	R4



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:

$$(a)\quad fT = \frac{fCLOCK}{16TP_{10}} \qquad (b)\quad TP_{10} = 256CT_{10} + FT_{10}$$

Where:

- fT = Desired tone frequency
- $fCLOCK$ = Input clock frequency
- TP_{10} = Decimal equivalent of the Tone Period bits TPll to TPO
- CT_{10} = Decimal equivalent of the Coarse Tune register bits B3 to BO (TPll to TP8)
- FT_{10} = Decimal equivalent of the Fine Tune register bits B7 to BO (TP7 to TP0)

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

fCLOCK/65520 (wherein TP_{10} = 4095)

to a high of:

fCLOCK/16 (wherein TP_{10} = 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) $TP_{10} = \frac{fCLOCK}{16\ fT}$

(b) $CT_{10} + \frac{FT_{10}}{256} = \frac{TP_{10}}{256}$

Example 1: fT = 1 kHZ fCLOCK = 1.76475 MHz

$$TP_{10} = \frac{1.76475 \times 10^{6}}{16(1 \times 10^{3})} = 110.3$$

Substituting this result into equation (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{110.3}{256}$$

resulting in:

$CT_{10} = 0 = 0000_{2}$ (B3-B0)

$FT_{10} = 110 = 01101110_{2}$ (B7-B0)

Example 2: fT = 100 Hz fCLOCK = 1.76475 MHz

$$TP_{10} = \frac{1.76475 \times 10^{6}}{16(1 \times 10^{2})} = 1103$$

Substituting this result into equation (b):

$$CT_{10} + \frac{FT_{10}}{256} = \frac{1103}{256} = 4 + 79/256$$

resulting in:

$CT_{10} = 4 = 0100_{2}$ (B3-B0)

$FT_{10} = 79 = 01001111_{2}$ (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-BO) of Register R6 as illustrated by Figure 2.1.6-3.

FIGURE 2.1.6-3

NOISE PERIOD REGISTER R6

B/	B6	B5	B4	B3	B2	B1	BO
NOT USED			5-BIT NOISE PERIOD (NP) TO NOISE GENERATOR				

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).
10

The noise frequency equation is: $fN = \frac{fCLOCK}{16\ NP_{10}}$

Where:
- fN = Desired noise frequency
- fCLOCK = Input clock frequency
- NP_{10} = 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 fCLOCK/496 (wherein NP_{10} =31) to a high of fCLOCK/16 (wherein NP_{10} = 1). Using a 1.76475 MHz 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:

$NP_{10} = fCLOCK/16fN$

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 BO 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-l and 2.1.6-2 below.

FIGURE 2.1.6-4

MIXER CONTROL - I/O ENABLE REGISTER R7

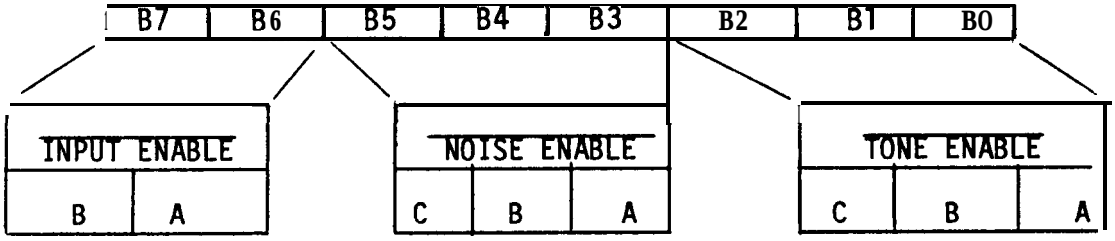


TABLE 2.1.6-1

I/O ENABLE TRUTH TABLE

NOISE ENABLE TRUTH TABLE		TONE ENABLE TRUTH TABLE	
R7 BITS B5 B4 B3	Noise tnabled on Channel	R/ BITS B2 Bl BO	Tone tnabled on Channel
0 0 0	C B A	0 0 0	C B A
0 0 1	C B -	0 0 1	C B -
0 1 0	C - A	0 1 0	C - A
0 1 1	C - -	0 1 1	c - -
1 0 0	B A	1 0 0	B A
1 0 1	B -	1 0 1	B -
1 1 0	- - A	1 1 0	A
1 1 1	- - -	1 1 1	- - -

TABLE 2.1.6-2

I/O PORT TRUTH TABLE

R7 BITS B6	I/O Port Status IOA
0	Input
1	output

NOTE

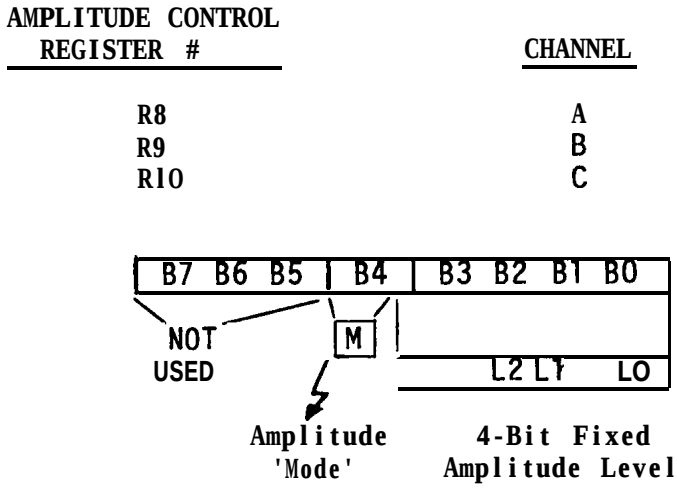
Disabling noise and tone does not turn off a channel. Turning a channel off can only be accomplished by writing all zeroes into the corresponding Amplitude Control register, R8, R9 or R10 (refer to Paragraph 2.1.6.4).

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



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

AMPLITUDE CONTROL REGISTER #	CHANNEL
R8	A
R9	B
R10	C

B7 B6 B5	B4	B3	B2	B1	B0	Amplitude Control output				
NOT USED	M	L3	L2	L1	L0					
	0	0	0	0	0	*0	0	0	0	The amplitude is fixed at 1 of 16 levels as determined by L3-L0.
	0	.	*.*			.	.	.	.	
	0	.	.	.	.	.	.	.	.	
	0	.	.	.	.	.	.	.	.	
	0					.	.	.	.	
	0	1	1	1	1	1	1	1	1	
	1	X	X	X	X	E3	E2	E1	E0	The amplitude is variable at 16 levels as determined by the output of the Envelope Gen.
		(X=Don't Care)								

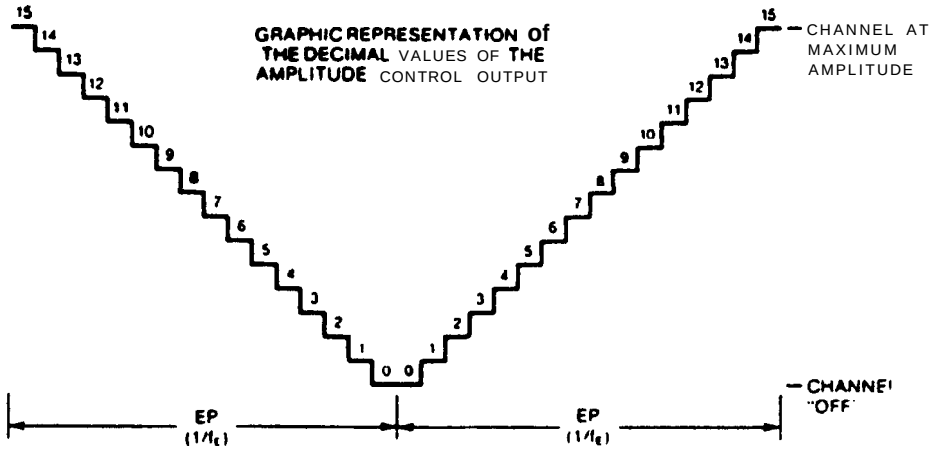
*The all zeros code is used to turn a channel "off".

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)

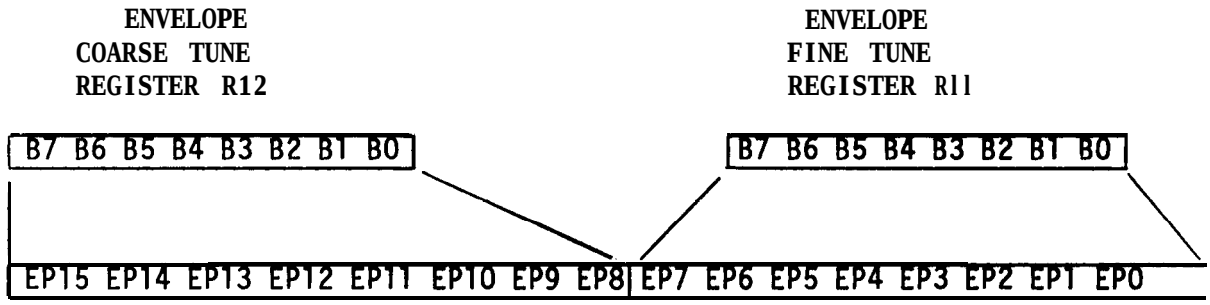
To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG; first, it is possible to vary the frequency of the envelope using registers R11 and R12; and second, the relative shape and cycle pattern of the envelope can be varied using register 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 programed 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



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 000000000000001 (divide by 1); the highest p-value is 1111111111111111 (divide by 65,535).
2 10

The envelope frequency equations are:

(a) $fE = \frac{fCLOCK}{256\ EP_{10}}$

(b) $EP_{10} = 256\ CT_{10} + FT_{10}$

Where:

fE	=	Desired envelope frequency
fCLOCK	=	Input clock frequency
EP_{10}	=	Decimal equivalent of the Envelope Period bits EP15-EP0
CT_{10}	=	Decimal equivalent of the Coarse Tune register bits B7-B0 (EP15-EP8)
FT_{10}	=	Decimal equivalent of the Fine Tune register bits B7-B0 (EP7-EP0)

From the above equation it can be seen that the envelope frequency can range from a low of fCLOCK/16,766,960 (wherein EP_{10} = 65,535) to a high of fCLOCK/256 (wherein EP_{10} =1). Using a 1.76475 MHz clock, 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) $EP_{10} = \frac{fCLOCK}{256fE}$

(b) $CT_{10} + \frac{FT_{10}}{256} = \frac{EP_{10}}{256}$

Example:

fE = 0.5 Hz
fCLOCK = 1.76475 MHz

$$EP_{10} = \frac{1.76475 \times 10^{6}}{256(0.5)} = 13787$$

Substituting this result into equation (b):

$$CT_{10} + \frac{FT_{10}}{256} = 13787 = 53 + \frac{219}{256}$$

$CT_{10} = 53 = 00110101_{2}$ (B7-B0)

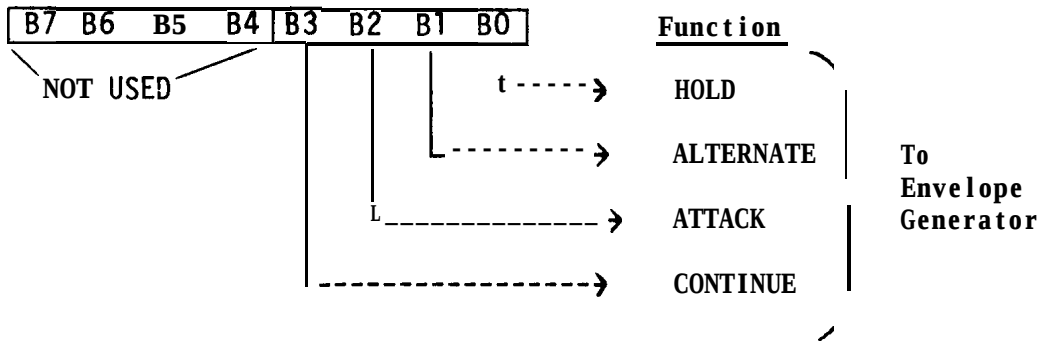
$FT_{10} = 219 = 11011011_{2}$ (B7-B0)

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.

FIGURE 2.1.6-9

ENVELOPE SHAPE/CYCLE CONTROL REGISTER (R13)



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

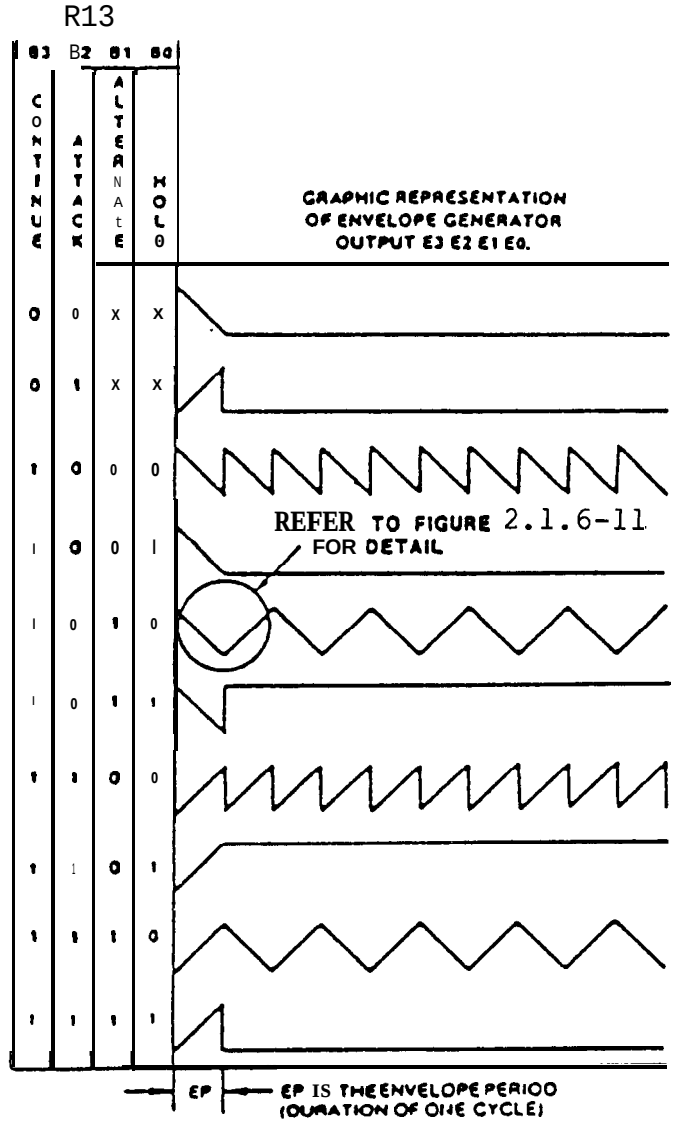
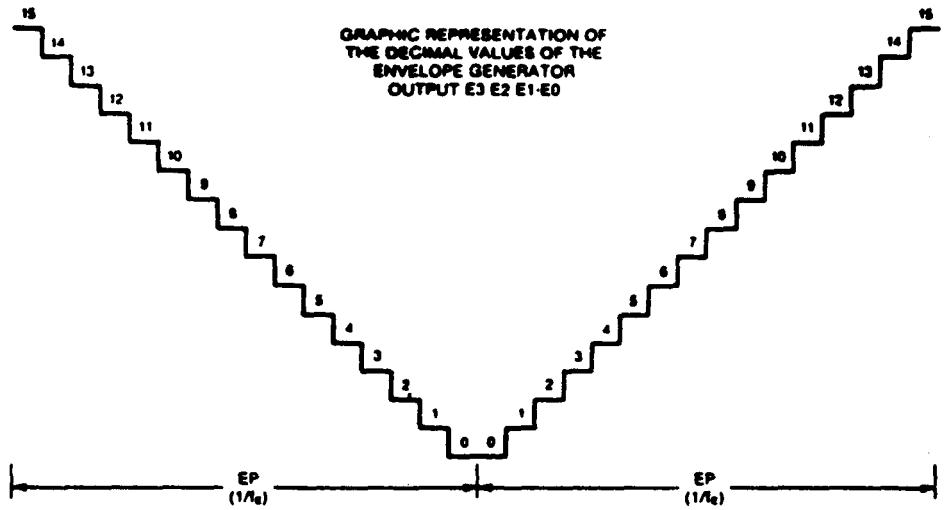


FIGURE 2.1.6-11

DETAIL OF TWO CYCLES OF FIGURE 2.1.6-10



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 2.1.6.6) 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-l. In this state, a read of port F6H with address bit A8 high, causes actions as follows:

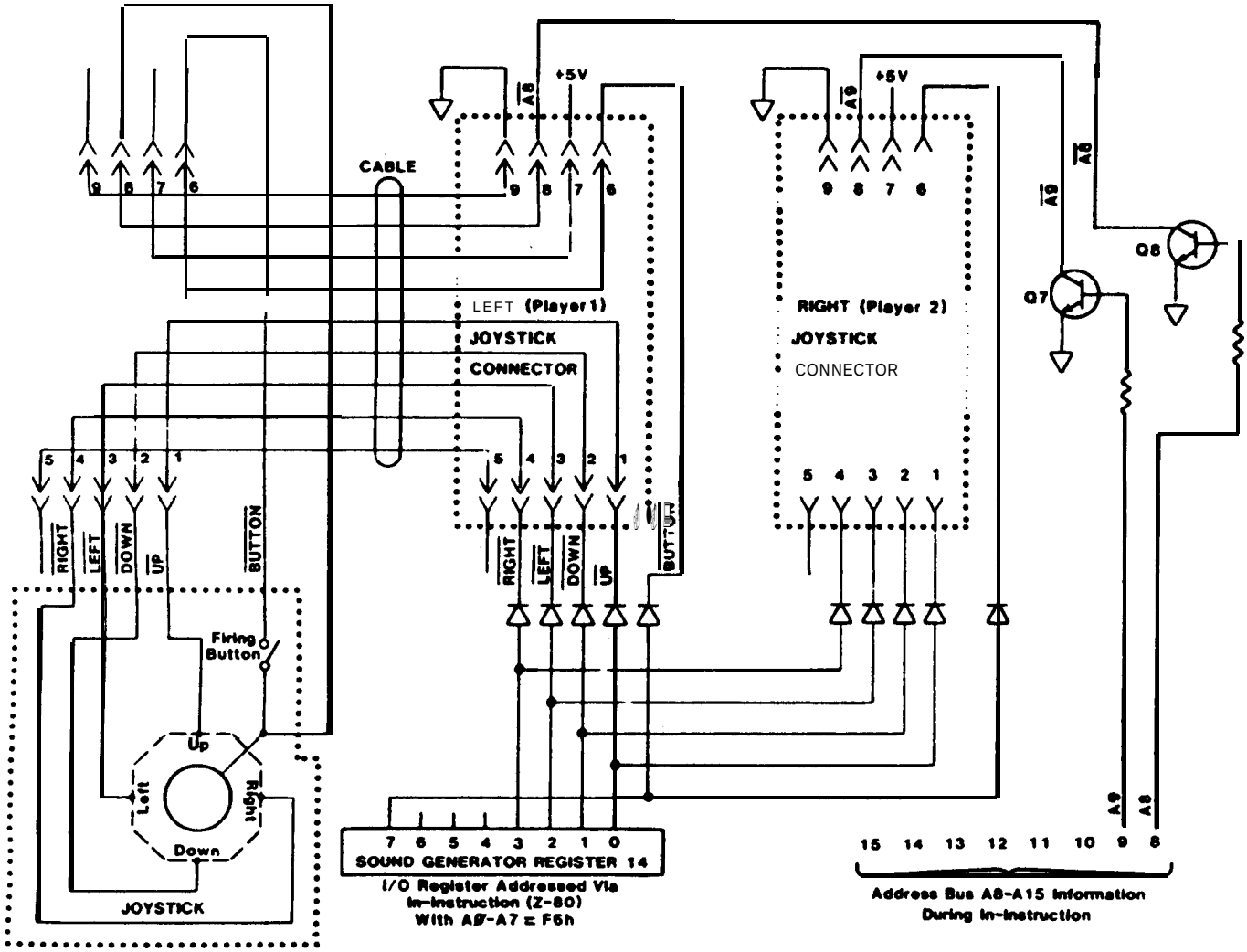
(1) Address A8 high turns on transistor Q8
(2) Q8 drives cable pin-8 low
(3) The movable center stick of the joystick in contact with the up-contact results in a conductive path from cable pin-8 to cable pin-l.
(4) Pin-l low results in a 0 in bit position 0 of the I/O register via the isolation diode.

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:

SECTION	FUNCTION
2.1.8.1	Bank Selection Logic
2.1.8.2	Z-80 Clock Generation
2.1.8.3	Display Timing, DMA Display File Access, Attribute Control, and Pixel Data Serial Shift
2.1.8.4	Interruption Generation
	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 $\overline{BE}$ (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. $\overline{BE}$ is high

causing activation of $\overline{ROSCS}$ (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. $\overline{BE}$ is high

causing the activation of $\overline{EXROM}$ (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. $\overline{BE}$ 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 $\overline{ROSCS}$ to its low active state

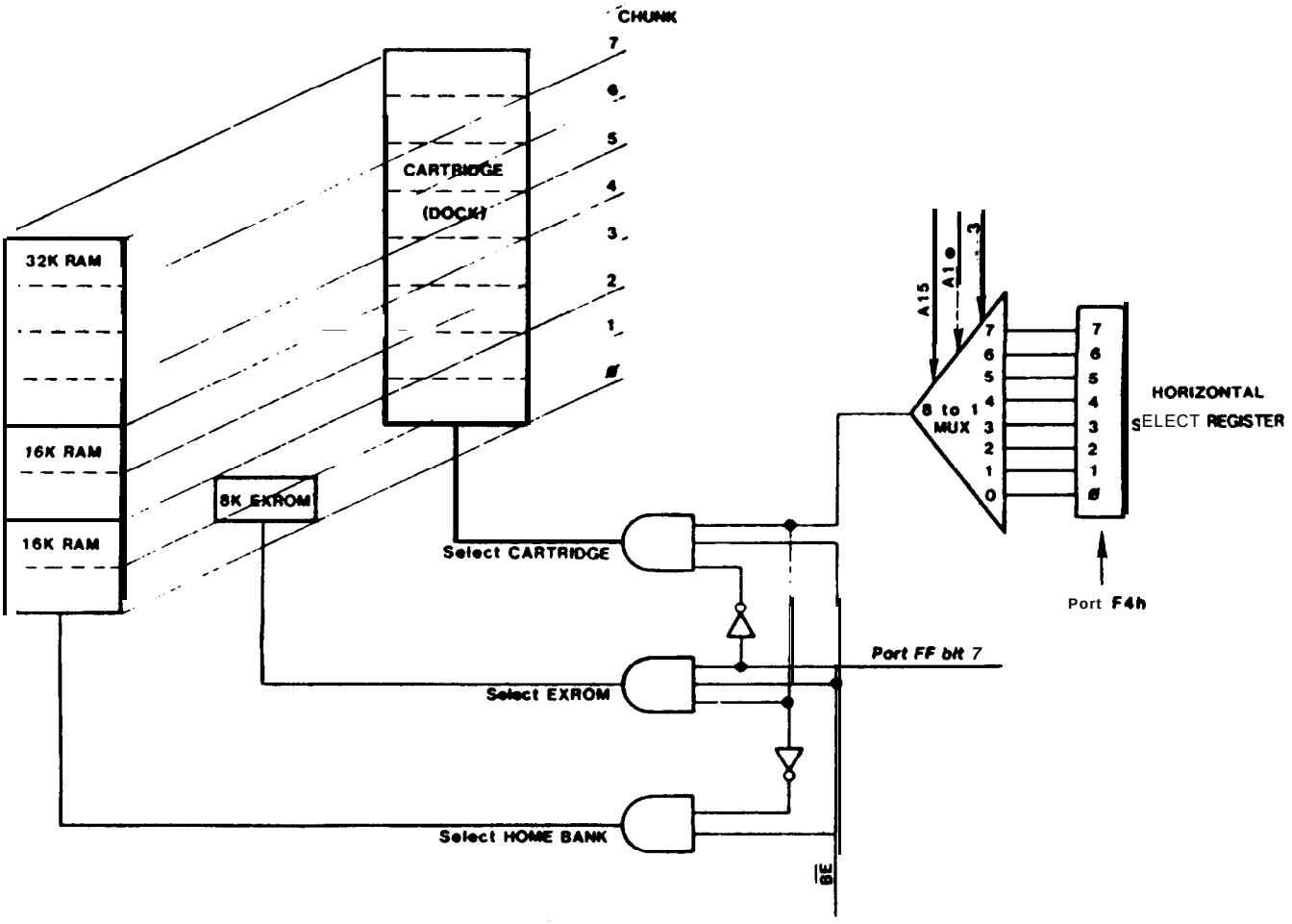
(2) SELECT EXROM of Figure 2.1.8-1 involves activating $\overline{EXROM}$ to its low active state

(3) SELECT HOME BANK of Figure 2.1.8-1 involves

a. Activating $\overline{ROMCS}$ to its low active state when A15=0 and A14=0
b. Activating $\overline{CAS1}$ to its low active state when A15=0 and A14-1
c. Activating $\overline{CAS2}$ to its low active state when A15=1 and A14-0
d. Activating $\overline{CAS3}$ to its low active state when A15=1 and A14=1.

FIGURE 2.1.8-1

BANK SELECTION LOGIC



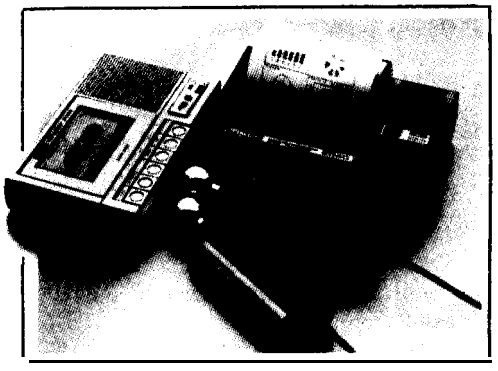


TABLE 2.1.8-l

SCLD I/O PIN FUNCTION DEFINITIONS

SYMBOL	NAME	DIRECTION OF SCLD IN/OUT	FUNCTION
AO-A7 A13-A15	Address Bus	In	Address Bus lines Input from Z80A
DO-D7	Data Bus	In/Out	Data Bus inputs/outputs from/to Z80A through U9-74LS245 or inputs from display RAM (16K) - U6 and U7
KBO-KB4	Keyboard Outputs	In	Inputs from 5 lines of keyboard matrix - goes low at one of 8 address line (active low) sequences on I/O Request
A7R	A7+Refresh	out	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)
MAO-MA7	Muxed Adrs.Bus	out	Display memory muxed address bus and refresh
$\overline{TS}$	Tri-State Display Memory Ctl.	out	Tri-State control for address and data buffers when CPU is addressing display memory at same time display controller is addressing the display memory
OCPU	Clock to CPU	out	CLK - Clock to Z80A CPU which is interrupted to stop CPU when CPU wants to address display RAM at same time as display controller
$\overline{RD}$*	Read Direction Control to SCLD	out	To control read/write direction of 74LS245 Data Bus Buffer between CPU and SCLD
$\overline{ROMCS}$	Home ROM Chip Select	out	To activate the 16K Home ROM (first 16K) when memory selection (MS) is set to Home Bank
$\overline{RAST}$	Row Address Strobe #1	out	To activate row address strobe for display memory only during memory read/write, refresh and display read

TABLE 2.1.8-1

SCLD I/O PIN FUNCTION DEFINITIONS
(continued)

SYMBOL	NAME	DIRECTION OF SCLD IN/OUT	FUNCTION
$\overline{CAS1}$	Column Address Strobe #l	out	To activate column address strobe for display memory only (2nd 16K) durinq memory read/write and display read
$\overline{CASZ}$	Column Address Strobe #2	out	To activate column address strobe for Home Bank RAM (3rd 16K)
$\overline{CAS3}$	Column Address Strobe #3	out	To activate column address strobe for Home Bank RAM (4th 16K)
$\overline{DRAMWE}$	Dynamic RAM Write Enable	out	When active low, enables a write into the display RAM only
MUX	Mux Control of RAM Address	out	Mux control to 74LS157 (U10 & Ull to multiplex the row and column addresses to all dynamic RAM's
V	Chroma Vector V	out	Color vector level for quadrature (R-Y) input to video modulator
$\overline{Y}$	Luminance $\overline{Y}$	out	Luminance (briqhtness) control level
$\overline{RD}$	Read to CPU	In	CPU is reading from a memory or I/O location
$\overline{WR}$	Write from CPU	In	CPU is writing to a memory or I/O location
$\overline{MREQ}$	Memory Request	In	CPU is requesting access to a memory location to read or write
$\overline{IORQ}$	I/O Request	In	CPU is requesting access to an I/O location to read or write

TABLE 2.1.8-l

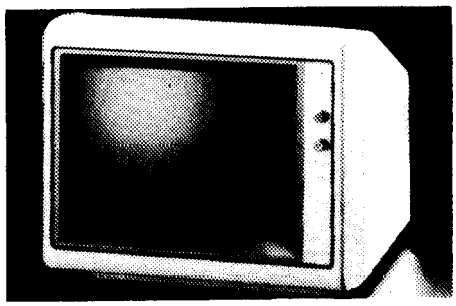
SCLD I/O PIN FUNCTION DEFINITIONS
(continued)

SYMBOL	NAME	DIRECTION OF SCLD IN/OUT	FUNCTION
$\overline{\text{RFSH}}$	Refresh	In	CPU is generating a refresh address to refresh dynamic RAM's
Tape In	Tape Input	In	Magnetic tape signal input
$\overline{\text{BE}}$	Bank Enable	In	When active low, indicates that internal memory is disabled (Home, Extension and Dock Banks) and an external memory is in use
$\overline{\text{EXROM}}$	Extension ROM Select	out	Active low chip select signal for Extension ROM
vcc	+5 Volt Power	In	Power (+5V) input to SCLD
$\overline{\text{INT}}$	Interrupt to CPU	out	Interrupts CPU to handle keyboard strobing and timer for PAUSE command. Open drain N channel with internal pull-up
$\overline{\text{ROSCS}}$	ROS Chip Select	out	ROM-Oriented Software (Cartridge Bank) Chip Select
SPKR/TAPE OUT	Speaker and Tape Output	out	Digital output to magnetic tape and to sound amplifier for speaker output
oc	Clock "C"	out	Clock for sound chip 81.764 MHz.
BDIR	Bus Direction to Sound Chip	out	A bus direction control signal to the PSG. When high the sound chip either receives a write to PSG or latches addresses from the data bus
BCl	Bus Control to Sound Chip	out	A bus control signal to the PSG. When high the sound chip either is read to data bus or latches addresses from the data bus

TABLE 2.1.8-1

SCLD I/O PIN FUNCTION DEFINITIONS
(continued)

SYMBOL	NAME	DIRECTION OF SCLD IN/OUT	FUNCTION
$\overline{OSC}$ out	OSC1'llator Out	out	Xtal Oscillator amplifier output to drive crystal
OSC In	Oscillator In	In	Xtal Oscillator amplifier input to sense crystal signal
U	Chroma Vector U	out	Color vector level for quadrature (B-Y) input to video modulator
GND	Ground	In	Ground return of SCLD
$\overline{O}$	Buffered Clock	out	Buffered CPU clock to outside (Jl - connector)
R	Red Color Output	Out	Produce color signals to RGB monitor (TTL level)
G	Green Color output	out	Produce color signals to RGB monitor (TTL level)
B	Blue Color output	out	Produce color signals to RGB monitor (TTL level)



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.

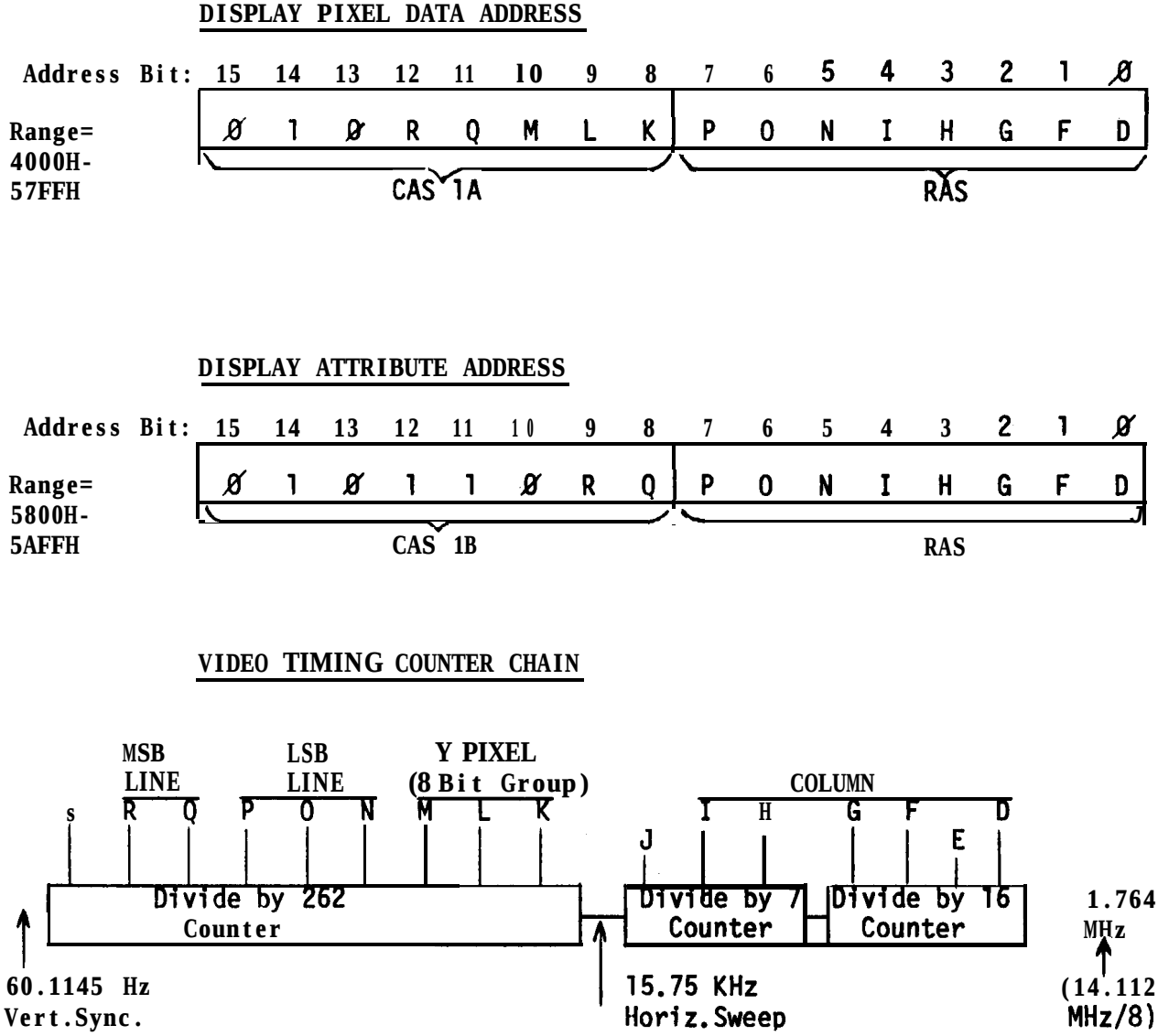
During each horizontal scan the video display processor accesses, in the standard video mode, 32 bytes of pixel data plus 32 bytes of attributes by 32 memory accesses reading 2 bytes per access in RAM page mode, i.e. the low order address bits are provided to the RAM once via RAS activation, then the 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 Fiq. 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 qated 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 $\overline{Y}$ outputs for use by the 1889 to create composite video.

The address information provided to the RAM's duri nq 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)



2.1.8.4 Interruption Generation (17 ms)

During the vertical blanking interval (once each 15.635 ms) the SCLD, if enabled by the INTEN bit (Bit 6) of I/O Port FFH, activates the $\overline{INT}$ signal which directly connects to the $\overline{INT}$ input to the 280. 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 (KBO through KB4).

Scanning is by software algorithm as described in Section 4.1.1. During the IN instruction, address bits AO-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 O's on KBO, KBl, KB2, KB3, and/or KB4 if the CAP SHIFT, Z, X, C, and/or V keys are respectively denressed.

Note that when reading the I/0 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 400DH and 600DH respectively. Figure 2.1.10-l 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 2.1 .9. KEYBOARD SCHEMATIC

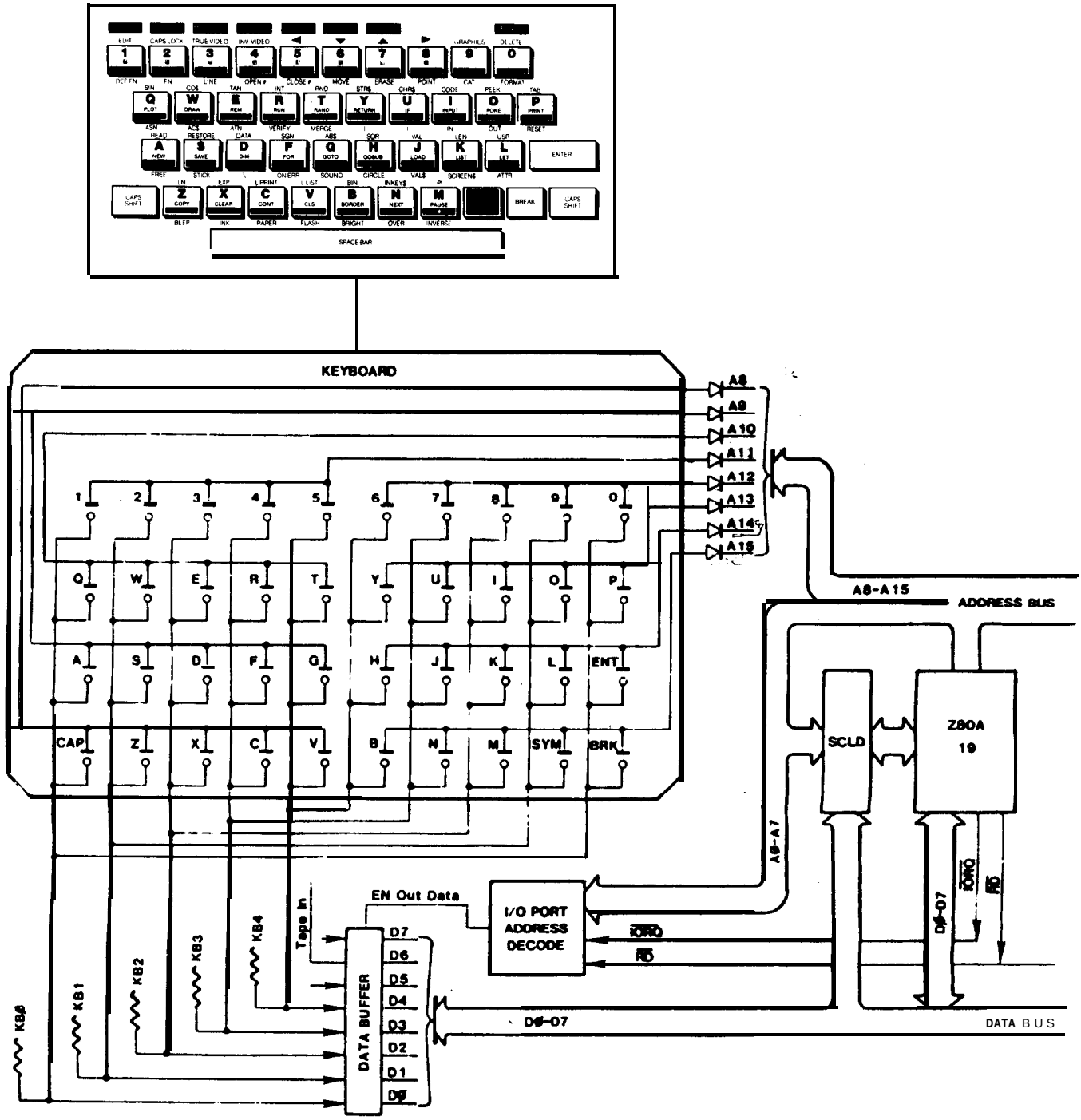


FIG. 2.1.10-1
DISPLAY FILE ORGANIZATION (NORMAL MODE)

		32 BYTES LINE 0		32 BYTES LINE 1		32 BYTES LINE
BLOCK 0	Scan 0	4000	4001.....401F	4020.....F	.....	40E0 40E1 .....F
	1	4100	4101.....411F	4120.....413F	.....	41E0 41E1 .....41F F
	2	4200	4201.....421 F	4220.....423 F	.....	42E0 42E1 .....42F F
	3	4300	4301.....431 F	4320.....433 F	.....	43E0 43E1 .....43F F
	4	4400	4401.....441 F	4420.....443 F	.....	44E0 44E1 .....44F F
	5	4500	4501.....451 F	4520.....453 F	.....	45E0 45E1 .....45F F
	6	4600	4601.....461F	4620.....463F	.....	46E0 46E1 .....46F F
	7	4700	4701.....471F	4720.....473F	.....	47E0 47E1 .....47F F
		CHAR. POS. 0/0	CHAR. POS. 0/1	CHAR. POS. 0/31		CHAR. POS. 7/0 CHAR. POS. 7/1 CHAR. POS. 7/31

		32 BYTES LINE 8		32 BYTES LINE 9		32 BYTES LINE 15
BLOCK 1	Scan 0	4800	4801.....481F	4820.....483F	.....	48E0 48E1 .....48FF
	1	4900	4901.....491F	4920.....493F	.....	49E0 49E1 .....49FF
	2	4A00	4A01.....4A1F	4A20.....4A3F	.....	4AE0 4AE1 .....4AFF
	3	4B00	4B01.....4B1F	4B20.....4B3F	.....	4BE0 4BE1 .....4BFF
	4	4C00	4C01.....4C1F	4C20.....4C3F	.....	4CE0 4CE1 .....4CFF
	5	4D00	4D01.....4D1F	4D20.....4D3F	.....	4DE0 4DE1 .....4DFF
	6	4E00	4E01.....4E1F	4E20.....4E3F	.....	4EE0 4EE1 .....4EFF
	7	4F00	4F01.....4F1F	4F20.....4F3F	.....	4FE0 4FE1 .....4FFF
		CHAR. POS. 8/0	CHAR. POS. 8/1	CHAR. POS. 8/31		CHAR. POS. 15/0 CHAR. POS. 15/1 CHAR. POS. 15/31

		32 BYTES LINE 16		32 BYTES LINE 17		32 BYTES LINE 23
BLOCK 2	Scan 0	5000	5001.....501F	5020.....3F	.....	E0 50E1 .....50FF
	1	5100	5101.....511F	5120.....513F	.....	51E0 51E1 .....51FF
	2	5200	5201.....521F	5220.....523F	.....	52E0 52E1 .....52FF
	3	5300	5301.....531F	5320.....533F	.....	53E0 53E1 .....53FF
	4	5400	5401.....541F	5420.....543F	.....	54E0 54E1 .....54FF
	5	5500	5501.....551F	5520.....553F	.....	55E0 55E1 .....55FF
	6	5600	5601.....561F	5620.....563F	.....	56E0 56E1 .....56FF
	7	5700	5701.....571F	5720.....573F	.....	57E0 57E1 .....57FF
		CHAR. POS. 16/0	CHAR. POS. 16/1	CHAR. POS. 16/31		CHAR. POS. 23/0 CHAR. POS. 23/1 CHAR. POS. 23/31

ATTRIBUTE FILE:

BLOCK 0	LINE 0 5800.....581F	LINE 1 5820.....583F	LINES 2 - 6 5840.....58DF	LINE 7 58E0.....58FF
BLOCK 1	LINE 8 5900.....591F	LINE 9 5920.....593F	LINES 10-14 5940.....59DF	LINE 15 59E0.....59FF
BLOCK 2	LINE 16 5A00.....5A1F	LINE 17 5A20.....5A3F	LINES 18-22 5A40.....5ADF	LINE 23 5AE0.....5AFF

2.1.11 Video Generation

2.1.11.1 Composite Video

The U, V, and $\overline{Y}$ 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:

PHASE	TS 2068 (Degrees)	NTSC STANDARD (Degrees)
Blue	350	350
Magenta	64	62
Red	116	112
Green	242	240
Cyan	284	284
Yellow	170	170
Reference	224	180

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.

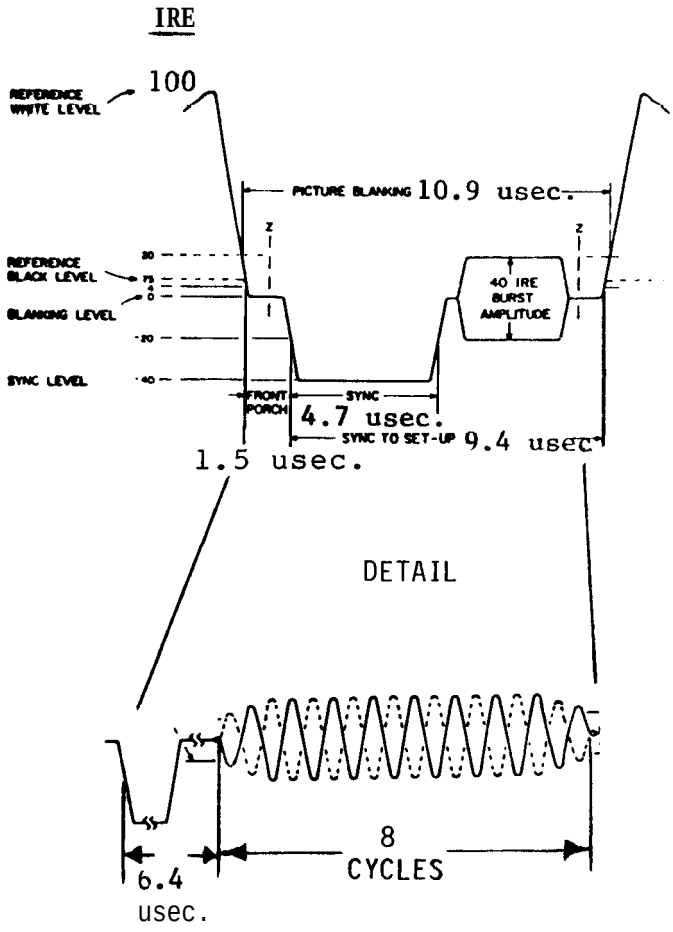
1. The color burst is not synchronous with the waveform since it is generated from the 3.579545 MHz crystal and the waveform is derived from the 14.112 MHz crystal. The result is observed ripples at color boundaries, e.g. green to magenta.

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.

FIGURE 2.1.11-1

COMPOSITE VIDEO SIGNALS



2.1.11.2 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)

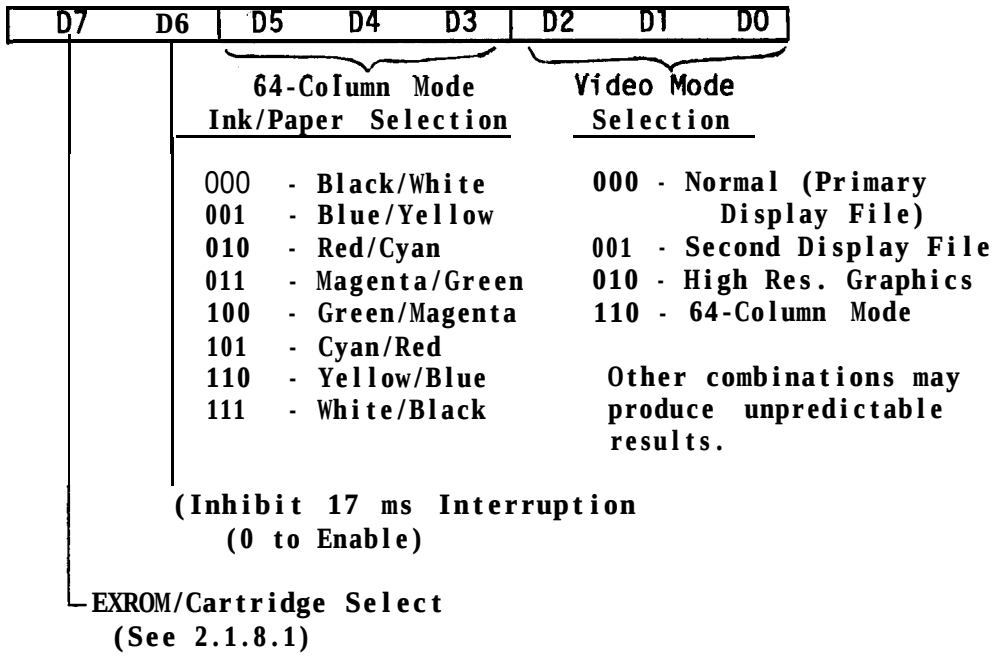


TABLE 2.1.13-1

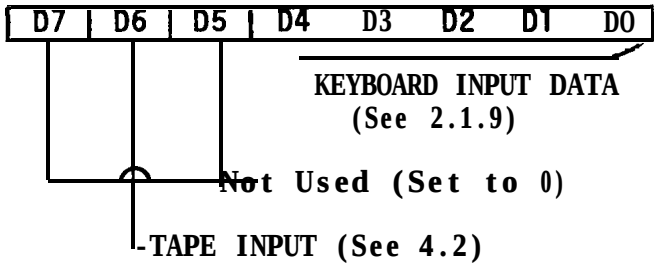
I/O PORT MAP

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

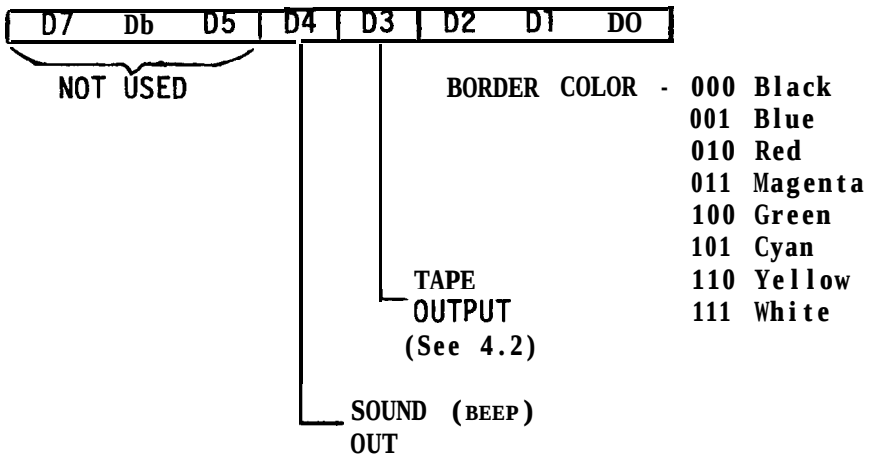
2.1.13.2 Keyboard/Tape I/O (Port FEH)

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

READ (IN)



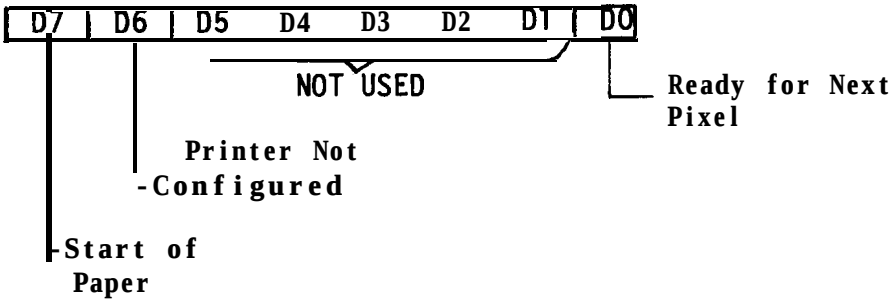
WRITE (OUT)



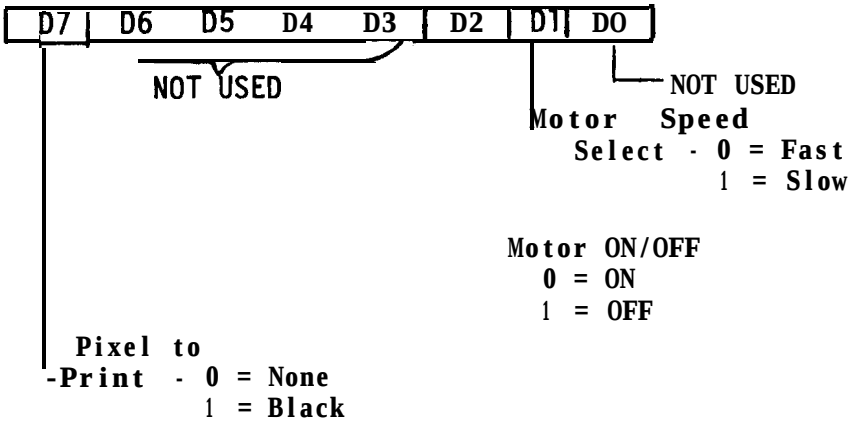
2.1.13.3 TS 2040 Printer (Port 1XXXXOXX)

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)



WRITE (OUT)



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 f n 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 fn 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 entf re Home Bank.

2.2 Schematic Diagram

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

2.3 Unit Absolute Ratings

FUNCTION	DESCRIPTION	MIN	MAX
TS	Storage Temperature	-40c	+65C
VAC	AC Line Voltage	105v	130v
Ta	Operating Ambient Temp	0C	40c
Vfn	Voltage on any Logic Pin	-0.3v	+5.3v
Vfn (EAR)	EAR input Peak AC	-2.0v	+5.0v
Vdc (IN)	Input DC Voltage	14.75V	26V

2.4 Interfaces and Connectors

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

CONNECTOR	TYPE	LOCATION
System Bus	2X32 Card Edge	Right Rear
Cartridge	2X18 Card Edge	Under TCC door
MIC	1/8" Mini Phone	Rear
EAR	1/8" Mini Phone	Rear
Player 1 Joystick	9-pin "D"	Left Side
Player 2 Joystick	9-pin "D"	Right Side
Monitor	RCA Phono	Rear
TV	RCA Phono	Rear
Keyboard	14-pin SIP	Inside-Left Rear
AC Adapter		Rear

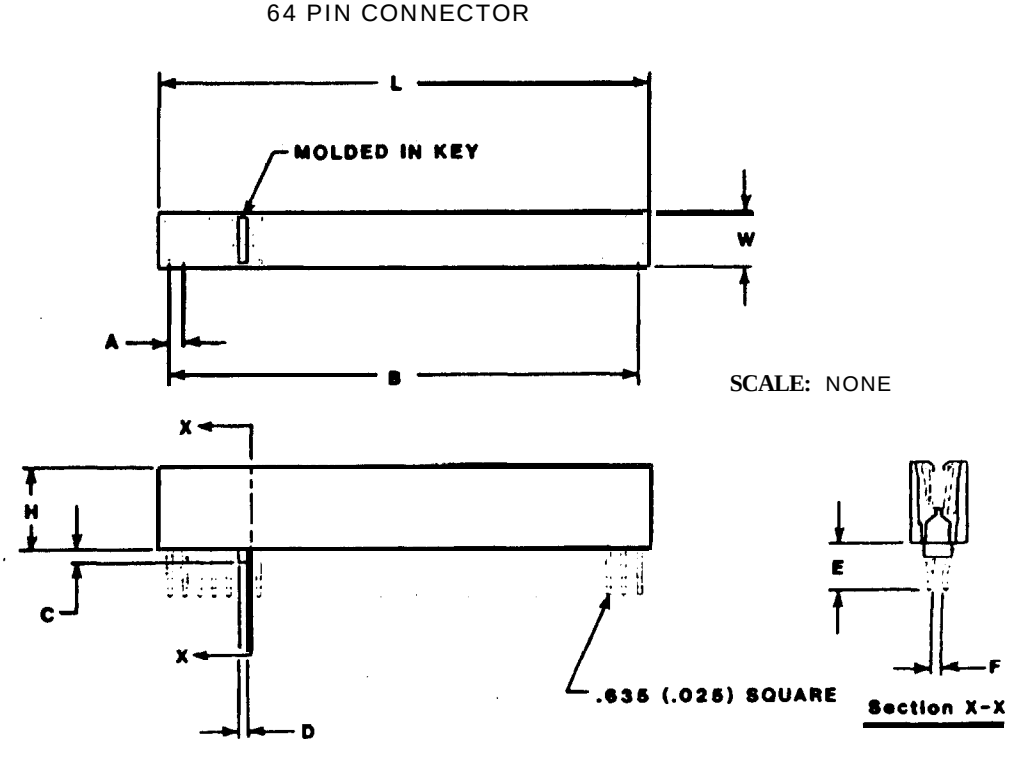
2.4.1 System Bus Connector - P1

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 buss connector are detailed in the following tables and figures:

FIGURE/TABLE	TITLE
Figure 2.4.1-1 P1	Mating Connector Mechanical Requirements
Figure 2.4.1-2 P1	Signal Layout
Table 2.4.1 - 1 P1	Signal Definition
Table 2.4.1 - 2 P1	Signal Electrical Characteristics

FIGURE 2.4.1-1

P1 MATING CONNECTOR MECHANICAL REQUIREMENTS



SPECIFICATIONS:

LTR	DIMENSION *
L	82.55 (3.25)
W	9.525±0.127 (.375±.005)
H	13.97±0.254 (.550±.010)
A	2.54 (.100)
B	31 EQUAL SPACES AT 2.54 (.100)=78.74 (3.100)
C	2.54 (.10)
D	1.727 (.068) MAX
E	8.382±0.508 (.330±.020)
F	FOR 1.575 (.062) BOARD

*All dimensions are in millimeters, dimensions shown (XX.X) are in inches.

NOTES:

1. INSULATOR MATERIAL: Insulator body shall be 30% glass-filled polyester and shall meet UL94V-0 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 1.575 (.062) flat steel test blade
5. WITHDRAWAL FORCE: Withdrawal forces shall be 226.8-340.2 grams (8-12 oz) per contact pair using a 1.575 (.062) 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

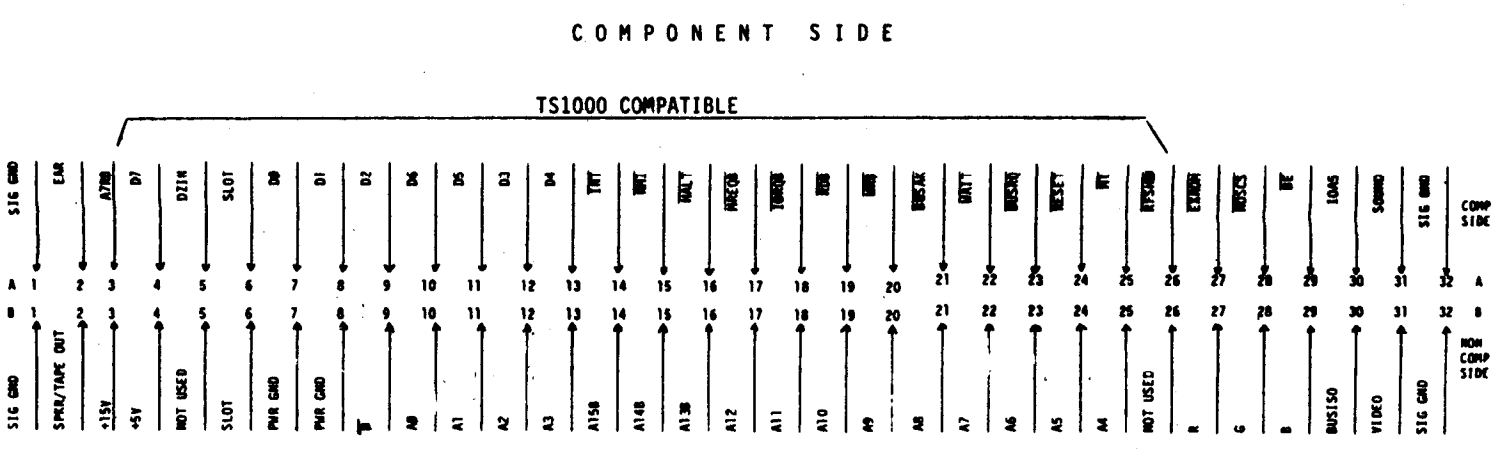


TABLE 2.4.1 - 1

P1 SIGNAL DEFINITION

PIN #	SIGNAL NAME	DESCRIPTION
1A	**GND**	Signal Ground
1B	**GND**	Signal Ground
2A	EAR	EAR Input
2B	SPKR/TAPE OUT	Speaker/Tape Output
3A	A7RB	Refresh Address Bit 7 Buffered
3B	+15v	+15 Volts DC
4A	D7	Data Bus Bit 7
4B	+5v	+5 Volts
5A	DZIN	Daisy In (Not Connected)
5B	Not Used	—
6A	Slot	—
6B	Slot	—
7A	**DO**	Data Bus Bit 0
7B	**GND**	Power Ground
8A	Dl	Data Bus Bit 1
8B	GND	Power Ground
9A	D2	Data Bus Bit 2
9B	**Ō**	CPU Clock (Inverted)
10A	D6	Data Bus Bit 6
10B	A0	Address Bus Bit 0
11A	D5	Data Bus Bit 5
11B	Al	Address Bus Bit 1
12A	D3	Data Bus Bit 3
12B	A2	Address Bus Bit 2
13A	D4	Data Bus Bit 4
13B	A3	Mdress Bus Bit 3
14A	**INT**	Interrupt Bequest (Active Low)
14B	Al5B	Address Bus Bit 15, Buffered
15A	**NMI**	Non-Maskable Int.(Active Low)
15B	A14B	Address Bus Bit 14, Buffered
16A	**HALT**	CPU HALT Indicator (Active Low)
16B	A13B	Address Bus Bit 13, Buffered
17A	**MREQB**	Memory Request (Active Low),Bfrd.
17B	A l 2	Address Bus Bit 12
18A	**IORQB**	I/O Request (Active Low), Bfrd.
18B	**All**	Mdress Bus Bit 11
19A	**RDB**	Read (Active Low), Buffered
19B	A10	Mdrees Bus Bit 10
20A	**WRB**	Write (Active Low), Buffered
20B	A9	Mdress Bus Bit 9
21A	**BUSAK**	Bus Acknowledge (Active Low)
21B	A8	Mdress Bus Bit 8
22A	**WAIT**	CPU WAIT (Active Low)
22B	A7	Mdresa Bus Bit 7
23A	BUSRQ	Bus Request (Active Low)
23B	A6	Address Bus Bit 6
24A	**RESET**	CPU Reset (Active Low)
24B	A5	Address Bus Bit 5
25A	**Ml**	CPU Ml State (Active Low)
25B	A4	Address Bus Bit 4
26A	RFSHB	Refresh (Active Low),Buffered
26B	DZOUT	Daisy Out (Not Connected)
27A	EXROM	Extension ROM Enable (Active Low)
27B	R	Color Signal - Red
28A	**ROSCS**	ROS Chip Select (Active Low) (Dock Bank Enable)
28B	G	Color Signal - Green
29A	**BE**	Bank Enable (Active Low)
29B	B	Color Signal - Blue
30A	IOA5	
30B	BUSISO	
31A	SOUND	Analog Sound Signal Output(0-5V)
31B	VIDEO	Composite Video Signal Output
32A	GND	Signal Ground
32B	**GND**	Signal Ground

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

	OUTPUTS FROM TS2068				INPUTS TO TS2068			
MNEMONIC	CAPACITIVE LOADING MAX (PF)	V(OL) MAX VOLTS	I(LOAD) MAX (MA)	V(OH) MIN VOLTS	V(IL) MAX VOLTS	V(IH) MIN VOLTS	I IN (MAX) uA	INPUT CAPACITIVE LOADING MAX (PF)
A15	30	0.5	1.8	2.4	0.8	2.0	1800	40
A14	30	0.5	1.8	2.4	0.8	2.0	1800	40
A13		0.5	1.8	2.4	0.8	2.0	1800	40
A12	30	0.4	1.8	2.4	0.8	2.0	1800	74
A11	30	0.4	1.8	24	0.8	2.0	1800	74
A10	30	0.4	1.8	24	0.8	2.0	1800	74
A9	30	0.4	1.8	2.4	0.8	2.0	1800	76
A8	30	0.4	1.8	2.4	0.8	2.0	1800	76
A7	30	0.4	1.8	2.4	0.8	2.0	1800	72
A6	30	0.4	1.8	2.4	0.8	2.0	1800	72
A5	30	0.4	1.8	2.4	0.8	2.0	1800	72
A4	30	0.5	1.8	2.4	0.8	2.0	1800	12
A3		0.4	1.8	2.4	0.8	2.0	1800	72
A2	30	0.4	1.8	2.4	0.8	2.0	1800	72
A1	30	0.4	1.8	2.4	0.8	2.0	1800	72
A0	30	0.4	1.8	2.4	0.8	2.0	1800	98
A7RB	30	0.5	0.35	2.7	0.8	2.0	----	120
	30	0.5	12	2.4	0.8	2.0	20	10
WRB	30	0.5	12	2.4	0.8	2.0	20	10
RFSHB	30	0.5	12	2.4	0.8	2.0	20	10
EXROM	30	0.5	12	2.4		---	____	--
ROSCS		0.5	12	2.4	---	---	____	--
MREQB	30	0.5	12	2.4	0.8	2.0	20	10
RDB	30	0.5	12	2.4	0.8	2.0	20	10
MI	30	0.4	1.8	2.4	0.8	2.0	20	10
BE	--	---	---		0.8	2.0	10	12
BUSAK	30	0.4	1.8	2.4	___	___	----	--
WAIT	__	---	---	---	0.8	2.0	----	10
HALT	30	0.4	1.8	2.4	0.8	2.0	----	10
NMI	--	---	___		0.8	2.0	----	10
INT	OPEN COLLECTOR WITH PULL-UP							
R	50	0.4	1.8	2.4	---	---	----	--
G	50	0.4	1.8	2.4	___	---	----	__
B	50	0.4	1.8	2.4	---	---	____	--
VIDEO	0.40 75 ohm COAX							
D0	30	0.4	1.8	2.4	0.8	2.0	20	120
D1	30	---	1.8	2.4				
D2	30	0.4	1.8	2.4	0.8 0.8	2.0 2.0	20 20	120 120
D3	30		1.8	2.4	0.8	2.0		120
D4	30	0.4...					20	
D5	30	0,4 0,4	1.8 1.8	2.4 2.4	0.8 0.8	2.0	20 20	120 120
D6	30	0.4				2.0		
D7	30	0.2	1.8 1.8	2.4 2.4	0.8 0.8	2.0 2.0	20 20	120 120
SPKR/TAPE OUT	500	0.5	0.04	0.3-0.5	---	---	----	__
EAR	15		1.6	2.4	+/- 1.3	+/- 5.0	----	__
SOUND	100	0	---	2.5	-0.3	+5.0	me--	__
y ! ! &	--	---	---	---	0.8	2.0	----	10
	1 uF WITH 220K PULL-UP							
IOA5	-_	---	---	---	_-_	---	----	__

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/separator circuit described here.

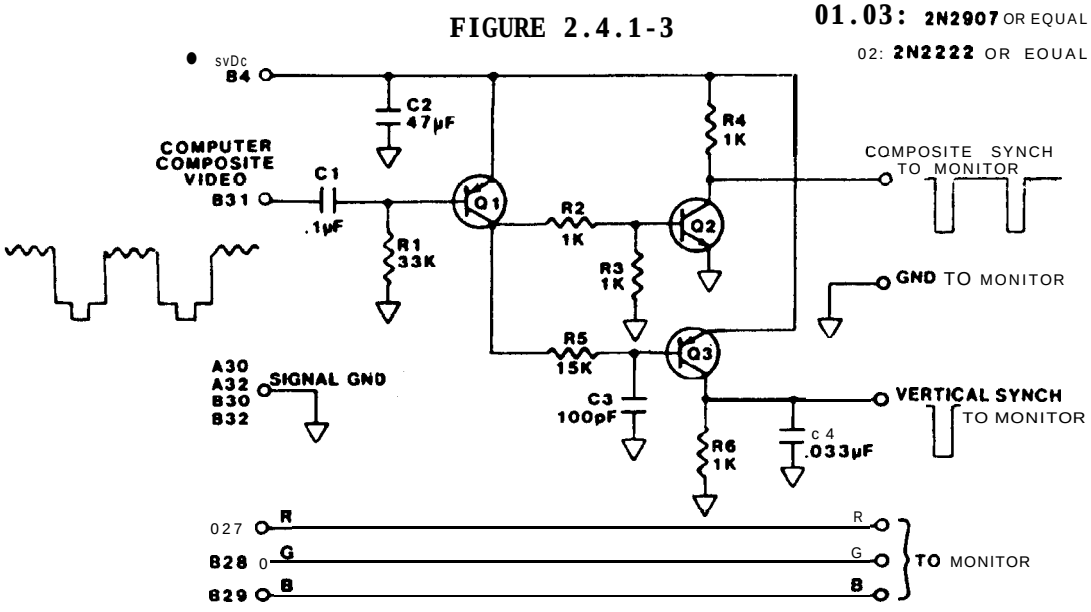
The Schematic of Figure 2.4.1-3 shows the required connections and electronics. Attachment is via the 64-pin keyed P1 connector. Shielding should not normally be required, but ferrite beads are recommended on each wire to minimize EMI, TVI, etc.

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 827, 828, and B29 can be supplied directly to the attached monitor.

FIGURE 2.4.1-3



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.

FIGURE/TABLE	TITLE
Figure 2.4.2-1	J4 Mating PCB Mechanical Requirements
Figure 2.4.2-2	J4 Signal Layout
Table 2.4.2-1	J4 Signal Definition
Table 2.4.2-2	J4 Signal Electrical Characteristics

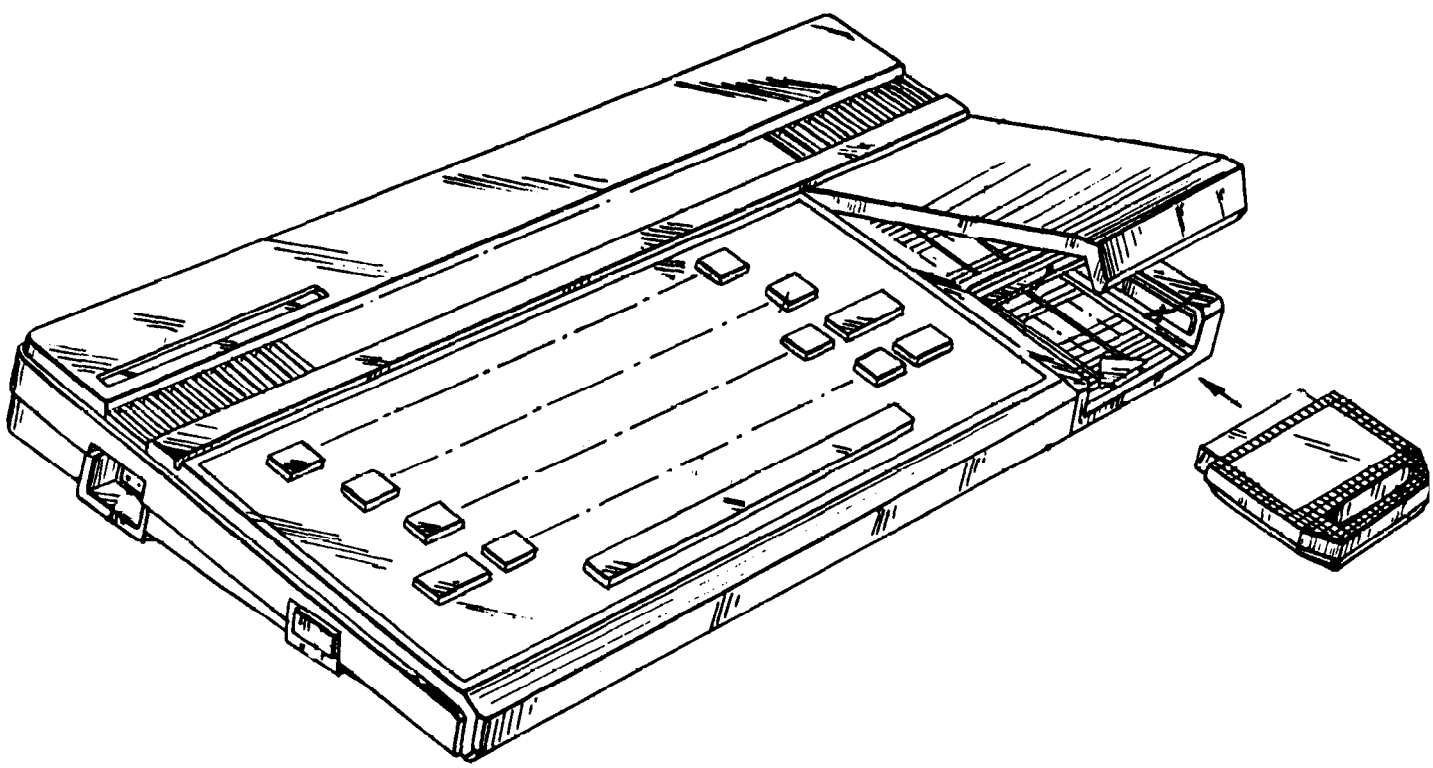
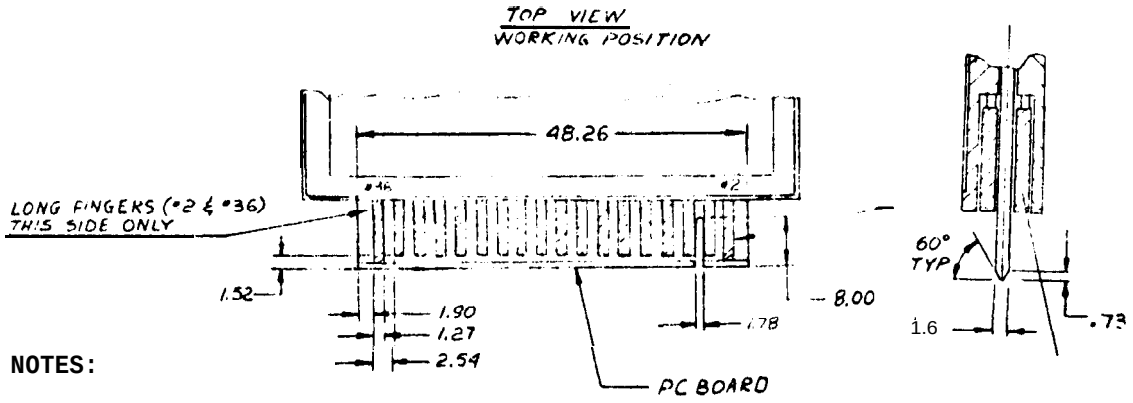


FIGURE 2.4.2-1

J4 MATING PCB MECHANICAL REQUIREMENTS



NOTES:

(1) Circuit Board Material:
FLGFN C62
Cl/lA2A (94V-0)
Copper 1 or 2 sides

(2) Contact Fingers: Min. 10 millionth MIL-G - 45204 Gold over .00005 to .000l0 inch low stress nickel.

(3) Contact Fingers 2 and 36 should be longer than other fingers to latch-up when inserted with power on.

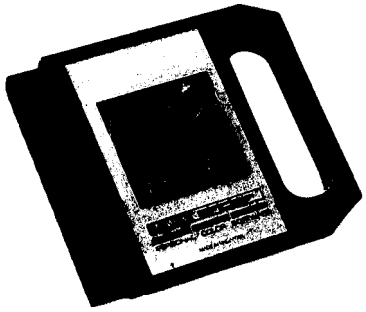


FIGURE 2.4.2-2

J4 SIGNAL LAYOUT

(View from Front)

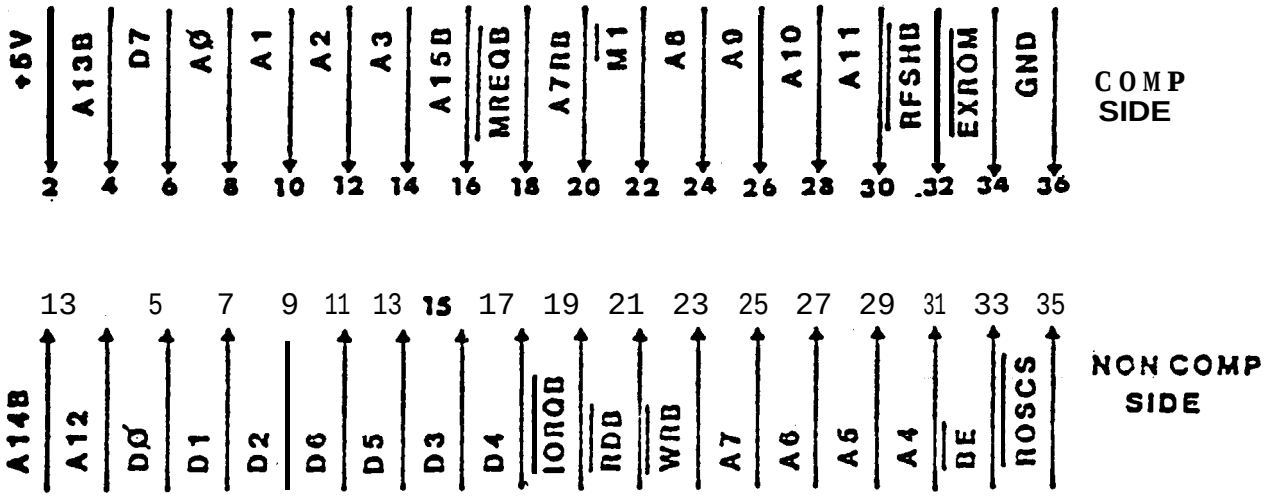


TABLE 2.4.2-1

J4 CONNECTOR SIGNAL DEFINITIONS

PIN #	SIGNAL NAME	DESCRIPTION
1	A14B	Address Bus Bit 14, Buffered
2	+5v	+5 volts DC
3	A12	Address Bus Bit 12
4	A13B	Address Bus Bit 13, Buffered
5	D0	Data Bus Bit 0
6	D7	Data Bus Bit 7
7	D1	Data Bus Bit 1
8	A0	Address Bus Bit 0
9	02	Data Bus Bit 2
10	A1	Address Bus Bit 1
11	D6	Data Bus Bit 6
12	A2	Address Bus Bit 2
13	D5	Data Bus Bit 5
14	A3	Address Bus Bit 3
15	D3	Data Bus Bit 3
16	A15B	Address Bus Bit 15,Buffered
17	D4	Data Bus Bit 4
18	$\overline{\text{MREQB}}$	Memory Request (Active Low),Bfrd.
19	$\overline{\text{IORQB}}$	I/O Request (Active Low),Buffered
20	A7RB	Refresh Address Bit 7, Buffered
21	$\overline{\text{RDB}}$	Read (Active Low), Buffered
22	$\overline{\text{M1}}$	CPU M1 State (Active Low)
23	$\overline{\text{WRB}}$	Write (Active Low), Buffered
24	A8	Address Bus Bit 8
25	A7	Address Bus Bit 7
26	A9	Address Bus Bit 9
27	A6	Address Bus Bit 6
28	A10	Address Bus Bit 10
29	A5	Address Bus Bit 5
30	A11	Address Bus Bit 11
31	A4	Address Bus Bit 4
32	$\overline{\text{RFSHB}}$	Refresh (Active Low), Buffered
33	$\overline{\text{BE}}$	Bank Enable (Active Low)
34	$\overline{\text{EXROM}}$	Extension ROM Enable (Active Low)
35	$\overline{\text{ROSCS}}$	ROS Chip Select (Active Low) (Dock Bank Enable)
36	GND	Ground

TABLE 2.4.2-2

J4 SIGNAL ELECTRICAL CHARACTERISTICS

	OUTPUTS FROM TS2068					INPUTS TO TS2068			
MNEMONIC	CAPACITIVE LOADING MAX (PF)	V(OL) MAX VOLTS	I(LOAD) MAX (MA)	V(OH) MIN VOLTS	I(LOAD)*V(IL) MIN (uA)	MAX VOLTS	V(IH) MIN VOLTS	I IN (MAX) uA	INPUT CAPACITIVE LOADING MAX (PF)
A15B	30	0.5	1.8	2.4	10				
A14B	30	0.5	1.8	2.4	10				
A13B	30	0.5	1.8	2.4	10				
A12	30	0.4	1.8	2.4	10				
All		0.4	1.8	2.4	10				
A10	30	0.4	1.8	2.4	10				
A9	30	0.4	1.8	2.4	10				
A8	30	0.4	1.8	2.4	10				
A7	30	0.4	1.8	2.4	10				
A6	30	0.4	1.8	2.4	10				
A5	30	0.4	1.8	2.4	10				
A4	30	0.4	1.8	2.4	10				
A3	30	0.4	1.8	2.4	10				
A2	30	0.4	1.8	2.4	10				
Al	30	0.4	1.8	2.4	10				
A0	30	0.4	1.8	2.4	I?				
A7RB	30	0.5	0.35	2.7					
ROSCS	30	0.4	1.8	2.4	10				
MREQB		0.5	1.8	2.4	10				
RDB	30	0.5	1.8	2.4	10				
IORQB	30	0.5	12	2.4	10				
WRB	30	0.5	12	2.4	10				
RFSHB		0.5	12	2.4	10				
EXROM	30	0.5	12	2.4	10				
iii	30	0.5	12	2.4	10				
DO	30	0.4	1.8				2.0	15	120
D1	30	0.4	1.8	2.4		0.8	2.0	15	120
D2	30	0.4	1.8	2.4		0.8	2.0	15	120
D3	30	0.4	1.8	2.4		0.8	2.0	15	120
D4	30	0.4	1.8	2.4		0.8	2.0	15	120
D5	30	0.4	1.8	2.4		0.8	2.0	15	120
D6	30	0.4	1.8	2.4		0.8	2.0	15	120
D7	30	0.4	1.8	2.4		0.8	2.0	15	120
Vcc (+5V)	--	5.25	300	4.75					
GND	--	--	---	---					

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 voltaqes should be between 4.0 and 10.0 V p-p.

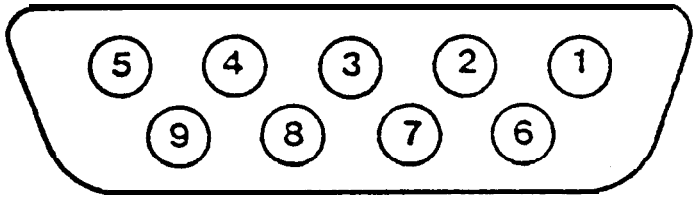
2.4.4 Joystick

The joystick input connectors, one on each side of the TS2068 case, are standard D-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



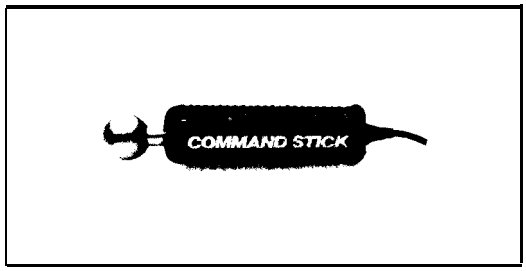


TABLE 2.4.4-1

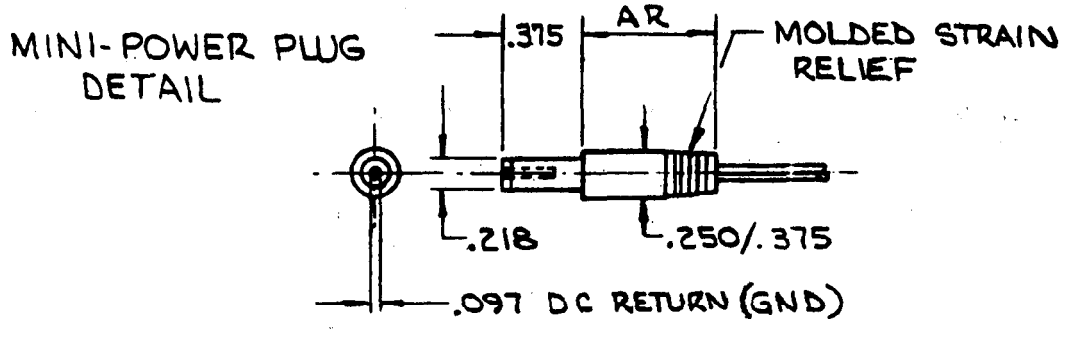
JOYSTICK CONNECTOR SIGNAL ASSIGNMENT

P/N	SIGNAL NAME	I/O PORT BIT	FUNCTION
1	$\overline{\text{DIR1}}$	0	STICK UP
2	$\overline{\text{DIR2}}$	1	STICK DOWN
3	$\overline{\text{DIR3}}$	2	STICK LEFT
4	$\overline{\text{DIR4}}$	3	STICK RIGHT
5	---	----	not used
6	$\overline{\text{BUTTON}}$	7	PUSH BUTTON
&	5v	---	5 VOLT POWER
8	$\overline{\text{READ STROBE}}$	---	ADDRESS BIT 8 OR 9*
9	GND	---	POWER GROUND

*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:



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:

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

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 13 (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:

RESTART	FUNCTION
8	ERROR - Error exit from BASIC (Address on Stack points to Error Number)
15	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).
24	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)

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=lst Byte of Space; HL=Last.

Location 56 (38H) is the entry to service the hardware generated interruption which occurs approximately every l/60 of a second (16.67 ms). Z80 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 fur 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 (3DOOH to 3FFFH). The address of this table minus 256 (lOOH) 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-l 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-l

Extension ROM Interruption Fielder

LOCAT ION	OBJECT CODE		SOURCE CODE		COMMENTS
0038	F5		PUSH	AF	Save AF
0039	F3		DI		Disable Ints.
003A	3AC25C		LD	A,(VIDMOD)	Test Vidmod
003D	A7		AND	A	
003E	00		NOP		
003F	2804		JR	Z,CHK3	Vidmod=0
0041	Fl		POP	AF	Restore AF
0042	C36EFA		JP	INT7	Chunk 7 if Vidmod not 0
0045	Fl	CHK3	POP	AF	Restore AF
0046	C3AE62		JP	INT3	Chunk 3 if Vidmod = 0

3.2.2.3 Video Mode Change Service

The routine CHNG VID takes as input a single byte in Register3 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 6000H-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

VALUE IN A	VIDEO MODE		DESCRIPTION
0	Normal		Primary Display File Only(Close 2nd Display File if Open)
128 (80H)	Dual Screen		Two Display Files Available. Primary Display File Active at Screen.
1	Dual Screen		Two Display Files Available. Second Display File Active at Screen
2	High Resolution Graphics		Primary Display File contains data for 256X192 pixels. Second Display File contains 6144 Attribute Bytes, each one controlling 8X1 pixels. NOTE 1.
	64-Column		The two display files are combined to provide a 64 column X 24 line screen. Even columns are derived from data in the Primary Display File and odd columns from the 2nd Display File. Bits 3-5 of the mode select the ink color which determines the complementary paper color. The Flash and Bright Attributes are fixed at 0; the Border is fixed at the paper color. NOTE 1.
	Ink	Paper	
6	Black	White	
14 (0EH)	Blue	Yellow	
22 (16H)	Red	Cyan	
30 (1EH)	Magenta	Green	
38 (26H)	Green	Magenta	
46 (2EH)	Cyan	Red	
54 (36H)	Yellow	Blue	
62 (3EH)	White	Black	

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

```
                                1  ;                                    ; EXTENSION ROM INTERFACE ROUTINE
=OOFC                           2  R_lAPE       EQU     OOFCH        ; READ TAPE ROUTINE
=0068                           3  W_TAPE       EQU     0068H        ; WRITE TAPE ROUTINE
=OE8E                           4  CHNG_VID     EQU     0E8EH        ; CHANGE VIDEO MODE ROUTINE
=5CC2                           5  VIDMOD       EQU     5CC2H        ; VIDEO MODE SYSTEM VARIA8LE
                                6  ;
                                7  ;
                                8  ;                                    ; CALL READTP WITH REGISTERS SET
                                9                                       ; UP FOR R_TAPE ROUTINE
                               10  ;
0000' 21 OOFC                  11  READTP   LO      HL,R_TAPE        ; ADDRESS TO HL
0003' cc 0020'                 12            CALL    IFRTN            ; ENABLE EXT./EXECUTE QTN
0006' 18 17                    13            JR      EXIT             ; RESTORE HOME BANK AND RETURN
                               14
                               15  :                                    ; CALL WRITETP WITH REGISTERS SET
                               16  :                                    ; UP FOR W_TAPE ROUTINE
                               17  ;
OOO8' 21 0068                  18  WRITTP   LO      HL,W_TAPE        ; ADDRESS TO HL
000B' cc 0020'                 19            CALL    If RTN           ;
OOOE' 18 OF                    20            JR      EXIT
                               21  :
                               22                                      ; CALL CHGVID WITH DESIRED VIDEO
                               23  t                                   ; MODE IN A
                               24  ·
0010' 21 OE8E                  25  CHGVID   LD      HL,CHNG_VID      ; ADDRESS TO HL
1313' F5                       26            PUSH    AF               ; SAVE VIDEO MODE
0014' CD 0020'                 27            CALL    IFQTN
                               28  :
                               29  :                                    ; COMPENSATE FOR 'BUG' IN
                               30  :                                    ; CHNG_VID RTN.WHICH SETS
                               31  :                                    ; VIDMOD=0 INSTEAD OF 80
                               32  :                                    ; WHEN BOTH DISPLAY FILES
                               33                                      ; ARE OPEN
                               34  ;
                               3s  ;
0017' F1                       36            POP     AF               ; TEST VIDEO MODE
0018' FE 80                    37            CP      80H              ; TEST IF 80
00111' 20 03                   38            JR      NZ,EXIT
001C' 32 5CC2                  39            LO      (VIDMOD),A       ; SET VIDMOD=80H
OOlF' 3A 002C'                 40  EXIT     LO      A,(HSSAVE)       ; GET PREV. HOR.SEL.
0022' D3 F4                    Cl            OUT     (0F4H),A         ; RESTORE
0024' DB FF                    42            IN      A,(0FFH)         ; READ PORT FF
0026' CB 8F                    43            RES     7,A              ; TURN OFF RCM SEL.
0028' 03 FF                    44            OUT     (0FFH),A
002A' FB                       45            EI
0023' C9                       46            RET
                               47  ;
002C' 00                       48  HSSAVE   DEFB    0                ; SAVE HOR.SEL. (PORT OF4H)
                               49  ;
0020' F3                       53  IFRTN    DI                       ; MASK INTERRUPTIONS
OOZE' F5                       54            PUSH    AF               ; PRESERVE REG. A
002F' DB FF                    55            IN      A,(0FFH)         ; EXT.ROM SELECT BIT
0031' CB FF                    56            SET     7,A              ; SEL. EXT.ROM
0033' 03 FF                    57            OUT     (0FFH),A
0035' DB F4                    58            IN      A,(0F4H)         ; HORIZONTAL SELECT FCR DOCK/EXT
0037' 32 002C'                 59            LD      (HSSAVE),A       ; SAVE
003A' 3E 01                    60            LD      A,1              : SELECT CHUNK 0 IN EXT.ROM
003C' 03 F4                    61            OUT     (0F4H),A         ;
003E' F1                       62            POP     AF               ; RESTORE REG. A
003F' E9                       63            JP      (HL)             ; EXECUTE TARGET ROUTINE AND
                               64  :                                    : RETURN TO CALLER CF IFRTN
                               65            END
```

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 (STRMNM - 23755 (5CCB)) 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 0F7C0H to 0F8BFH, and modifies the Stack Pointer and MSTBOT (0F8C0H), 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.

YOTE: 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 he 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-l 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-l

OS RAM SERVICE ROUTINES

LABEL	SERVICE COOE (Decimal)	LOCATION H.	H.	DESCRIPTION
GET_WORD	_	6316	FAD6	Returns in HL the word from the address in HL in the bank specified in B.
PUT_WORD		6336	FAFB	Writes the word in DE to the address in HL in the bank specified in B.
GET STATUS	14	6405	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.
GET_CHUNK		644D	FCOD	Returns a single byte mask in A with all bits 0 except the one corresponding to the chunk for the address in HL.
GET NUMBER	15	645E	FClE	Returns in Reg. A the bank number currently controlling the address in HL.
BANK ENABLE	-	6499	FC59	Enables the memory selected (Horizontal Select byte - low active) in the specified bank. (Bank # in B; Mem.Sel.in C)
GOT0 BANK	-	6572	F032	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 GOT0 BANK. (Return address is discarded).
CALL BANK	-	65D0	FD90	Like GOT0 BANK except saves current hankstatus, 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-bits) following by PRM_IN (16 bits) as described for the Function Dispatcher.

TABLE 3.3.4-1

OS RAM SERVICE ROUTINES
(continued)

LABEL	SERVICE CODE (Decimal)		LOCATION		DESCRIPTION
XFER	BYTES	-	6722	FEE2	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: (O=Asscending -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: F9C0H

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                    PUSH HL
```

Addendum To TS 2068 Function Dispatcher Services:
On page 84, COLOR and HIFLSH (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. HIFLSH can be accessed similarly by coding CALL #2410.

TABLE 3.3.4-2

TS 2068 FUNCTION DISPATCHER SERVICES

SERVICE	SERVICE CODE	DESCRIPTION
	1 - 13 (1-0DH)	Reserved
GET STATUS	14 (0EH)	Returns Memory Selection (Low Active) in C for Bank # in B
GET NUMBER	15 (0FH)	Returns Bank # in A for Address in HL
	16-24 (10-18H)	Reserved
UPD K	25 (19H)	Process Keyboard Input (See Section 4.1 .1)
PARP	26 (1AH)	Generates DE+1 Cycles of a Tone having the Period 8N+236 to 8N+246 T-States. HL=N. (See 4.4)
BEEP	27 (1BH)	BEEP Command - processes parameters on Calculator Stack. Exits via PARP. (See 4.4)
K_DUMP	28 (1CH)	COPY Command. Dumps Primary Display File to Printer. (See 4.1.3)
SENDTV	29 (1DH)	Char.Output to Screen/Printer. Character Code in A. (See 4.1.2)
SETAT	30 (1EH)	Set Print Position to value in BC. B=Line No. (0-23); C=Column No. (0-31)
ATTBYT	31 (1FH)	Set Attribute Byte for Display File Adrs. in HL using ATTR_T, MASK_T and P-FLAG.
R ATTS	32 (20H)	Permanent Attribute Info. to Temporary Attribute Variables
CLLHS	33 (21H)	Clear Lower Screen (Primary Display File)
CLS	34 (22H)	Clear Entire Screen(Primary Display File)
DUMPPR	35 (23H)	Print/Clear Print Buffer. (See 4.1.3)

TABLE 3.3.4-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
PRSCAN	36 (24H)	Send (an 32 bytes) to Printer. Pixel Data Address in HL Number of Scans remaining in B (=1-8). (See 4.1.3)
DESLUG	37 (25H)	Remove Number Slugs from Edit Line Buffer (Address in HL)
K NEW	38 (26H)	NEW command. See Fig. 1.1-4
INIT	39 (27H)	Initialize: DE=Maximum RAM Address. A=0 for Power-On; = -1 (FFH) for NEW. (See Fig.l.l-4)
INCH	40 (28H)	Input Character to A from currently Selected Channel. Returns NC if no input.
SELECT	41 (29H)	Select Channel (Stream) - # in A. (See 4.1)
INSERT	42 (2AH)	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=adrs.of last byte of inserted space; HL=adrs.of byte before first.
RESET	43 (2BH)	Reset Calculator Stack. Sets (STKEND) =(STKBOT) and (MEM)=MEMBOT (5C92H).
CLOSE	44 (2CH)	CLOSE # Command. Channel # on Calculator Stack.
CLCHAN	45 (2DH)	Close Channel. BC=Value from STRMS (Index into CHANS).
OPEN	46 (2EH)	OPEN # Command. Channel # an4 Device Spec. on Calculator Stack
OPCHAN	47 (2FH)	Open Channel. Device Spec. on Calculator Stack. DE=pointer into STRMS based on Ch.#. (See 4.1 for more info. on OPEN and CLOSE)

TABLE 3.3.4-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
CAT	48 (30H)	CAT Command (Not Applicable)
ERASE	49 (31H)	ERASE Comnand (Not Applicable)
FORMAT	50 (32H)	FORMAT Command (Not Applicable)
MOVE	51 (33H)	MOVE Command (Not Applicable)
FLASHA	52 (34H)	Flash Char.in A to Screen. (Calls SENDTV; assumes Lower Screen selected. Used to Flash Cursor.)
FIND_L	53 (35H)	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).
SUBL IN	54 (36H)	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-l 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 (0DH).

TABLE 3.3.4-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
RECLEN	55 (3/H)	Returns in BC the lenqth 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.
DELREC	56 (38H)	Delete record pointed to by HL having length BC from Program or Variables memory. Updates affected system variables.
PUT BC	57 (39H)	Converts number in BC from binary to ASCII and outputs to currently selected channel, If BC less than 0, outputs a 0.
SYNTAX	58 (3AH)	Check syntax of command or program line in Edit Line Buffer (E LINE). ERR NR= -1 if no errors, otherwisecontains Error Number-l.
EXCUTE	59 (3BH)	Execute command(s) from Edit Line buffer.
FOR	60 (3CH)	FOR command. *
STOP	61 (3DH)	STOP command. Does RESTART 8 with Error No. 9.
NEXT	62 (3EH)	NEXT command. *
READ	63 (3FH)	READ command. *
DATA	64 (40H)	DATA statement. *
RESTBC	65 (41H)	RESTORE command - Line No. in BC
RAND	66 (42H)	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.

TABLE 3.3.3-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
CON'T	67 (43H)	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.
JUMP	68 (44H)	Jump to Line - Loads Line Number from Calculator Stack to NEWPPC and sets NSPPC to 0 and returns.
FIX_Ul	69 (45H)	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.
FIX U	70 (46H)	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.
CLEAR	71 (47H)	CLEAR command. Processes parameter on Calculator Stack to value in BC for CLR BC.
CLR BC	72 (48H)	Value in BC is new RAMTOP. Deletes Variables, clears screen, and Calculator Stack, etc.
GO SUB	73 (49H)	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.).

TABLE 3.3.3-Z

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
CHK SZ	74 (4AH)	Checks if room for BC + 80 (50H) bytes between (STKEND) and (RAMTOP). Addition of 80 bytes is "left-over" from Spectrum to guarantee minimum machine stack where the stack was at the top of RAM. Error 4 if not enough room.
RETURN	75 (4BH)	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).
PAUSE	76 (4CH)	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.)
BREAK?	77 (4DH)	Reads BREAK key. Returns NC if it is pressed and ON ERROR is not active.
DEF	78 (4EH)	Define Function.*
K LPR	79 (4FH)	LPRINT - Selects Channel 3 and processes items in LPRINT statement for output via WRCH.
K PRIN	80 (50H)	PRINT - Selects Channel 2 and processes items in PRINT statement for output via WRCH (same code used for K_LPR).
P_SEQ	81 (51H)	Code used by K LPR and K PRIN to process output-data and controls in BASIC statement (address in CH ADD).
INPUT	82 (52H)	INPUT command. Selects Channel 1 and processes I/O for Keyboard/Lower Screen using a buffer at (WORKSP) for input. *

TABLE 3.3.3-Z

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
I_SEQ	83 (53H)	Code used by INPUT to process input items and controls in BASIC statement (address in CH ADD).
NOTKB?	84 (54H)	Returns Z if current channel is Keyboard/Lower Screen (device specification="K").
COLOR	85 (55H)	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.
HIFLSH	86 (56H)	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.
SCRMBL	87 (57H)	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=lefthand or most significant bit. Error B if Y is greater than 175.
PLOT	88 (58H)	PLOT command. Processes X/Y parameters on the Calculator Stack to BC for plotting of pixel via PLOTBC.
PLOTBC	89 (59H)	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.
GET_XY	90 (5AH)	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.

TABLE 3.3.3-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
CIRCLE	91 (5BH)	CIRCLE command. Calculates successive plot positions from the parameters in the BASIC statement. *
DRAW	92 (5CH)	DRAW command. Calculates successive plot positions from the parameters in the BASIC statement. *
DRAW L	93 (5DH)	Plots a straight line from current position (COORDS) based on parameters from Calculator Stack (X,Y). *
EXPRN	94 (5EH)	Evaluates expression in BASIC program line (CH ADD), putting value on Calculator Stack. *
F SCRN	95 (5FH)	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.
F ATTR	96 (60H)	ATTR function. Returns attribute byte value controlling screen pixel position based on parameters on Calculator Stack (X,Y).
RND	97 (61H)	RND function. Uses value in SEED to generate a pseudo-random number which is placed on the Calculator Stack (Floating Point number).
F PI	98 (62H)	PI function. Places value of PI on Calculator Stack.

TABLE 3.3.3-2

TS **2068 FUNCTION DISPATCHER SERVICES**
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
F_INKY	99 (63H)	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.
FIND N	100 (64H)	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. *
PSHSTR	101 (65H)	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.
PAEDCB	102 (66H)	Same code as for PSHSTR but preserves state of bit NO of FLAGS (Bit 6).
LET	103 (67H)	LET command. Processes existing or creates new variables. *
POPSTR	104 (68H)	Pop String - Pops end of Calculator Stack ((STKNXT)-1 through (STKNXT)-5) to Regs. BCDEA, adjusting (STKNXT) downwards.
DIM	105 (69H)	DIM statement. Creates or initializes numeric or string arrays. *
STKUSN	106 (6AH)	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).

TABLE 3.3.3-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
STK A	107 (6BH)	1-byte unsigned integer in A to top of Calculator Stack (binary to floating point). Loads 0 to B and A to C, then executes STK BC.
STK BC	108 (6CH)	2-byte unsigned integer in BC to top of Calculator Stack (binary to floating point).
ININT	109 (6DH)	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.
FP2BC	110 (6EH)	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.
FP2A	111 (6FH)	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.
OUTPUT	112 (70H)	outputs number on top of Calculator Stack to currently selected channel via WRCH. (Converts from floating point to ASCII.)
		Full explanation of the following Calculator Routines is beyond the scope of this document.
SUB	113 (71H)	Subtract floating point format numbers (HL) minus (DE). (DE) assumed to be (HL) + 5.

TABLE 3.3.3-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
ADD	114 (72H)	Add (HL) + (DE). See SUB.
MULT	115 (73H)	Integer multiply HL * DE. Returns C if overflow.
TIMES	116 (74H)	Floating Point Multiply (HL) * (DE).
DIVIDE	117 (75H)	Floating Point Divide (HL)/(DE).
TRUNC	118 (76H)	Truncates a floating point number (HL) towards zero to an integer. Assumes (DE) = (HL) + 5.
FLOAT	119 (77H)	Converts number (HL) to floating point format. Assumes HL points to an integer in 5-byte format.
INTDIV	120 (78H)	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.
INT	121 (79H)	Replaces the top of the Calculator Stack by its integer part. Returns with HL = top of Calc. Stack and DE = next free space.
EXP	122 (7AH)	Replaces the top of the Calculator Stack, X, by EXP(X). Returns with DE and HL = Calc.Stack Pointers.
LN	123 (7BH)	Replaces the top of the Calculator Stack by its natrual logarithm. Returns DE and HL = Calc.Stack Pointers.
ANGLE	124 (7CH)	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 +l and the SIN X = SIN (PI/2 * Y).

TABLE 3.3.3-Z

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
cos	125 (7DH)	Replaces the top of the Calculator Stack by its COSINE.
SIN	126 (7EH)	Replaces the top of the Calculator Stack by its SINE.
TAN	127 (7FH)	Replaces the top of the Calculator Stack by its TANGENT.
ATN	128 (80H)	Replaces the top of the Calculator Stack by its inverse TANGENT.
ASN	129 (81H)	Replaces the top of the Calculator Stack by its inverse SINE.
ACS	130 (82H)	Replaces the top of the Calculator Stack by its inverse COSINE.
ROOT	131 (83H)	Replaces the top of the Calculator Stack by its Square Root.
TO THE	132 (84H)	Replaces the top two numbers on the Calculator Stack (X, Y) by x**y.
RDCH	133 (85H)	Wait for character from currently selected channel (calls INCH). Returns character code in A. See 4.1.1.
SENDCH	134 (86H)	Write character whose code is in A to currently selected output channel. See 4.1.2.
WRCH	135 (87H)	See 3.2.1.1, RESTART 16.
K-SCAN	136 (88H)	Keyboard Scan. See 4.1.1

TABLE 3.3.3-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
P LFT	137 (89H)	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.)
P_RT	138 (8AH)	Outputs a space to currently selected device.
P_NL	139 (8BH)	End-of-Line. Sets current position to start of next line if screen, or outputs printer buffer if printer.
PUTMES	140 (8CH)	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.
K_CLS	141 (8DH)	CLS command. Executes both CLS and CLLHS.
SCRL	142 (8EH)	Scrolls entire screen (primary display file) up 1 line.
F_PNT	143 (8FH)	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.
DRAWLN	144 (90H)	Same as DRAW L but enter with BC register containing coordinates, B=Y and C=X.
PUT_LN	145 (91H)	Output Line Number as 4 digits, right aligned and space filled to currently selected output channel. HL points to MSB of Z-byte Line Number.

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.

	Channel/ Stream #	Device Specification	Command/Function
RESERVED	-3	"K"	Keyboard/Lower Screen
RESERVED	-2	"S"	Main Screen
RESERVED	-1	"R"	RAM Write (not used)
	0	"K"	Output to Lower Screen
	1	"K"	INPUT command
	2	"S"	PRINT/LIST commands
	3	"P"	LPRINT/LLIST commands

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 followinq values:

LOCATION	VALUE	
5C10	0100 (Channel -3)	(Copied from SMINIT in module EDIT of the Home ROM)
5c12	0600 (Channel -2)	
5c14	0600 (Channel -1)	
5C16	0100 (Channel 0)	
5C18	0100 (Channel 1)	
5C1A	0600 (Channel 2)	
5C1C	1000 (Channel 3)	
5C1E	0000 (Channel 4)	
.	. .	
5C34	0000 (Channel 15)	

This table is accessed using ((Ch.# * 2) + 16H) as an index added to 5COOH. 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 initilization 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

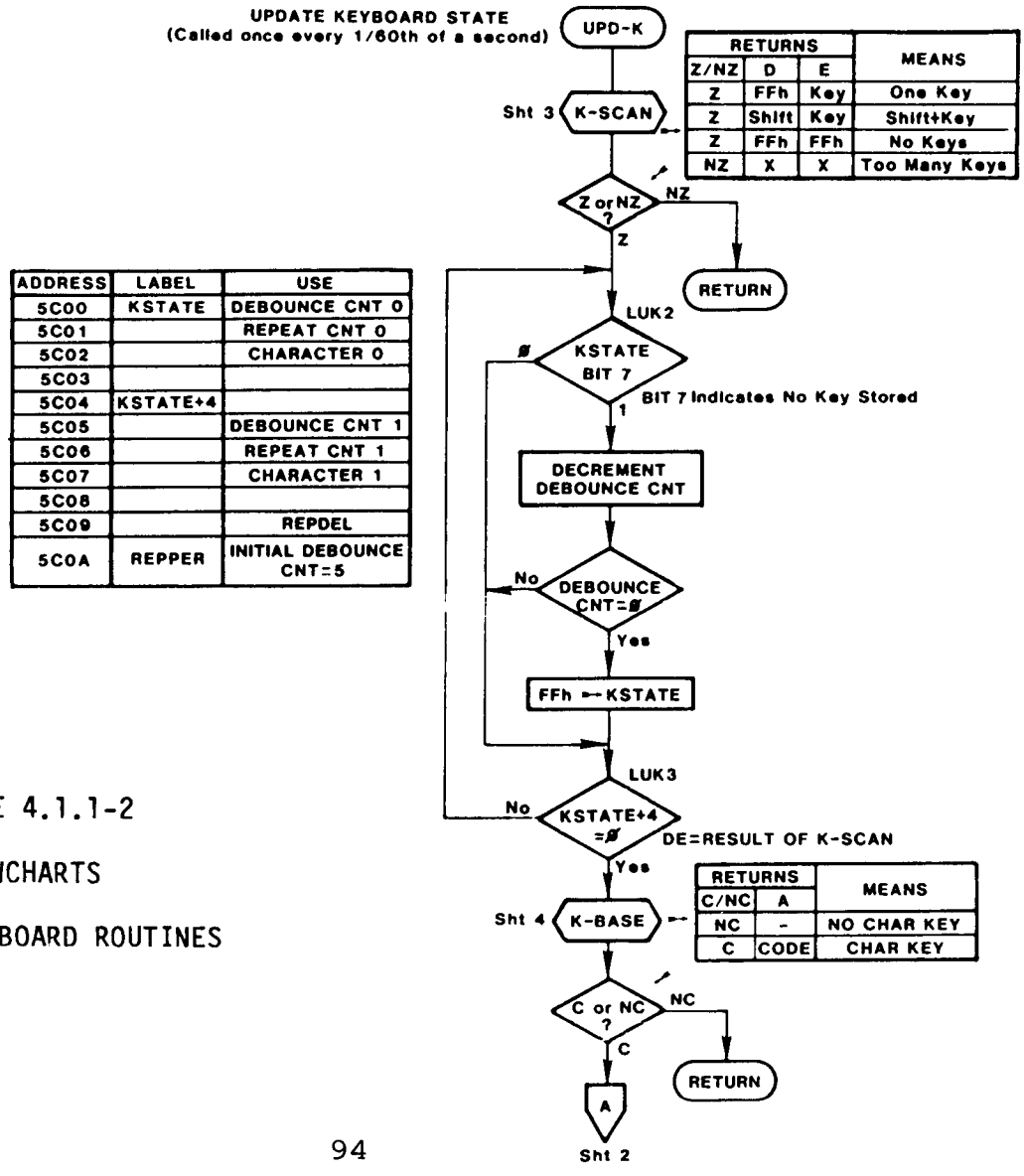
The low-level routines supporting keyboard input are executed every l/60 of a second out of the Interruption Handler (Location 56 (38H)). The controlling routine is labelled UPD K. This routine calls K SCAN to determine if any key(s) are currently being depressed, controls the debouncing and repeat algorithms, 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-l 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=l, the contents of LAST K are returned to the requestor.

FIGURE 4.1.1-1

TS 2068 MODE CONTROLS

System Variable	Location	Description
MODE	23617(5C41H)	Value 0 = "K" or "L" Mode 1 = "E" Mode 2 = "G" Mode
FLAGS	23611(5C3BH)	If MODE = 0 then: Bit 3 = 0 for "K" Mode = 1 for "L" Mode
FLAGS2	23658(5C6AH)	If in "L" Mode then: Bit 3 = 0 CAPS Lock Off = 1 CAPS Lock On

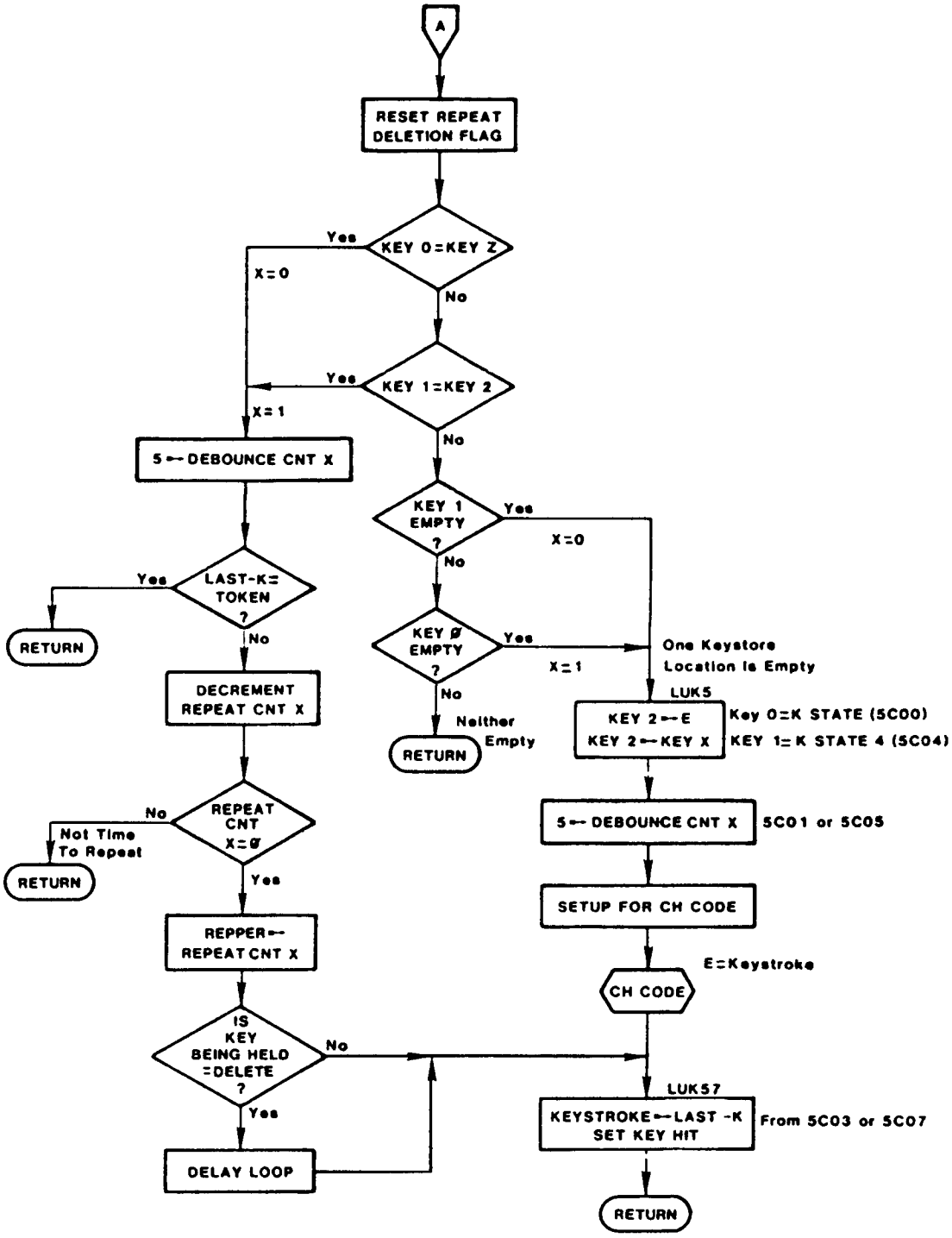


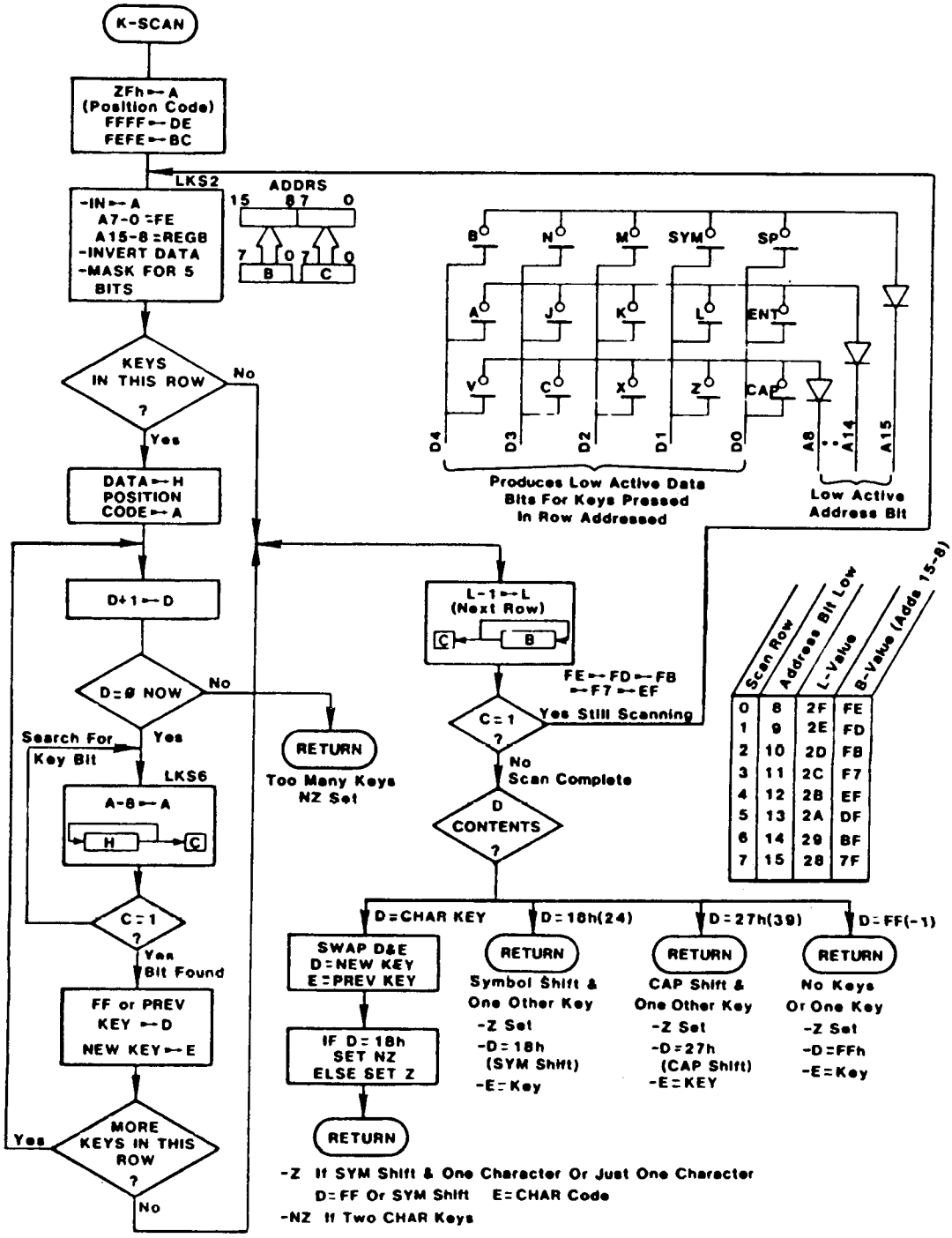
ADDRESS	LABEL	USE
5C00	KSTATE	DEBOUNCE CNT 0
5C01		REPEAT CNT 0
5C02		CHARACTER 0
5C03		
5C04	KSTATE+4	
5C05		DEBOUNCE CNT 1
5C06		REPEAT CNT 1
5C07		CHARACTER 1
5C08		
5C09		REPDEL
5C0A	REPPER	INITIAL DEBOUNCE CNT=5

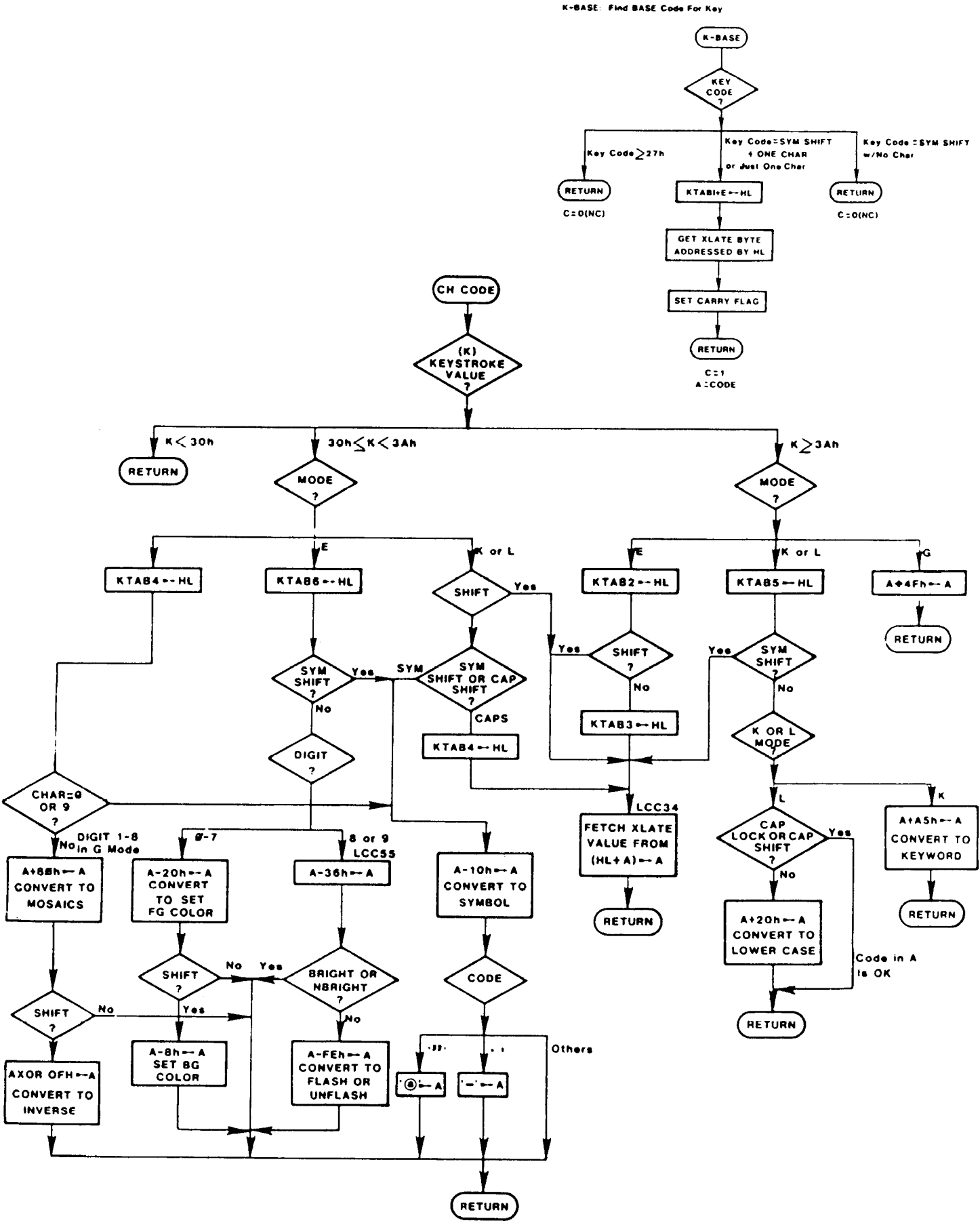
FIGURE 4.1.1-2

FLOWCHARTS

TS 2068 KEYBOARD ROUTINES


A
RESET REPEAT DELETION FLAG
KEY 0=KEY Z
Yes
X=0
No
KEY 1=KEY 2
Yes
X=1
No
5 ← DEBOUNCE CNT X
KEY 1 EMPTY ?
Yes
X=0
No
LAST-K= TOKEN ?
Yes
RETURN
No
KEY Ø EMPTY ?
Yes
X=1
One Keystore Location Is Empty
No
Neither Empty
RETURN
DECREMENT REPEAT CNT X
LUK5
KEY 2 ← E
KEY 2 ← KEY X
Key 0=K STATE (5C00)
KEY 1= K STATE 4 (5C04)
REPEAT CNT X=Ø ?
No
Not Time To Repeat
RETURN
Yes
5 ← DEBOUNCE CNT X
5C01 or 5C05
SETUP FOR CH CODE
REPPER ← REPEAT CNT X
E=Keystroke
CH CODE
IS KEY BEING HELD =DELETE ?
No
Yes
LUK57
KEYSTROKE ← LAST -K
SET KEY HIT
From 5C03 or 5C07
DELAY LOOP
RETURN


K-SCAN
ZFh ← A
(Position Code)
FFFF ← DE
FEFE ← BC
LKS2
-IN ← A
A7-0 = FE
A15-8 = REGB
-INVERT DATA
-MASK FOR 5
BITS
ADDRS
15 8 7 0
7 0 7 0
B C
KEYS
IN THIS ROW
?
No
Yes
DATA ← H
POSITION
CODE ← A
D+1 ← D
D = Ø NOW
No
Yes
Search For
Key Bit
LKS6
A-8 ← A
H
C
C = 1
?
Yes
Bit Found
FF or PREV
KEY ← D
NEW KEY ← E
MORE
KEYS IN THIS
ROW
?
Yes
No
RETURN
Too Many Keys
NZ Set
L-1 ← L
(Next Row)
C
B
FE ← FD ← FB
← F7 ← EF
C = 1
?
Yes Still Scanning
No
Scan Complete
D
CONTENTS
?
B N M SYM SP
A J K L ENT
V C X Z CAP
D4 D3 D2 D1 D0
A8 A14 A15
Produces Low Active Data
Bits For Keys Pressed
In Row Addressed
Low Active
Address Bit
Scan Row
Address Bit Low
L-Value
B-Value (Adds 15-8)
0 8 2F FE
1 9 2E FD
2 10 2D FB
3 11 2C F7
4 12 2B EF
5 13 2A DF
6 14 29 BF
7 15 28 7F
D = CHAR KEY
D = 18h(24)
D = 27h(39)
D = FF(-1)
SWAP D&E
D = NEW KEY
E = PREV KEY
IF D = 18h
SET NZ
ELSE SET Z
RETURN
Symbol Shift &
One Other Key
-Z Set
-D = 18h
(SYM Shift)
-E = Key
RETURN
CAP Shift &
One Other Key
-Z Set
-D = 27h
(CAP Shift)
-E = KEY
RETURN
No Keys
Or One Key
-Z Set
-D = FFh
-E = Key
RETURN
-Z If SYM Shift & One Character Or Just One Character
D = FF Or SYM Shift E = CHAR Code
-NZ If Two CHAR Keys


K-BASE: Find BASE Code For Key
K-BASE
KEY CODE ?
Key Code≥27h
Key Code=SYM SHIFT + ONE CHAR or Just One Char
Key Code=SYM SHIFT w/No Char
RETURN
C=0(NC)
KTABLE←HL
GET XLATE BYTE ADDRESSED BY HL
SET CARRY FLAG
RETURN
C=1
A=CODE
CH CODE
(K) KEYSTROKE VALUE ?
K<30h
RETURN
30h≤K<3Ah
MODE ?
K≥3Ah
MODE ?
E
K or L
KTAB4←HL
KTAB6←HL
SHIFT
Yes
SYM SHIFT ?
Yes
SYM
SYM SHIFT OR CAP SHIFT ?
No
CAPS
KTAB4←HL
DIGIT ?
CHAR=0 OR 9 ?
No
DIGIT 1-8 in G Mode
0-7
8 or 9
LCC55
A+88h←A CONVERT TO MOSAICS
A-20h←A CONVERT TO SET FG COLOR
A-36h←A
A-10h←A CONVERT TO SYMBOL
SHIFT ?
No
Yes
BRIGHT OR NBRIGHT ?
CODE
SHIFT ?
No
Yes
AXOR 0FH←A CONVERT TO INVERSE
A-8h←A SET BG COLOR
A-FEh←A CONVERT TO FLASH OR UNFLASH
'@'←A
' '←A
Others
RETURN
E
K or L
G
KTAB2←HL
KTAB5←HL
A+4Fh←A
RETURN
Yes
SHIFT ?
No
KTAB3←HL
Yes
SYM SHIFT ?
No
K OR L MODE ?
LCC34
FETCH XLATE VALUE FROM (HL+A)←A
RETURN
L
K
CAP LOCK OR CAP SHIFT ?
Yes
No
A+20h←A CONVERT TO LOWER CASE
Code in A is OK
A+A5h←A CONVERT TO KEYWORD
RETURN
RETURN

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 lines1 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-l00H) 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

Character Set	No.of Chars.	Char.Codes	Location
Standard	96	32-127 (20-7FH)	Home ROM (3D00-3FFFH) (Address-100H in CHARS)
Std.Graphics	16	128-143 (80-8FH)	Dynamically Generated by Software
User Defined Graphics	21	144-164 (90-A4H)	Home RAM (Address in UDG)

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 oasses 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 nexfscreen 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

BASIC Parameters	Internal Representation	
Line/Row 0	24 (18H)	
1	23 (17H)	
.	.	UPPER SCREEN
21	3	
22	2	LOWER SCREEN
23	1	
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=l, 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 (5B00H) and is 256 (l00H) 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 (ODH), 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), one 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 genrating 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-l 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=O. 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 (O 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, Roth 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 hit 3 = 0. This creates a final transition on the tape followina a write ooeration. 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-l for a definition of HDLEN.

FIGURE '4.2-l

TAPE HEADER FORMATS

	HDTYPE(1)	HDNAME(10)	HDLEN (LSB/MSB)	HDADD (LSB/MSB)	HDVARS (LSB/MSB)
PROGRAM	0	up to 10 ASCII Chars.	Length of Program + Variables (E_LINE - (PROG)	Starting Line No. or 8000H E.G.: 0500=Line 5 or 0080H if no Line No.	Length of Program = Offset to Variables) (VARS) - (PROG)
NO.ARRAY	1	"	Length Field from Data Structure	LSB=00 MSB=Array ID 7.....0 100 . . . . . (ASCII - 60H)	N/A(=0)
CHAR.ARRAY	2	"	Length Field from Data Structure	LSB=00 MSB=Array ID 7.....0 110 . . . . . (ASCII - 60H)	N/A(=0)
CODE (BINARY)	3	"	Length Specified in SAVE	Address Specified in SAVE	N/A (=0)

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 280 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,

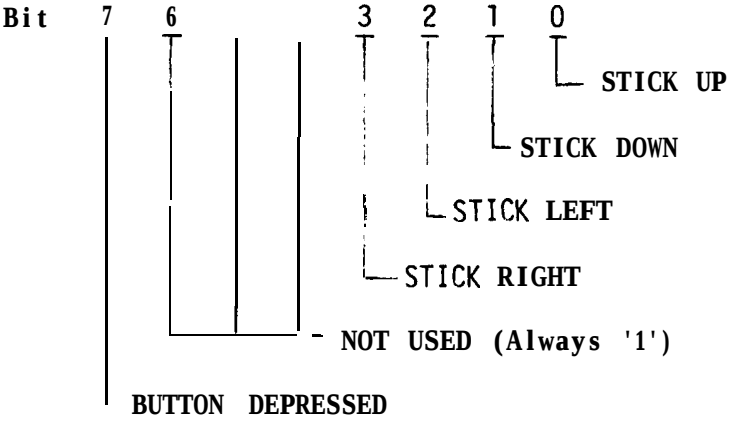
Sample routine:

```
GETJOY  LD    A,0EH          Load A = 14
        OUT   A,(0F5H)       Address the joystick port
        LD    B,playerno
        LD    C,0F6H         Data Port address to C
        IN    A,(C)          Joystick data to A
        CPL                  Complement to High Active
        AND   8FH            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



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 ROY. 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 (SOUND)

The SOUND 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:

Location	Description
0000	Not Used
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 Sank 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 hy the TS 2068 and interruptions are enahled 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, hut 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 hit 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 hy writing to Port OF4H with each bit set to 1 for the corresponding chunk to he enabled in the Dock Bank (high active format) or 0 to he enabled in the Home Bank. Thus, an LROS may contain code in Chunk 3, hut 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 hank 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 enabl inq 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:

Location	Description
32768 (8000H)	Language Type 1 = BASIC [and machine code] 2 = Machine code only (Any other value will result in Error S, Missing LROS)
32769 (8001H)	Cartridge Type 2 = AROS
32770/32771 (8002/8003H)	Starting Address(LSB/MSB) BASIC AROS = Addrs. of First Program Line Machine Code AROS = Addrs. of First Z80 Instruction
32772 (8004H)	Memory Chunk Specification Bits 0-7 represent Chunks 0-7 respectively in the Dock Bank in low active format as follows: 0 if in use 1 if not in use NOTE: Bits 0-3 must he set to 1 for proper execution.
32773 (8005H)	Autostart Specification 0 = No Autostart 1 = Autostart
32774/32775 (8006/8007H)	Number of bytes of RAM to be Reserved for Machine Code Variables (LSB/MSB - 0100H=1 byte Reserved; 0002H=512 bytes Reserved.

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, 1.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 Rank 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 proqram 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 ore-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, althouqh internal flags have been set noting that a BASIC AROS is present. The cartridge will be started when you execute a RUN or GOT0 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 he 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 (7B00H) 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 Sank.

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-l 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

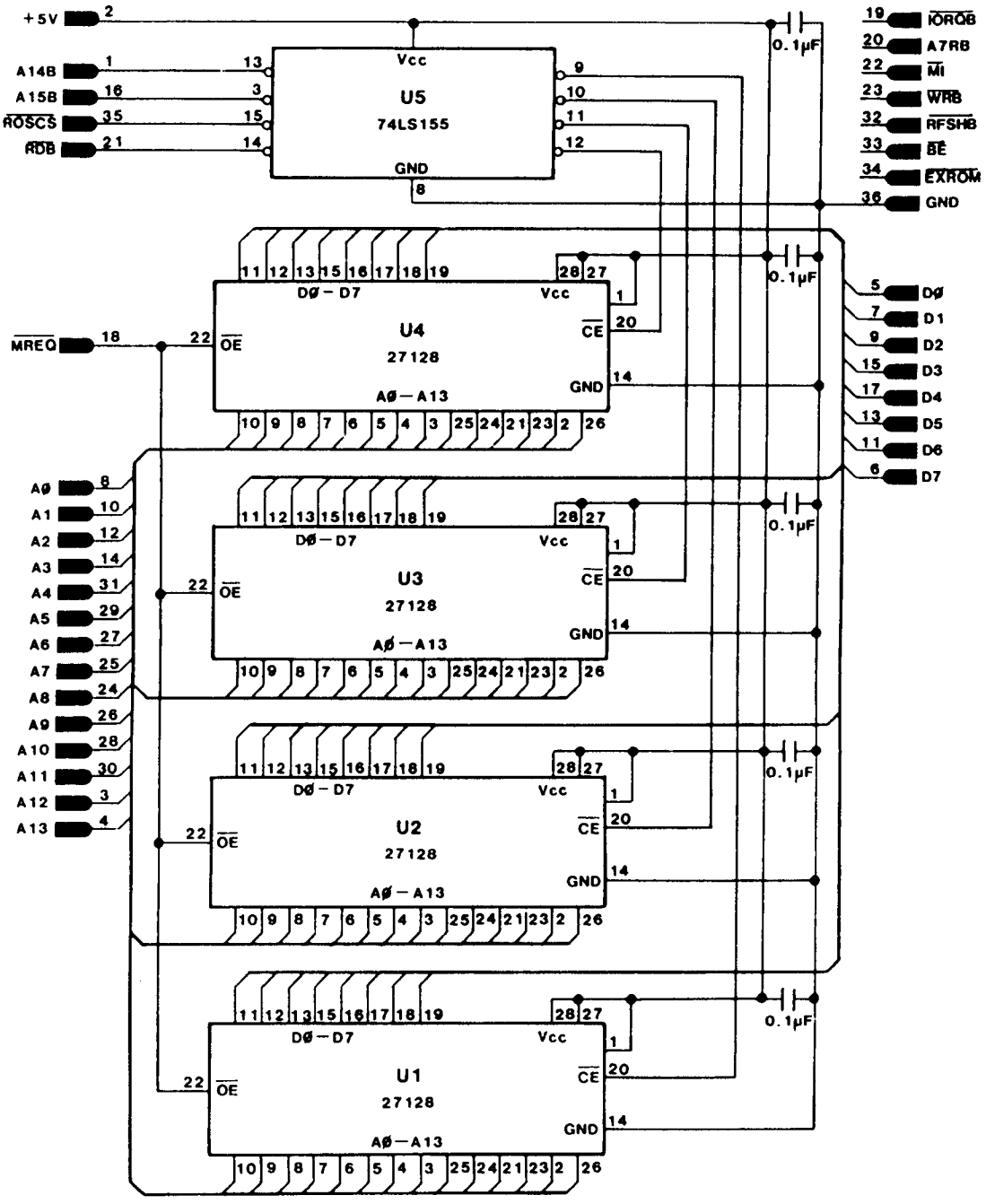
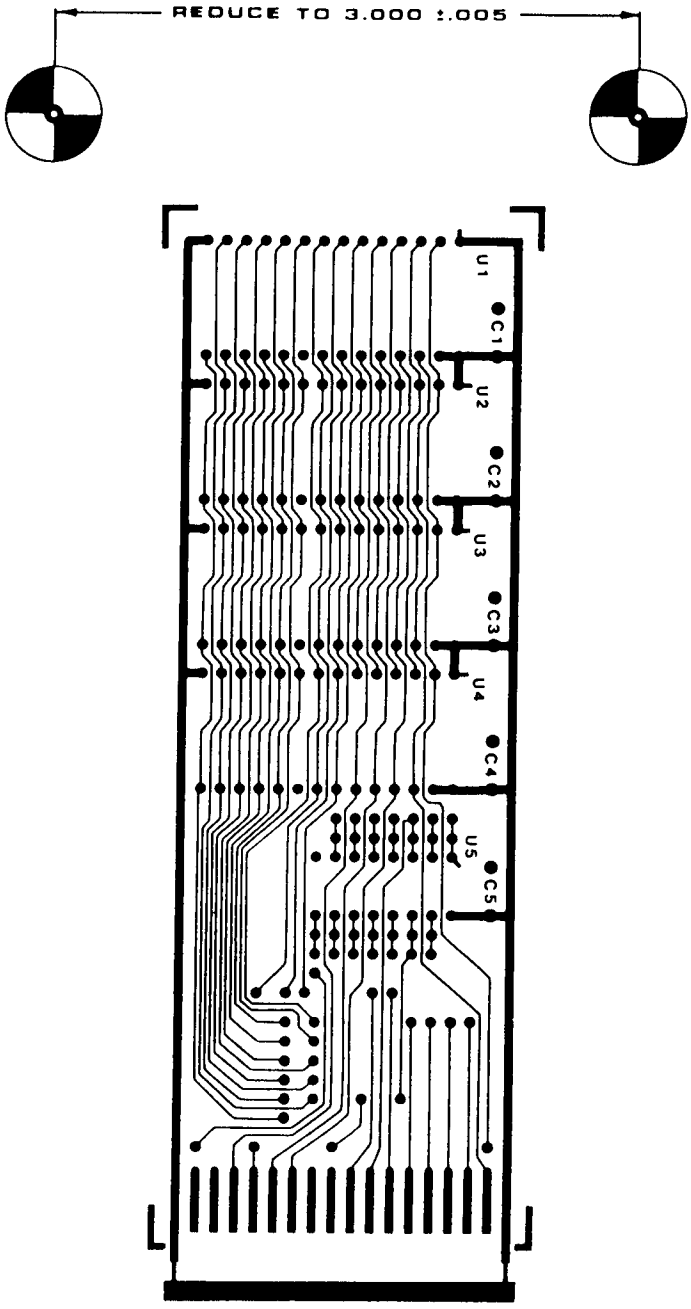


FIGURE 5.1-2

EPROM CARTRIDGE BOARD

COMPONENT SIDE ARTWORK



COMPONENT SIDE

SK2000-81

FIGURE 5.1-3

EPROM CARTRIDGE BOARD

SOLDER SIDE ARTWORK

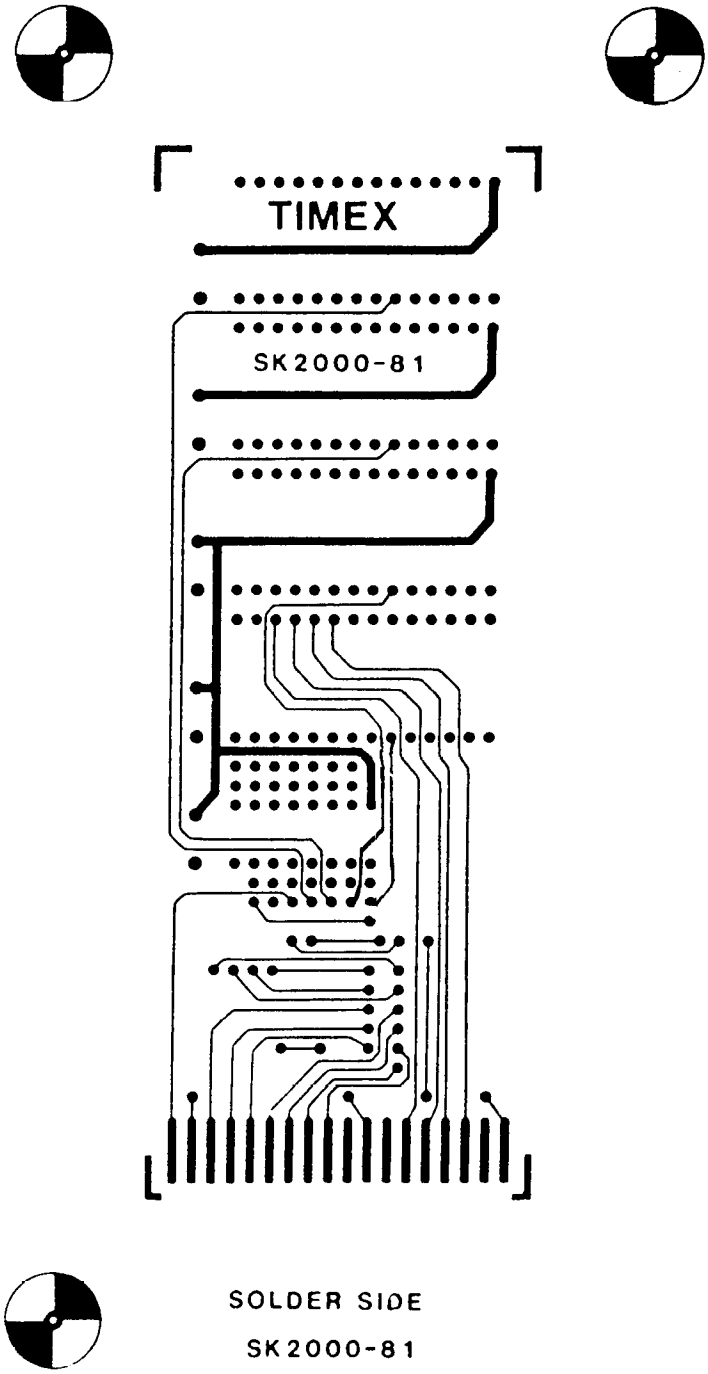
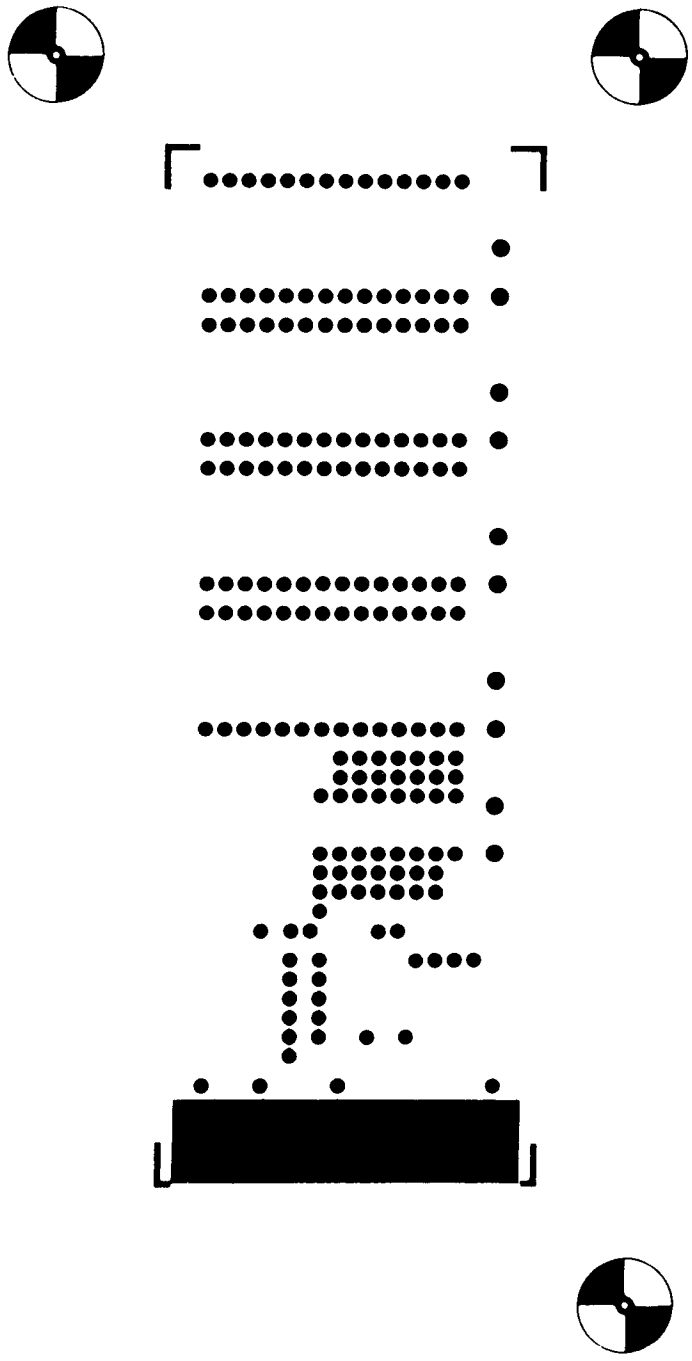


FIGURE 5.1-4

EPROM CARTRIDGE BOARD

SOLDER MASK



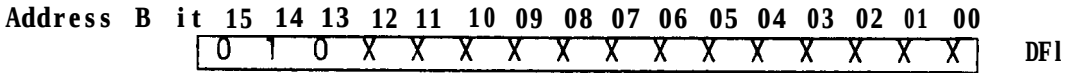
SOLDER MASK

SK2000-81

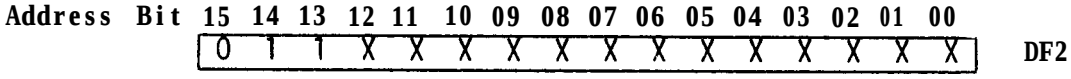
5.2 Advanced Video Modes

The following sections describe the various video modes available on the TS 2068 and the major software supoort 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 5912 (1B00H) 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:



4000 - 5AFFH (Bit 13 = 0)



6000 - 7AFFH (Bit13 = 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 VIDMOD (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:

LINE #	INDEX	TABLE LSB/MSB		
0	0	00 40	4000H =	Line 0 (Top of Screen)
1	2	20 43		Line 1
2	4	40 40		Line 2
.	.	(+20H)		
		(+20H)		
7	14(0EH)	EO 40		Line 7 (End of Upper Block)
8	16(10H)	00 48	4800H =	Line 8 (Top of Middle Block)
9	18(12H)	20 48		Line 9
.	.	(+20H)		
		(+20H)		
15	30(1EH)	EO 48		Line 15(End of Middle Block)
16	32(2OH)	00 50	5000H =	Line 16(Top of Bottom Block)
17	34(22H)	20 50		Line 17
.	.	(+20H)		
	.	(+20H)		
23	46(2EH).....	EO 50		Line 23(End of Bottom Block)

Line 17, Column 23 (llH/17H) would yield a display file address of 5020H + 17H = 5037H. If VIDMOD 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=l) 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 (OBH) which added to the starting address results in a display file address of 702BH. 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.

```
        LD      B,8         Set Scan Count
LOOP    LD      A,(DE)      Get pixel pattern
        LD      (HL),A      Write to Display File
        INC     DE          Next pixel Pattern byte
        INC     H           Next DF Position (+100H)
        DJNZ    LOOP        Continue for 8 Scan Rows
```

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&l
        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 orimary 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 VIDMOD to determine which display file is the target of the current operation. Special values for VIDMOD 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 VIDMOD (POKE 23746,128) to keep things consistent inside the dual screen support code.

5.2.2 High Resolution Graphics Hode

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 SCRMBL.

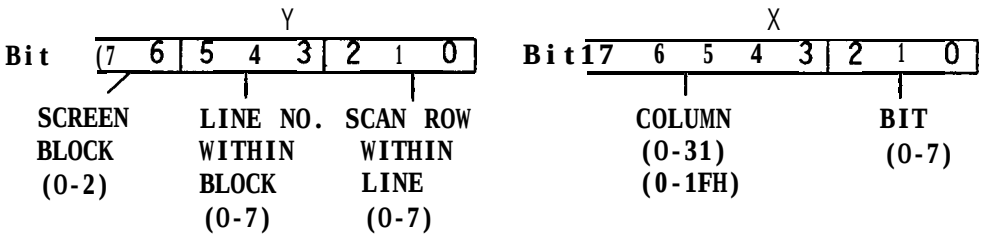
The two inputs are assumed to be as follows:

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:



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   0C0H           Get Block No. (0-2)
        RRA                  Shift Bits to Pos. 3&4
        RRA
        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 Addrs.
```

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               O=Leftmost (MSB);7=
                                Rightmost (LSB)
        LD      B,A             Use as Control Count
        INC     B               B=l-8
        LD      A,00000001B     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 64-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.

B. 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 0FFH = 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 3.7.3.1 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 0FFH.

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 YEM 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 RROS 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 1BOOH) 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 systen 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:

168 bytes (A8H)	User Defined Graphics (21 X 8)
2112 bytes (840H)	Machine Stack and OS Routines
2280 (8E8H)	

Example:		
RAMTOP =	63255	(F717H)
+ Reserved Area	2280	(08E8H)
	65535	(FFFFH)

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 he 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-z). 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-l 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 (erg. 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

	LOCATION (HEX)	OBJ. CODE (HEX)	SOURCE STATEMENT			COMMENTS
			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	FEFF		CP	OFFH-	Test if Home(255)
*	640E	2837		JR	Z,GS HOME	
	6410	A7		AND	A	Test if Dock (0)
*	6411	2827		JR	Z,GS DOCK	
	6413	.		.		
	.	.		.		(Code for Expansion Banks
	.	.		.		not applicable)
	.	.		.		
*	6430	0EFF	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,(0F4H)	Get Hor.Select Reg.
*	6444	2F		CPL		Invert to Low Active
*	6445	1802		JR	GS XT0	Exit
*	6447	DBF4	GS HOME	IN	A,0F4H	All bits set are not active in Home Bank
*	6449	4F	GS XT0	LD	C,A	Memory Select to C
	644A	Dl	GS XT1	POP	DE	Restore Regs.
	644B	F1		POP	AF	
	644C	C9		RET		Return

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

FIGURE 6.5-l

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-Z

PUT WORD CORRECTIONS

	LOCATION (HEX)	OBJ.CODE EX)	SOURCE STATEMENT		COMMENTS
			Input: Data in DE, Address in HL, Bank # in B		
	6338	F5	PUT-WORD	PUSH AF	Save Regs.
	633C	c5		PUSH BC	
	633D	CD5E64		CALL GET NUMBER	Bank # of Owner
*	6340	D5		PUSH DE	Save Data
	6341	50		LD D,B	Save Target Bank #
	6342	47		LD B,A	Bank # of Owner
	6343	CD0564		CALL GET-STATUS	Get Bank Status
	6346	C5		PUSH BC	Save It
	6347	CD4D64		CALL GET CHUNK	Get Bit Map
	634A	2F		CPL	Set High Active
	634B	42		LD B,D	Target Bank # to B
	634C	4F		LD C.A	Memory Select Byte
	634D	CD9964		CALL BANK ENABLE	Enhl.Target Mem.
*	6350	Cl		POP BC	Saved Bank Status
*	6351	D1		POP DE	Saved Data
*	6352	73		LD (HL),E	Write LSB
*	6353	23		INC HL	Increment Adrs.
*	6354	72		LD (HL),D	Write MSB
*	6355	2B		DEC HL	Restore HL
	6356	CO9964		CALL BANK ENABLE	Restore Bank St.
	6359	Cl		POP BC	Restore Regs.
	635A	F1		POP AF	
	635B	C9		RET	Return

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.

FIGURE 6.5-3

BANK_ENABLE AND RESTORE STATUS CORRECTIONS

				From BASIC	
BANK ENABLE:	**Location**	**Object Code**		**POKE Address**	**Value**
	6499H	**00**	**NOP**	**25753**	**0**
	649DH	**F3**	DI	25757	243
	651 CH	**FB**	EI	25884	251
RESTORE _STATUS:					
	654AH	**F3**	DI	25930	243
	6570H	**FB**	EI	25968	251

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 exitted. 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 26474,951.**

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 -lE-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/Z 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/0) 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

LOAD MAP

MODULE	ORIGIN	LENGTH
BLOCK	0000	0000
BASIC	0000	9227
KSCAN	0227	02D9
IO_1	0500	0502
IO_2	0A02	031B
EDIT	0D1D	0682
CHANS	139F	0142
LIST	14E1	02D4
AROS	17B5	0190
SYNTAX	1945	080A
SYNTWO	214F	04B4
GRAPHS	2603	0251
EXPRN	2854	041C
IDENT	2C70	03E9
INOUT	3059	0301
SUMS	335A	032A
CALC	3684	0437
FUNCTS	3ABB	01CE
TAPEMSG	3C89	0053
CH_SET	3D00	0300

GLOBAL	ADDRESS	MODULE
ACS	3C5E	FUNGTS
ADD	33D3	SUMS
ALNUM?	3046	IDENT
ALPHA?	304B	IDENT
ANGLE	3B9E	FUNGTS
AROS	18C6	AROS
ARRAY	37C5	GALG
AR_LN	17EA	AROS
AR_NXT	17FF	AROS
ASN	3C4E	FUNCTS
ATN	3BFD	FUNGTS
ATTBYT	0710	IO_1
BEEP	0436	KSCAN
BORDER	2436	SYNTWO
BREAK?	2009	SYNTAX
CAT	25C8	SYNTWO
CHCODE	0371	KSCAN
CHINIT	11AA	EDIT
CHK_SZ	1FBB	SYNTAX
CIRCLE	2679	GRAPHS
CLCHAN	13BE	CHANS
CLEAR	1F36	SYNTAX
CLEL	133F	EDIT
CLLHS	08A9	IO_1
CLOSE	139F	CHANS
GLPR	Cm35	IO_2
CLR_BC	1F39	SYNTAX
CLS	08EA	IO_1
CLS_B	097F	IO_1
COLITM	23A6	SYNTWO
COLOUR	23DE	SYNTWO
CONT	1EE4	SYNTAX
COS	3BC5	FUNCTS
GP-EC	16E8	LIST
CTRO	371A	CALC
DATA	1E82	SYNTAX
DEF	201D	SYNTAX
DELREC	1750	LIST
DELSYM	0B7E	IO_2
DEL-DE	174D	LIST
DEL-C:	0BFD	IO_2
DESLUG	0D0D	IO_2
DE_HL	1668	LIST
DIGIT?	30D9	INOUT
DIM	2FC0	IDENT
DIVIDE	356E	SUMS
DRAW	26DB	GRAPHS
DRAWLN	2813	GRAPHS
DRAW_L	2810	GRAPHS
DUMPPR	0A23	IO_2
DYADIC	1BDC	SYNTAX
ECHO	0Ca3	IO_2
EDIT-K	0A82	IO_2
END?	1B44	SYNTAX
ENDSTT	1AB9	SYNTAX
ENDTEM	1B4A	SYNTAX
ERASE	25D4	SYNTWO
ERR2	1B91	SYNTAX
ERR4	1FCF	SYNTAX
ERRS	07C1	IO_1
ERR6	356C	SUMS
ERRB	1F29	SYNTAX
ERRH	237E	SYNTWO
ERRO	123D	EDIT
EXCUTE	1AD8	SYNTAX
EXP	3ADF	FUNCTS
EXPRN	2854	EXPRN
FIND-L	16D6	LIST
FIND_N	2C70	IDENT
FIX_U	1F23	SYNTAX
FIX_U1	1F1E	SYNTAX
FLASHA	160D	LIST
FLOAT	3656	SUMS
FOR	1C78	SYNTAX
FORMAT	25CC	SYNTWO
FP2A	3193	INOUT
FP2BC	3160	INOUT
F_ATTR	28D7	EXPRN
F_INKY	29F2	EXPRN
F-PI	29E5	EXPRN
F_PNT	2624	GRAPHS
F_SCRN	288E	EXPRN
GETAL	17CF	AROS
GET_EL	2D54	IDENT
GET_LN	1324	EDIT
GET_XY	2660	GRAPHS
GO_SUB,	1F99	SYNTAX
GR_COL	238C	SYNTWO
HIFLSH	241D	SYNTWO
INGH	11E1	EDIT
ININT	30F9	INOUT
INIT	0D31	EDIT
INPUT	222B	SYNTWO
INSI	12B8	EDIT
INSA	0AE7	ID-2
INSERT	12BB	EDIT
INT	3ACA	FUNCTS
INTDIV	3ABB	FUNCTS

INTPT?	2389	EXPRN
IN_K	000E	IO_2
I_SEQ	226B	SYNTWO
JUMP	1EF1	SYNTAX
K_BASE	035C	KSCAN
K_CLS	08A6	IO_1
K_DUMP	0A02	IO_2
K_LIST	1545	LIST
K_LLST	1541	LIST
K_LPR	2155	SYNTWO
K_NEW	0D1D	EDIT
K_PRIN	2159	SVNTWG
K_SCAN	02B0	KSCaN
LCU2	132D	EDIT
LDDE	3130	INOUT
LDMES	3CA8	TAPEMSG
LDTVCU	061A	IO_1
LE3	0055	BASIC
LED18	0E2F	EDIT
LED4	0E8D	EDIT
LET	2EBD	I DENT
LINENG	1768	LIST
LIST	14E1	LIST
LN	3B2E	FUNCTS
LPO	15AC	LIST
LS4	1 a44	SYNTaX
LT22	1 BBC	SYNTaX
MOVE	25D0	SYNTWO
MULT	3468	SUMS
NC_HL	0077	BASIC
NEGATE	382D	CALC
NEW	0D82	EDIT
NEWDEV	24D2	SYNTWO
NEXT	1D55	SYNTAX
NEXTCH	0074	BASIC
NEXT-L	165B	LIST
NOTKB?	2380	SYNTWO
NXT-HL	2C69	EXFRN
OPCHAN	1465	CHANS
OPEN	142A	CHaNS
OPTNO	1C49	SYNTaX
OUTPUT	31A1	INOUT
PAEDCB	2E74	I DENT
PARP	03F3	KSCaN
PASSEM	25B9	SYNTWO
PAUSE	1FEB	SYNTAX
PHLAF	004F	BASIC
PLOT	2635	GRAPHS
PLOTBC	263E	GRAPHS
PLUG IN	0000	BASIC
PGPSTR	2FAF	IDENT
PRSCAN	0A4A	IO_2
PR_CUR	162D	LIST
PR_TV2	0776	IO_1
PSHSTR	2E70	IDENT
PUT	15C9	LIST
PUTDIG	11EA	EDIT
PUTMES	073F	IO_1
PUT-BC	1788	LIST
PUT_LN	1795	LIST
PUT_SR	15A1	LIST
P_LFT	053A	IO_1
P_NL	0566	IO_1
P_RT	0554	IO 1
P_SEQ	217E	SYNTWO
RAMNO	377F	CALC
RAND	1 ED4	SYNTAX
RDCH	11CF	EDIT
READ	1D97	SYNTaX
RECLEN	1720	LIST
REMGSZ	12CA	EDIT
RESET	1354	EDIT
HESTBC	1ECA	SYNTaX
RETURN	1 FD4	SYNTaX
RND	29B6	EXPRN
ROOM?	3768	CALC
ROOT	3C65	FUNCTS
RSET	2454	SYNTWO
RSTSTR	13A8	CHANS
R_ATTS	0838	IO_1
SCRL	0939	IO_1
SCRMBL	2603	GRAPHS
SEARCH	136B	EDIT
SELECT	1230	EDIT
SEL_HL	1248	EDIT
SENDCH	11ED	EDIT
9ENDTV	0500	IO_1
SEPRMT	3C89	TAPEMSG
SETCUR	0914	IO_1
SETTVC	0914	IO_1
SET_AT	05B2	IO_1
SHIFT	339C	SUMS
SIN	3BD0	FUNCTS
SKIP	1 D28	SYNTAX
SKIPIT	2569	SVNTWO
SLICER	2E10	I DENT
SMINIT	11C1	EDIT
SOUND	2128	SYNTAX
SRCHSC	1.374	EDIT
STBOOL	3926	CALC
STDE_S	314c	INOUT
STDE_U	314A	I NOUT
STKUSN	3059	INOUT
STK_O	1C51	SYNTAX
STK_A	30E6	I NOUT
STK-EC	30E9	INOUT
STK_M	3773	CALC
STOP	1C59	SYNTAX
STRITO	220F	SVNTWO
STTVCU	05F3	IO_1
SUB	33CE	SUMS
SUBLIN	16F0	LIST
SUBLN 1	16F3	LIST
SUMSLD	3379	SUMS
SVNERR	1 BED	SYNTAX
SYNTAX	1 A27	SYNTAX
TAN	3BF5	FUNCTS
TC_HL	0078	BASIC
TEM1	1B82	SYNTAX
TEM10	1BEF	SYNTaX
TEM6	1BE5	SYNTAX
TEMP38	19E0	SYNTAX
TEMP39	19E1	SYNTaX
TERM?	21E7	SYNTWO
TESTO	3904	CALC
TIMES	3489	SUMS
TOKENS	0098	BASIC.
TO-THE	3C6C	FUNCTS
TRUNC	35D3	SUMS
TVFUL?	0790	IO_1
TV_COL	23BB	SYNTWO
UPD_K	02E1	KSCAN
USRRET	3882	CALC
WRCH	0010	BASIC
XEV	310D	INOUT
X_CALC	134E	EDIT
X_T_HL	1363	EDIT

PROGRAM BLOCK -- 4000 BYTES
ENTRY: 0000

EXTENSION ROM MAP

```
LINK 1.7

LOAD MAP
MODULE      ORIGIN LENGTH

XBASIC        0000   0068
TAPE          0068   087F
INIT          08E7   04C9
CHNG_VID      0DB0   0193
PASSING       0F43   0047
BS            0F8A   001E

GLOBAL      ADDRESS   MODULE

AKEY          08AA   TAPE
BLDSCT        09F4   INIT
CALL_B        0F99   BS
CHNG_V        0E8E   CHNG_VID
CLDFIL        0E27   CHNG_VID
EXINIT        08E7   INIT
GOTO_B        0F8A   BS
LOAD          05CC   TAPE
MERGE         06E5   TAPE
OPDFIL        0DB0   CHNG_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      1000   0624    THIS MODULE IS COPIED TO RAM 6200 (space reserved 6200-683FH).

GLOBAL      ADDRESS   MODULE  Relocated to RAM F9C0-FFFFH when second Display File is used.

BANK_E        6499   DISPATCH
BS_MAX        6315   DISPATCH
BS_SP         65CE   DISPATCH
CALL_B        65D0   DISPATCH
CREATE        66E8   DISPATCH
DISPAT        6200   DISPATCH
GET_CH        644D   DISPATCH
GET_NU        645E   DISPATCH
GET_ST        6405   DISPATCH
GET_WO        6316   DISPATCH
GOTO_B        6572   DISPATCH
GOTO_E        6815   DISPATCH
INT           62AE   DISPATCH

                                          ORIGIN  LENGTH
TABLES:    FIXTBL      1D00     007C
           JMPTBL      1EDC     0124
UNUSED:                1624     06DC
                       1D7C     0160
```

```
                    420  *LIST ON
                    421  *LIST ON
                    422  *INCLUDE NEW_SYSVAR2.S
                    423  ;  HERE ARE SOME NEW SYSTEM VARIABLE DEFINITIONS
                    424  ;
                    425  ;
                    426  STKSZ           EQU     200H
                    427  SADDPT          EQU     0F5H    ;SOUND CHIP ADDR PORT
                    428  SDATPT          EQU     0F6H    ;SOUND CHIP DATA PORT
                    429  HS              EQU     40H
                    430  HS_LSN          EQU     HS
                    431  BNA             EQU     20H
                    432  HS_MSN          EQU     BNA
                    433  ABN             EQU     0A0H
                    434  HSP             EQU     ABN     ;HS' REG ADDR
                    435  STA_L           EQU     ABN
                    436  CMD             EQU     0C0H
                    437  STA_0           EQU     CMD
                    438  LOWNYB          EQU     0C0C0H          ;RESET NYBBLE STEERING LOGIC CMD
                    439  FREE_BYTES      EQU     32
                    440  PRM_OUT         EQU     8
                    441  HOR_SEL         EQU     10
                    442  BANK            EQU     11
                    443  UPD_K           EQU     02E1H
                    444  ;
                    445  ;
                    446
                    447          GLOBAL DISPATCH, INT, NMI, PUT_WORD, GET_WORD
                    448          GLOBAL WRITE_BS_REG, READ_BS_REG, GET_STATUS, GET_NUMBER
                    449          GLOBAL GET_CHUNK, BANK_ENABLE, GOTO_BANK, CALL_BANK
                    450          GLOBAL XFER_BYTES, BS_MAX_BANK, BS_SP
                    451          GLOBAL CREATE_BITMAP, MOVE_BYTES
                    452
                    453  ;  DISPATCH (SVC_CODE: PASSED ON STACK)
                    454  ;
                    455  ;    SVC_CODE IS A 16 BIT QUANTITY.  BIT 15 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  JMPTBL          EQU     1FFFH
                    462  LAST_EXT_SVC    EQU     13
                    463  LAST_RAM_SVC    EQU     24
                    464  ;
                    465  ;
                    466  ;
6200                467                  ORG     6200H
                    468
6200  DD210000      469  DISPATCH        LD      IX, 0
6204  DD39          470                  ADD     IX, SP          ;IX = SP
6206  C5            471                  PUSH    BC              ;RESERVE A WORD ON THE STACK
6207  F5            472                  PUSH    AF              ;SAVE REGS
6208  C5            473                  PUSH    BC
6209  D5            474                  PUSH    DE
620A  E5            475                  PUSH    HL
620B  DD5E02        476                  LD      E, (IX+2)
620E  DD5603        477                  LD      D, (IX+3)       ;DE = SVC_CODE
6211  AF            478                  XOR     A
6212  CB23          479                  SLA     E
6214  CB12          480                  RL      D               ;DE = 2*DE
6216  17            481                  RLA                     ;A = JUMP FLAG
6217  210D00        482                  LD      HL, LAST_EXT_SVC
621A  CB25          483                  SLA     L
621C  CB14          484                  RL      H               ;HL = 2*HL
621E  A7            485                  AND     A
621F  ED52          486                  SBC     HL, DE          ;COMPARE HL AND DE
6221  3015          487                  JR      NC, D_EXT       ;IF DE <= HL
6223  211800        488                  LD      HL, LAST_RAM_SVC
6226  CB25          489                  SLA     L
6228  CB14          490                  RL      H
622A  A7            491                  AND     A
622B  ED52          492                  SBC     HL, DE
622D  380F          493                  JR      C, D_HOME
622F  06FF          494                  LD      B, 255          ;HERE FOR RAM-BASED SERVICES
6231  CD0564        495                  CALL    GET_STATUS      ;GET STATUS OF HOME BANK
6234  06FF          496                  LD      B, 255          ;BC = HOME BANK / HORIZ-SELECT
6236  180A          497                  JR      D_SAVE
6238  06FE          498  D_EXT           LD      B, 254          ;HERE FOR EXT. ROM BASED SERVICES
623A  0EFE          499                  LD      C, 0FEH
623C  1804          500                  JR      D_SAVE
623E  06FF          501  D_HOME          LD      B, 255          ;SET BANK_ENABLE PARMS FOR HOME
6240  0E00          502                  LD      C, 0
6242  F5            503  D_SAVE          PUSH    AF
6243  C5            504                  PUSH    BC              ;SAVE JUMP FLAG AND BANK_ENABLE PARMS
6244  21FF1F        505                  LD      HL, JMPTBL      ;CALC. ADDR OF TABLE ENTRY
6247  37            506                  SCF
6248  ED52          507                  SBC     HL, DE
624A  06FE          508                  LD      B, 254
624C  CD1663        509                  CALL    GET_WORD        ;READ TABLE ENTRY
624F  EB            510                  EX      DE, HL
6250  C1            511                  POP     BC
6251  F1            512                  POP     AF              ;RESTORE JUMP FLAG, ETC.
6252  A7            513                  AND     A
6253  281F          514                  JR      Z, D_CALL
6255  DD71FE        515                  LD      (IX-2), C       ;PUT BANK# AND HOR-SEL ON STACK
6258  DD70FF        516                  LD      (IX-1), B
625B  DD6E00        517                  LD      L, (IX)         ;SAVE RET ADDR
625E  DD6601        518                  LD      H, (IX+1)
6261  DD7403        519                  LD      (IX+3), H       ;PUT RET ADDR BACK ON STACK
```

```
6264  DD7502      520                 LD      (IX+2), L
6267  DD7201      521                 LD      (IX+1), D       ;SET UP STACK FOR GOTO_BANK
626A  DD7300      522                 LD      (IX), E         ;PUT ADDR ON STACK
626D  E1          523                 POP     HL              ;RESTORE REGS
626E  D1          524                 POP     DE
626F  C1          525                 POP     BC
6270  F1          526                 POP     AF
6271  CD7265      527                 CALL    GOTO_BANK       ;HERE IF JUMP FLAG NOT SET
6274  DD6E00      528 D_CALL          LD      L, (IX)         ;SET UP STACK FOR CALL_BANK
6277  DD6601      529                 LD      H, (IX+1)       ;PUT RET_ADDR IN PROPER LOC
627A  E5          530                 PUSH    HL
627B  DD6E04      531                 LD      L, (IX+4)
627E  DD6605      532                 LD      H, (IX+5)
6281  DD75FE      533                 LD      (IX-2), L
6284  DD74FF      534                 LD      (IX-1), H
6287  DD6E06      535                 LD      L, (IX+6)       ;PUT PRM_OUT IN PRPER LOC
628A  DD6607      536                 LD      H, (IX+7)
628D  DD7500      537                 LD      (IX), L
6290  DD7401      538                 LD      (IX+1), H
6293  DD7102      539                 LD      (IX+2), C       ;PUT BANK #, HS ON STACK
6296  DD7003      540                 LD      (IX+3), B
6299  DD7304      541                 LD      (IX+4), E       ;PUT ADDR ON STACK
629C  DD7205      542                 LD      (IX+5), D
629F  E1          543                 POP     HL
62A0  DD7506      544                 LD      (IX+6), L
62A3  DD7407      545                 LD      (IX+7), H
62A6  E1          546                 POP     HL              ;RESTORE REGS
62A7  D1          547                 POP     DE
62A8  C1          548                 POP     BC
62A9  F1          549                 POP     AF
62AA  CDD065      550                 CALL    CALL_BANK       ;HERE IF JUMP FLAG NOT SET
62AD  C9          551                 RET
                  552 ;
                  553 ;
                  554                 ;RST 56: HERE TO SERVICE INTERRUPT BY READING KEYBOARD
                  555
62AE  F5          556 INT     PUSH AF
62AF  E5          557         PUSH HL
62B0  DDE5        558         PUSH    IX
62B2  210000      559         LD      HL, 0
62B5  39          560         ADD     HL, SP
62B6  D5          561         PUSH    DE
62B7  3A1563      562         LD      A, (BS_MAX_BANK)
62BA  5F          563         LD      E, A
62BB  1600        564         LD      D, 0
62BD  13          565         INC     DE
62BE  13          566         INC     DE
62BF  A7          567         AND     A
62C0  ED52        568         SBC     HL, DE
62C2  EB          569         EX      DE, HL
62C3  DD210000    570         LD      IX, 0
62C7  DD19        571         ADD     IX, DE
62C9  D1          572         POP     DE
62CA  DDF9        573         LD      SP, IX
62CC  CD1E65      574         CALL    SAVE_STATUS
62CF  C5          575         PUSH    BC
62D0  06FF        576         LD      B, 0FFH
62D2  CD0564      577         CALL    GET_STATUS
62D5  06FF        578         LD      B, 0FFH
62D7  79          579         LD      A, C
62D8  E6F8        580         AND     0F8H
62DA  4F          581         LD      C, A
62DB  CD9964      582         CALL    BANK_ENABLE
62DE  C1          583         POP     BC
62DF  2A785C      584         LD HL,(FRAMES)  ;NOW INCREMENT FRAME COUNTER
62E2  23          585         INC HL
62E3  22785C      586         LD (FRAMES),HL
62E6  7C          587         LD A,H
62E7  B5          588         OR L
62E8  2003        589         JR NZ,LIT3
62EA  FD3440      590         INC (IY-Y+FRAME2)
62ED  C5          591 LIT3:   PUSH BC
62EE  D5          592         PUSH DE
62EF  CDE102      593         CALL    UPD_K
62F2  D1          594         POP DE
62F3  C1          595         POP BC
                  596 PHLAF:                ;JUMP HERE TO POP HL, POP AF, ENABLE INTERRUPTS & RETURN
62F4  DD210000    597         LD      IX, 0
62F8  DD39        598         ADD     IX, SP
62FA  CD4A65      599         CALL    RESTORE_STATUS
62FD  DD23        600         INC     IX
62FF  DDF9        601         LD      SP, IX
6301  DDE1        602         POP     IX
6303  E1          603         POP HL
6304  F1          604         POP AF
6305  FB          605         EI
6306  C9          606         RET
                  607
                  608                   ;HERE TO SERVICE NON-MASKABLE INTERRUPT
                  609                   ;IF (NMIADD) = 0 THEN RETURNS STRAIGHT AWAY!
                  610                   ;ELSE, JUMPS TO (NMIADD) WITH HL (ON TOP), AF & RETN ADDR ON
                  611                   ;     THE STACK.
6307  F5          612 NMI     PUSH AF
6308  E5          613         PUSH HL
6309  2AB05C      614         LD HL,(NMIADD)
630C  7C          615         LD A,H
630D  B5          616         OR L
630E  2001        617         JR NZ,LNI3      ;IF NO USER-SUPPLIED SERVICE ROUTINE
6310  E9          618         JP (HL)
```

```
                        619
6311    E1              620 LNI3:   POP HL
6312    F1              621         POP AF
6313    ED45            622         RETN
                        623 ;
                        624 ;
6315                    625 BS_MAX_BANK     DEFS    1       ;THIS IS A COPY OF MAX_BAN
                        626 ;
                        627 ;  GET_WORD (ADDR: HL, BANK: B; WORD:HL)
                        628 ;
                        629 ;
6316    F5              630 GET_WORD        PUSH    AF              ;SAVE REGS
6317    C5              631                 PUSH    BC
6318    D5              632                 PUSH    DE
6319    CD5E64          633                 CALL    GET_NUMBER      ;GET BANK # OF OWNER OF ADDR
631C    F5              634                 PUSH    AF
631D    50              635                 LD      D, B
631E    47              636                 LD      B, A
631F    CD0564          637                 CALL    GET_STATUS      ;GET STATUS OF OWNER
6322    C5              638                 PUSH    BC
6323    CD4D64          639                 CALL    GET_CHUNK       ;SET HS FOR GETTING AT ADDR
6326    2F              640                 CPL                     ;PUT IN ACTIVE LOW FORMAT
6327    42              641                 LD      B, D
6328    4F              642                 LD      C, A
6329    CD9964          643                 CALL    BANK_ENABLE     ;ENABLE ADDR
632C    5E              644                 LD      E, (HL)         ;READ THE WORD
632D    23              645                 INC     HL
632E    56              646                 LD      D, (HL)
632F    2B              647                 DEC     HL
6330    EB              648                 EX      DE, HL
6331    C1              649                 POP     BC
6332    F1              650                 POP     AF
6333    47              651                 LD      B, A
6334    CD9964          652                 CALL    BANK_ENABLE     ;REENABLE OWNER OF ADDR
6337    D1              653                 POP     DE              ;RESTORE REGS
6338    C1              654                 POP     BC
6339    F1              655                 POP     AF
633A    C9              656                 RET
                        657 ;
                        658 ;
                        659 ;  PUT_WORD (WORD: DE, ADDR: HL, BANK: B)
                        660 ;
                        661 ;
633B    F5              662 PUT_WORD        PUSH    AF              ;SAVE REGS
633C    C5              663                 PUSH    BC
633D    CD5E64          664                 CALL    GET_NUMBER      ;GET BANK # OF OWNER OF ADDR
6340    F5              665                 PUSH    AF
6341    50              666                 LD      D, B
6342    47              667                 LD      B, A
6343    CD0564          668                 CALL    GET_STATUS      ;GET STATUS OF OWNER
6346    C5              669                 PUSH    BC
6347    CD4D64          670                 CALL    GET_CHUNK       ;SET HS FOR GETTING AT ADDR
634A    2F              671                 CPL                     ;PUT IN ACTIVE LOW FORMAT
634B    42              672                 LD      B, D
634C    4F              673                 LD      C, A
634D    CD9964          674                 CALL    BANK_ENABLE     ;ENABLE ADDR
6350    73              675                 LD      (HL), E         ;WRITE THE WORD
6351    23              676                 INC     HL
6352    72              677                 LD      (HL), D
6353    2B              678                 DEC     HL
6354    C1              679                 POP     BC
6355    F1              680                 POP     AF
6356    CD9964          681                 CALL    BANK_ENABLE     ;REENABLE OWNER OF ADDR
6359    C1              682                 POP     BC
635A    F1              683                 POP     AF
635B    C9              684                 RET
                        685 ;
                        686 ;
                        687 ;  WRITE_BS_REG (REG_ADDR: D, REG_DATA: E)
                        688 ;
                        689 ;
635C    F5              690 WRITE_BS_REG    PUSH    AF              ;SAVE REGISTERS
635D    C5              691                 PUSH    BC
635E    E5              692                 PUSH    HL
635F    62              693                 LD      H, D
6360    2E00            694                 LD      L, 0            ;HL = MEMORY MAPPED ADDR
6362    3A00C0          695                 LD      A, (LOWNYB)     ;SAVE (0E000H)
6365    F5              696                 PUSH    AF
6366    7E              697                 LD      A, (HL)         ;SAVE (HL)
6367    F5              698                 PUSH    AF
6368    3E07            699                 LD      A, 7
636A    D3F5            700                 OUT     (SADDPT), A     ;SAVE VALUES OF SOUND REGS 7 AND E
636C    DBF6            701                 IN      A, (SDATPT)
636E    47              702                 LD      B, A
636F    3E0E            703                 LD      A, 0EH
6371    D3F5            704                 OUT     (SADDPT), A
6373    DBF6            705                 IN      A, (SDATPT)
6375    4F              706                 LD      C, A
6376    3E07            707                 LD      A, 7            ;SET IOA CHANNEL TO OUTPUT
6378    D3F5            708                 OUT     (SADDPT), A
637A    3E40            709                 LD      A, 40H
637C    D3F6            710                 OUT     (SDATPT), A
637E    3E0E            711                 LD      A, 0EH
6380    D3F5            712                 OUT     (SADDPT), A
6382    AF              713                 XOR     A
6383    D3F6            714                 OUT     (SDATPT), A
6385    3E02            715                 LD      A, 2
6387    3200C0          716                 LD      (LOWNYB), A     ;RESET NYBBLE STEERING LOGIC
638A    7B              717                 LD      A, E
638B    77              718                 LD      (HL), A         ;WRITE LSN OF DATA
638C    CB2F            719                 SRA     A
```

```
63BE   CB2F      720                SRA     A
63C0   CB2F      721                SRA     A
63C2   CB2F      722                SRA     A
63C4   77        723                LD      (HL), A          ;WRITE MSN OF DATA
63C5   3E07      724                LD      A, 7             ;RESTORE SOUND REGS
63C7   D3F5      725                OUT     (SADDPT), A
63C9   78        726                LD      A, B
63CA   D3F6      727                OUT     (SDATPT), A
63CC   3E0E      728                LD      A, 0EH
63CE   D3F5      729                OUT     (SADDPT), A
63D0   79        730                LD      A, C
63D1   D3F6      731                OUT     (SDATPT), A
63D3   F1        732                POP     AF
63D4   77        733                LD      (HL), A          ;RESTORE (HL)
63D5   F1        734                POP     AF
63D6   3200C0    735                LD      (LOWNYB), A      ;RESTORE (0E000H)
63D9   E1        736                POP     HL               ;RESTORE REGISTERS
63DA   C1        737                POP     BC
63DB   F1        738                POP     AF
63DC   C9        739                RET
                 740  ;
                 741  ;
                 742  ;   READ_BS_REG (LEN_ADDR: D; MEM_ADDR: E; BYTE_DATA: C)
                 743  ;
                 744  ;
63DD   F5        745  READ_BS_REG    PUSH    AF               ;SAVE REGISTERS
63DE   C5        746                PUSH    BC
63DF   E5        747                PUSH    HL
63E0   62        748                LD      H, D
63E1   2E00      749                LD      L, 0             ;HL = MEMORY MAPPED ADDR
63E3   3A00C0    750                LD      A, (LOWNYB)      ;SAVE (0E000H)
63E6   F5        751                PUSH    AF
63E7   7E        752                LD      A, (HL)          ;SAVE (HL)
63E8   F5        753                PUSH    AF
63E9   3E07      754                LD      A, 7
63EB   D3F5      755                OUT     (SADDPT), A      ;SAVE VALUES OF SOUND REGS 7 AND E
63ED   DBF6      756                 N      A, (SDATPT)
63EF   47        757                LD      B, A
63F0   3E0E      758                LD      A, 0EH
63F2   D3F5      759                OUT     (SADDPT), A
63F4   DBF6      760                IN      A, (SDATPT)
63F6   4F        761                LD      C, A
63F7   C5        762                PUSH    BC
63F8   3E07      763                LD      A, 7             ;SET IOA CHANNEL TO OUTPUT
63FA   D3F5      764                OUT     (SADDPT), A
63FC   3E40      765                LD      A, 40H
63FE   D3F6      766                OUT     (SDATPT), A
6400   3E0E      767                LD      A, 0EH
6402   D3F5      768                OUT     (SADDPT), A
6404   AF        769                XOR     A
6405   D3F6      770                OUT     (SDATPT), A
6407   3E02      771                LD      A, 2
6409   3200C0    772                LD      (LOWNYB), A      ;RESET NYBBLE STEERING LOGIC
640C   7E        773                LD      A, (HL)          ;READ LSN OF DATA
640D   E60F      774                AND     0FH
640F   4F        775                LD      C, A
6410   63        776                LD      H, E
6411   7E        777                LD      A, (HL)          ;READ MSN OF DATA
6412   CB27      778                SLA     A
6414   CB27      779                SLA     A
6416   CB27      780                SLA     A
6418   CB27      781                SLA     A
641A   B1        782                OR      C
641B   5F        783                LD      E, A             ;RETURN BYTE DATA IN E
641C   C1        784                POP     BC
641D   3E07      785                LD      A, 7             ;RESTORE SOUND REGS
641F   D3F5      786                OUT     (SADDPT), A
6421   78        787                LD      A, B
6422   D3F6      788                OUT     (SDATPT), A
6424   3E0E      789                LD      A, 0EH
6426   D3F5      790                OUT     (SADDPT), A
6428   79        791                LD      A, C
6429   D3F6      792                OUT     (SDATPT), A
642B   F1        793                POP     AF
642C   77        794                LD      (HL), A          ;RESTORE (HL)
642D   F1        795                POP     AF
642E   3200C0    796                LD      (LOWNYB), A      ;RESTORE (0E000H)
6431   E1        797                POP     HL               ;RESTORE REGISTERS
6432   C1        798                POP     BC
6433   F1        799                POP     AF
6434   C9        800                RET
                 801  ;
                 802  ;
                 803  ;   GET_BANK_STATUS (BANK: B; STATUS : B, HORIZONTAL_SELECT: C)
                 804  ;
                 805  ;
6435   F5        806  GET_STATUS      PUSH    AF               ;SAVE SAVE REGS
6436   D5        807                PUSH    DE
6437   78        808                LD      A, B
6438   FEFE      809                CP      0FEH
643A   282E      810                JR      Z, GS_EXT        ;IF BANK = 254
643C   FEFF      811                CP      0FFH
643E   281D      812                JR      Z, GS_HOME       ;IF BANK = 255
6440   A7        813                AND     A
6441   281F      814                JR      Z, GS_DOCK       ;IF BANK = 0
6443   1680      815                LD      D, BNA           ;HERE IF EXP. BANK
6445   58        816                LD      E, B
6446   CD5C63    817                CALL    WRITE_BS_REG
6449   1640      818                LD      D, HS_LSN
644B   1E80      819                LD      E, HS_MSN
```

```
641D  CDAD63      820                   CALL    READ_BS_REG     ;READ HS
6420  7B          821                   LD      A, E
6421  2F          822                   CPL
6422  4F          823                   LD      C, A
6423  16A0        824                   LD      D, STA_L
6425  1EC0        825                   LD      E, STA_0
6427  CDAD63      826                   CALL    READ_BS_REG     ;READ STATUS NYBBLES
642A  43          827                   LD      B, E
642B  181D        828                   JR      GS_EXIT
642D  010000      829   GS_HOME         LD      BC, 0           ;RETURN 0 FOR HOME BANK STATUS
6430  1818        830                   JR      GS_EXIT
6432  DBF4        831   GS_DOCK         IN      A, (DKHSPT)     ;RETURN DOCK BANK STATUS
6434  2F          832                   CPL
6435  47          833                   LD      B, A
6436  0E00        834                   LD      C, 0
6438  1810        835                   JR      GS_EXIT
643A  DBFF        836   GS_EXT          IN      A, (HREXPT)
643C  E680        837                   AND     80H             ;CLEAR ALL BITS EXCEPT BIT 7
643E  2F          838                   CPL
643F  07          839                   RLCA                    ;PUT ACTIVE LOW BIT IN BIT ZERO
6440  47          840                   LD      B, A
6441  DBF4        841                   IN      A, (DKHSPT)
6443  2F          842                   CPL
6444  E601        843                   AND     1
6446  B0          844                   OR      B
6447  47          845                   LD      B, A
6448  0E00        846                   LD      C, 0
644A  D1          847   GS_EXIT         POP     DE              ;RESTORE D, C
644B  F1          848                   POP     AF
644C  C9          849                   RET
                  850   ;
                  851   ;
                  852   ;  GET_CHUNK (ADDR: HL; MASK: A)
                  853   ;
                  854   ;
644D  C5          855   GET_CHUNK       PUSH    BC              ;SAVE B
644E  7C          856                   LD      A, H            ;CHUNK NUMBER = HIGH 3 BITS OF
644F  0605        857                   LD      B, 5            ; H SO SHIFT H RIGHT 5 BITS
6451  CB3F        858   GC_SHIFT        SRL     A
6453  10FC        859                   DJNZ    GC_SHIFT
6455  3C          860                   INC     A               ;CREATE MASK BY ROLLING A 1
6456  47          861                   LD      B, A            ; LEFT CHUNK NUMBER+1 TIMES.
6457  AF          862                   XOR     A               ; THE 1 COMES FROM THE CARRY
6458  37          863                   SCF                     ; FLAG
6459  17          864   GC_ROLL         RLA
645A  10FD        865                   DJNZ    GC_ROLL
645C  C1          866                   POP     BC              ;RESTORE B
645D  C9          867                   RET
                  868   ;
                  869   ;
                  870   ;  GET_BANK_NUMBER (ADDR: HL; BANK_NUMBER: A)
                  871   ;
                  872   ;
645E  C5          873   GET_NUMBER      PUSH    BC              ;SAVE REGS
645F  D5          874                   PUSH    DE
6460  CD4D64      875                   CALL    GET_CHUNK
6463  4F          876                   LD      C, A
6464  3A1563      877                   LD      A, (BS_MAX_BANK);GET LARGEST BANK NUMBER
6467  A7          878                   AND     A
6468  280A        879                   JR      Z, ON_RD_DOCK   ;IF NO EXP. BANKS
646A  47          880                   LD      B, A
646B  58          881   ON_CHECK        LD      E, B            ;SEARCH ALL EXP. BANKS
646C  CD0564      882                   CALL    GET_STATUS
646F  A1          883                   AND     C
6470  2823        884                   JR      Z, ON_EXP       ;FOUND THE CHUNK, SO EXIT LOOP
6472  10F7        885                   DJNZ    ON_CHECK
6474  DBF4        886   ON_RD_DOCK      IN      A, (DKHSPT)     ;NOT IN EXP. BANKS, SO CHECK DOCK
6476  2F          887                   CPL
6477  A1          888                   AND     C
6478  2818        889                   JR      Z, ON_DOCK
647A  0D          890                   DEC     C               ;IF CHUNK > 1, THEN CAN'T BE IN EXT.
647B  2011        891                   JR      NZ, ON_HOME
647D  DBFF        892                   IN      A, (HREXPT)     ;CHECK IF IN EXT. BANK
647F  E680        893                   AND     80H
6481  57          894                   LD      D, A
6482  DBF4        895                   IN      A, (DKHSPT)
6484  E601        896                   AND     1
6486  0F          897                   RRCA
6487  A2          898                   AND     D
6488  2804        899                   JR      Z, ON_HOME      ;NOT IN EXT. BANK
648A  3EFE        900                   LD      A, 0FEH         ;IN EXT. BANK, SO RETURN 254
648C  1808        901                   JR      ON_EXIT
648E  3EFF        902   ON_HOME LD      A, 0FFH         ;IN HOME BANK, SO RETURN 255
6490  1804        903                   JR      ON_EXIT
6492  AF          904   ON_DOCK         XOR     A               ;FOUND CHUNK IN DOCK, SO RETURN 0
6493  1801        905                   JR      ON_EXIT
6495  78          906   ON_EXP          LD      A, B            ;RETURN EXP. BANK NUMBER
6496  D1          907   ON_EXIT         POP     DE              ;RESTORE REGS
6497  C1          908                   POP     BC
6498  C9          909                   RET
                  910   ;
                  911   ;
                  912   ;  BANK_ENABLE (BANK: B, HORIZONTAL_SELECT: C)
                  913   ;
                  914   ;
6499  F5          915   BANK_ENABLE     PUSH    AF              ;SAVE REGISTERS
649A  C5          916                   PUSH    BC
649B  D5          917                   PUSH    DE
649C  E5          918                   PUSH    HL
649D  60          919                   LD      H, B
```

```
649E  3A1563      920                   LD      A, (BS_MAX_BANK)!GET LARGEST BANK NUMBER
64A1  A7          921                   AND     A
64A2  2811        922                   JR      Z, BE_SKIP      !IF NO EXP. BANKS
64A4  1680        923                   LD      D, BNA
64A6  1E00        924                   LD      E, 0
64A8  CD5C63      925                   CALL    WRITE_BS_REG
64AB  16A0        926                   LD      D, HSP
64AD  F5          927                   PUSH    AF
64AE  79          928                   LD      A, C
64AF  2F          929                   CPL
64B0  5F          930                   LD      E, A
64B1  F1          931                   POP     AF
64B2  CD5C63      932                   CALL    WRITE_BS_REG    !TURN OFF APPROPRIATE BITS OF
                  933                                           ! ALL EXP. BANKS
64B5  78          934   BE_SKIP         LD      A, B
64B6  A7          935                   AND     A
64B7  2011        936                   JR      NZ, BE_NTDOCK
64B9  79          937                   LD      A,C
64BA  FEFF        938                   CP      OFFH
64BC  2806        939                   JR      Z, BE_EXT_OK
64BE  DBFF        940                   IN      A, (HREXPT)     !HERE FOR DOCK
64C0  CBBF        941                   RES     7, A
64C2  D3FF        942                   OUT     (HREXPT), A
64C4  79          943   BE_EXT_OK       LD      A, C
64C5  2F          944                   CPL
64C6  D3F4        945                   OUT     (DKHSPT), A     !ENABLE DOCK
64C8  184F        946                   JR      BE_EXIT
64CA  79          947   BE_NTDOCK       LD      A, B            !CHECK IF EXT.
64CB  FEFE        948                   CP      OFEH
64CD  201D        949                   JR      NZ, BE_NTEXT
64CF  DBFF        950                   IN      A, (HREXPT)     !HERE FOR EXT.
64D1  17          951                   RLA
64D2  CB19        952                   RR      C
64D4  3F          953                   CCF
64D5  1F          954                   RRA
64D6  D3FF        955                   OUT     (HREXPT), A
64D8  CB7F        956                   BIT     7, A
64DA  2003        957                   JR      NZ, BE_SET
64DC  DBF4        958                   IN      A, (DKHSPT)
64DE  CB87        959                   RES     0, A
64E0  D3F4        960                   OUT     (DKHSPT), A
64E2  1835        961                   JR      BE_EXIT
64E4  DBF4        962   BE_SET          IN      A, (DKHSPT)
64E6  CBC7        963                   SET     0, A
64E8  D3F4        964                   OUT     (DKHSPT), A
64EA  182D        965                   JR      BE_EXIT
64EC  DBF4        966   BE_NTEXT        IN      A, (DKHSPT)     !DISABLE DOCK
64EE  2F          967                   CPL
64EF  5F          968                   LD      E, A
64F0  79          969                   LD      A, C
64F1  2F          970                   CPL
64F2  B3          971                   OR      E
64F3  2F          972                   CPL
64F4  D3F4        973                   OUT     (DKHSPT), A
64F6  CB41        974                   BIT     0, C
64F8  2006        975                   JR      NZ, BE_CHK_HOME
64FA  DBFF        976                   IN      A, (HREXPT)     !DISABLE EXT.
64FC  CBBF        977                   RES     7, A
64FE  D3FF        978                   OUT     (HREXPT), A
6500  DBF4        979                   IN      A, (DKHSPT)
6502  CB87        980                   RES     0, A
6504  D3F4        981                   OUT     (DKHSPT), A
6506  78          982   BE_CHK_HOME     LD      A, B            !CHECK IF HOME
6507  FEFF        983                   CP      OFFH
6509  280E        984                   JR      Z, BE_EXIT      !IS HOME, SO DONE
650B  1680        985                   LD      D, BNA          !WRITE NEW EXP. BANK STATUS
650D  58          986                   LD      E, B
650E  CD5C63      987                   CALL    WRITE_BS_REG
6511  1640        988                   LD      D, HS
6513  79          989                   LD      A, C
6514  2F          990                   CPL
6515  5F          991                   LD      E, A
6516  CD5C63      992                   CALL    WRITE_BS_REG
6519  E1          993   BE_EXIT         POP     HL              !RESTORE REGISTERS
651A  D1          994                   POP     DE
651B  C1          995                   POP     BC
651C  F1          996                   POP     AF
651D  C9          997                   RET
                  998   !
                  999   !
                 1000   !  SAVE_BANK_STATUSES (STATUS_ADDR: IX)
                 1001   !
                 1002   !          PUSHES THE STATUS OF ALL BANKS ON THE STACK
                 1003   !
                 1004   !
651E  F5         1005   SAVE_STATUS     PUSH    AF              !SAVE REGS
651F  C5         1006                   PUSH    BC
6520  D5         1007                   PUSH    DE
6521  DBFF       1008                   IN      A, (HREXPT)     !SAVE EXT. BANK STATUS
6523  00         1009                   NOP                     !LEAVE BITS 0-6 ALONE! NOPS PUT IN
6524  00         1010                   NOP                     ! TO KEEP ADDRS THE SAME
6525  DD7700     1011                   LD      (IX), A
6528  DD23       1012                   INC     IX
652A  DBF4       1013                   IN      A, (DKHSPT)     !GET DOCK BANK STATUS
652C  DD7700     1014                   LD      (IX), A
652F  DD23       1015                   INC     IX
6531  3A1563     1016                   LD      A, (BS_MAX_BANK)!GET NUMBER OF BANKS
6534  A7         1017                   AND     A
6535  280D       1018                   JR      Z, SS_EXIT
6537  47         1019                   LD      B, A            !SET UP COUNTER
6538  58         1020   SS_LOOP         LD      E, B            !BANK NUMBER INTO E
6539  CD0564     1021                   CALL    GET_STATUS      !GET BANK STATUS OF BANK #B
```

```
653C  DD7100    1022                    LD      (IX), C
653F  DD23      1023                    INC     IX
6541  43        1024                    LD      B, E
6542  10F4      1025                    DJNZ    SS_LOOP         ;DO FOR ALL
6544  DD2B      1026  SS_EXIT           DEC     IX
6546  D1        1027                    POP     DE              ;RESTORE REGS
6547  C1        1028                    POP     BC
6548  F1        1029                    POP     AF
6549  C9        1030                    RET
                1031  ;
                1032  ;
                1033  ;  RESTORE_BANK_STATUSES (STATUS_ADDR: IX)
                1034  ;
                1035  ;         RESTORES BANK STATUS TO ALL BANKS
                1036  ;
                1037  ;
654A  F5        1038  RESTORE_STATUS   PUSH    AF              ;SAVE REGS
654B  C5        1039                    PUSH    BC
654C  D5        1040                    PUSH    DE
654D  DD7E00    1041                    LD      A, (IX)         ;GET EXT. ROM STATUS
6550  D3FF      1042                    OUT     (HREXPT), A
6552  DD23      1043                    INC     IX
6554  DD7E00    1044                    LD      A, (IX)         ;GET DOCK BANK STATUS
6557  D3F4      1045                    OUT     (DKHSPT), A
6559  DD23      1046                    INC     IX
655B  3A1563    1047                    LD      A, (BS_MAX_BANK);GET NUMBER OF BANKS
655E  A7        1048                    AND     A
655F  280B      1049                    JR      Z, RS_EXIT
6561  47        1050                    LD      B, A            ;SET UP COUNTER
6562  DD4E00    1051  RS_LOOP           LD      C, (IX)
6565  CD9964    1052                    CALL    BANK_ENABLE     ;WRITE BANK STATUS OF BANK #B
6568  DD23      1053                    INC     IX
656A  10F6      1054                    DJNZ    RS_LOOP         ;DO FOR ALL
656C  DD2B      1055  RS_EXIT           DEC     IX
656E  D1        1056                    POP     DE              ;RESTORE REGS
656F  C1        1057                    POP     BC
6570  F1        1058                    POP     AF
6571  C9        1059                    RET
                1060  ;
                1061  ;
                1062  ;  GOTO_BANK (BANK, HORIZONTAL_SELECT, ADDR) PASSED ON STACK)
                1063  ;
                1064  ;
                1065  ;         SETS UP THE DESTINATION BANK AND JUMPS WITHOUT RETURN TO ADDRESS
                1066  ;       IN BANK.
                1067  ;
                1068  ;
6572  DD210000  1069  GOTO_BANK        LD      IX, 0           ;SET IX TO SP
6576  DD39      1070                    ADD     IX, SP
6578  DD7100    1071                    LD      (IX), C         ;SAVE BC AND TRASH RET ADDR
657B  DD7001    1072                    LD      (IX+1), B
657E  DD4E02    1073                    LD      C, (IX+2)       ;GET PARMS FOR BANK_ENABLE
6581  DD4603    1074                    LD      B, (IX+3)
6584  CD9964    1075                    CALL    BANK_ENABLE
6587  C1        1076                    POP     BC              ;RESTORE BC
6588  DDE1      1077                    POP     IX              ;TRASH PARMS TO GOTO_BANK
658A  DDE1      1078                    POP     IX              ;GET ADDR
658C  DDE9      1079  JMPIX             JP      (IX)
                1080  ;
                1081  ;
                1082  ;  CALL_BANK (ADDR, BANK, HORIZONTAL_SELECT, PRM_OUT, PRM_IN)
                1083  ;     ALL INPUT PARAMETERS ARE PUSHED ON THE STACK
                1084  ;
                1085  ;  CLOBBERS IX
                1086  ;
                1087  ;
                1088  ;         SETS UP THE BANK AND MAKES A JUMP WITH RETURN ADDRESS TO ADDRESS
                1089  ;       IN BANK.
                1090  ;
                1091  ;
658E            1092  BS_STACK          DEFS    64
65CE            1093  BS_SP             DEFS    2
                1094  ;
                1095  ;
65D0  E3        1096  CALL_BANK         EX      (SP), HL        ;GET RET ADDR
65D1  DD2ACE65  1097                    LD      IX, (BS_SP)
65D5  DD2B      1098                    DEC     IX
65D7  DD7400    1099                    LD      (IX), H
65DA  DD2B      1100                    DEC     IX
65DC  DD7500    1101                    LD      (IX), L         ;PUSH HL ON BS_STACK
65DF  E1        1102                    POP     HL
65E0  E3        1103                    EX      (SP), HL        ;GET PRM_IN
65E1  DD2B      1104                    DEC     IX
65E3  DD7400    1105                    LD      (IX), H
65E6  DD2B      1106                    DEC     IX
65E8  DD7500    1107                    LD      (IX), L         ;PUSH PRM_IN ON BS_STACK
65EB  DD22CE65  1108                    LD      (BS_SP), IX     ;UPDATE BS_SP
65EF  D5        1109                    PUSH    DE              ;SAVE REGS
65F0  C5        1110                    PUSH    BC
65F1  F5        1111                    PUSH    AF
65F2  210000    1112                    LD      HL, 0
65F5  39        1113                    ADD     HL, SP          ;HL = SP
65F6  54        1114                    LD      D, H
65F7  5D        1115                    LD      E, L
65F8  3A1563    1116                    LD      A, (BS_MAX_BANK)
65FB  4F        1117                    LD      C, A
65FC  0600      1118                    LD      B, 0
65FE  03        1119                    INC     BC
65FF  03        1120                    INC     BC              ;BC = MAX_BANK + 2
6600  A7        1121                    AND     A
```

```
6601  ED42        1122                    SBC     HL, BC
6603  F9          1123                    LD      SP, HL
6604  DD210000    1124                    LD      IX, 0
6608  DD19        1125                    ADD     IX, DE
660A  EB          1126                    EX      DE, HL          ;DE, HL NOW CONTAIN DEST, SRC
                  1127                                            ; POINTERS FOR A BLOCK MOVE
660B  DD4E08      1128                    LD      C, (IX+PRM_OUT)
660E  DD4609      1129                    LD      B, (IX+PRM_OUT)
6611  3E0E        1130                    LD      A, 14
6613  81          1131                    ADD     A, C
6614  4F          1132                    LD      C, A
6615  3001        1133                    JR      NC, CB_NC1      ;BC = PRM_OUT + 14
6617  04          1134                    INC     B
6618  EDB0        1135  CB_NC1            LDIR                    ;MAKE ROOM FOR BANK STATUS
661A  D5          1136                    PUSH    DE
661B  DDE1        1137                    POP     IX              ;IX = DE
661D  CD1E65      1138                    CALL    SAVE_STATUS
6620  DD210000    1139                    LD      IX, 0
6624  DD39        1140                    ADD     IX, SP
6626  DD4E0A      1141                    LD      C, (IX+HOR_SEL) ;GET PARMS FOR BANK_ENABLE
6629  DD4608      1142                    LD      B, (IX+BANK)
662C  CD9964      1143                    CALL    BANK_ENABLE     ;ENABLE DESTINATION BANK
662F  F1          1144                    POP     AF              ;RESTORE REGS
6630  C1          1145                    POP     BC
6631  D1          1146                    POP     DE
6632  E1          1147                    POP     HL
6633  DDE1        1148                    POP     IX              ;TRASH PARMS TO CALL_BANK AND GET ADDR
6635  DDE1        1149                    POP     IX
6637  DDE1        1150                    POP     IX
6639  CD8C65      1151                    CALL    JMPIX           ;CALL ADDRESS IN IX
663C  F5          1152                    PUSH    AF              ;SAVE REGS
663D  C5          1153                    PUSH    BC
663E  D5          1154                    PUSH    DE
663F  E5          1155                    PUSH    HL
6640  DD2ACE65    1156                    LD      IX, (BS_SP)
6644  DD4E00      1157                    LD      C, (IX)
6647  DD23        1158                    INC     IX
6649  DD4600      1159                    LD      B, (IX)
664C  DD23        1160                    INC     IX              ;POP PRM_IN OFF BS_STACK
664E  DD22CE65    1161                    LD      (BS_SP), IX     ;UPDATE BS_SP
6652  DD210000    1162                    LD      IX, 0
6656  DD39        1163                    ADD     IX, SP
6658  3E08        1164                    LD      A, 8
665A  81          1165                    ADD     A, C
665B  4F          1166                    LD      C, A
665C  3001        1167                    JR      NC, CB_NC2      ;BC = PRM_IN + 8
665E  04          1168                    INC     B
665F  DD09        1169  CB_NC2            ADD     IX, BC          ;IX = SP + PRM_IN + 8
6661  DDE5        1170                    PUSH    IX
6663  E1          1171                    POP     HL              ;HL = IX
6664  2B          1172                    DEC     HL              ;HL = SRC POINTER FOR BLOCK MOVE
6665  CD4A65      1173                    CALL    RESTORE_STATUS  ;RESTORE STATUS OF ALL BANKS
6668  DDE5        1174                    PUSH    IX
666A  D1          1175                    POP     DE              ;DE = DEST POINTER FOR BLOCK MOVE
666B  EDB8        1176                    LDDR                    ;DEALLOCATE SPACE FOR BANK STATUS
666D  EB          1177                    EX      DE, HL
666E  23          1178                    INC     HL
666F  F9          1179                    LD      SP, HL          ;RESTORE SP
6670  DD2ACE65    1180                    LD      IX, (BS_SP)
6674  DD4E00      1181                    LD      C, (IX)
6677  DD23        1182                    INC     IX
6679  DD4600      1183                    LD      B, (IX)
667C  DD23        1184                    INC     IX              ;POP RET ADDR OFF BS_STACK
667E  DD22CE65    1185                    LD      (BS_SP), IX     ;UPDATE BS_SP
6682  C5          1186                    PUSH    BC
6683  DDE1        1187                    POP     IX
6685  E1          1188                    POP     HL              ;RESTORE REGS
6686  D1          1189                    POP     DE
6687  C1          1190                    POP     BC
6688  F1          1191                    POP     AF
6689  DDE5        1192                    PUSH    IX              ;PUT RET ADDR ON STACK
668B  C9          1193                    RET
                  1194  ;
                  1195  ;
                  1196  ;  HERE ARE SOME EQUATES WHICH ARE USED BY XFER_BYTES AND THE ROUTINES IT
                  1197  ;  CALLS.
                  1198  ;
                  1199  DIRECTION       EQU     0
                  1200  BUF_PTR         EQU     0
                  1201  LENGTH          EQU     2
                  1202  DEST_ADDR       EQU     4
                  1203  SRC_ADDR        EQU     6
                  1204  DEST_BANK       EQU     8
                  1205  SRC_BANK        EQU     9
                  1206  ;
                  1207  ;
                  1208  ;  MOVE_BYTES (BYTES_TO_MOVE: DE, DIRECTION : A)
                  1209  ;
                  1210  ;
668C  E5          1211  MOVE_BYTES:     PUSH    HL              ;SAVE REGISTERS
668D  D5          1212                  PUSH    DE
668E  C5          1213                  PUSH    BC
668F  48          1214                  LD      C, B
6690  DD4609      1215                  LD      B, (IX+SRC_BANK)
6693  CD9964      1216                  CALL    BANK_ENABLE     ;SELECT SRC BANK
6696  42          1217                  LD      B, D            ;MOVE FROM SRC TO STACK
6697  4B          1218                  LD      C, E
6698  DD5E00      1219                  LD      E, (IX+BUF_PTR)
669B  DD5601      1220                  LD      D, (IX+BUF_PTR+1)
669E  DD6E06      1221                  LD      L, (IX+SRC_ADDR)
66A1  DD6607      1222                  LD      H, (IX+SRC_ADDR+1)
```

```
66A4  07        1223                    RLCA
66A5  0F        1224                    RRCA
66A6  3805      1225                    JR      C, MB_RV1       ;IF A < 0
66A8  EDB0      1226                    LDIR
66AA  09        1227                    ADD     HL, BC          ;INCREMENT POINTER
66AB  1805      1228                    JR      MB_UP1
66AD  EDB8      1229  MB_RV1           LDDR
66AF  A7        1230                    AND     A
66B0  ED42      1231                    SBC     HL, BC          ;DECREMENT POINTER
66B2  DD7506    1232  MB_UP1            LD      (IX+SRC_ADDR), L  ;STORE NEW POINTER VALUE
66B5  DD7407    1233                    LD      (IX+SRC_ADDR+1), H
66B8  C1        1234                    POP     BC
66B9  E1        1235                    POP     HL
66BA  E5        1236                    PUSH    HL
66BB  C5        1237                    PUSH    BC
66BC  DD4608    1238                    LD      B, (IX+DEST_BANK)
66BF  CD9964    1239                    CALL    BANK_ENABLE     ;SELECT DEST BANK
66C2  44        1240                    LD      B, H            ;MOVE FROM STACK TO DEST
66C3  4D        1241                    LD      C, L
66C4  DD5E04    1242                    LD      E, (IX+DEST_ADDR)
66C7  DD5605    1243                    LD      D, (IX+DEST_ADDR+1)
66CA  DD6E00    1244                    LD      L, (IX+BUF_PTR)
66CD  DD6601    1245                    LD      H, (IX+BUF_PTR+1)
66D0  07        1246                    RLCA
66D1  0F        1247                    RRCA
66D2  3805      1248                    JR      C, MB_RV2       ;IF A < 0
66D4  EDB0      1249                    LDIR
66D6  09        1250                    ADD     HL, BC          ;INCREMENT POINTER
66D7  1805      1251                    JR      MB_UP2
66D9  EDB8      1252  MB_RV2           LDDR
66DB  A7        1253                    AND     A
66DC  ED42      1254                    SBC     HL, BC          ;DECREMENT POINTER
66DE  DD7504    1255  MB_UP2            LD      (IX+DEST_ADDR), L  ;STORE NEW POINTER VALUE
66E1  DD7405    1256                    LD      (IX+DEST_ADDR+1), H
66E4  C1        1257                    POP     BC              ;RESTORE REGISTERS
66E5  D1        1258                    POP     DE
66E6  E1        1259                    POP     HL
66E7  C9        1260                    RET
                1261  ;
                1262  ;
                1263  ;  CREATE_BITMAP (ADDR: HL; BITMAP: A)
                1264  ;
                1265  ;
66E8  54        1266  CREATE_BITMAP    LD      D, H            ;SAVE START ADDR
66E9  5D        1267                    LD      E, L
66EA  DD4E02    1268                    LD      C, (IX+LENGTH)
66ED  DD4603    1269                    LD      B, (IX+LENGTH+1)
66F0  DD7E00    1270                    LD      A, (IX+DIRECTION)
66F3  07        1271                    RLCA                    ;CALCULATE END ADDR
66F4  0F        1272                    RRCA
66F5  3803      1273                    JR      C, CB_SUB       ;IF A < 0
66F7  09        1274                    ADD     HL, BC
66F8  1802      1275                    JR      CB_CONT
66FA  ED42      1276  CB_SUB            SBC     HL, BC
66FC  CD4D64    1277  CB_CONT           CALL    GET_CHUNK       ;GET END CHUNK BIT
66FF  2F        1278                    CPL
6700  47        1279                    LD      B, A
6701  EB        1280                    EX      DE, HL
6702  CD4D64    1281                    CALL    GET_CHUNK       ;GET START CHUNK BIT
6705  2F        1282                    CPL
6706  4F        1283                    LD      C, A
6707  A8        1284                    XOR     B
6708  2816      1285                    JR      Z, CB_EXIT
670A  79        1286                    LD      A, C            ;HERE IF START AND END CHUNKS
670B  A0        1287                    AND     B               ; ARE NOT THE SAME
670C  47        1288                    LD      B, A            ;PUT START AND END BITS TOGETHER AND
670D  0E00      1289                    LD      C, 0            ; FILL IN BTEWEEN THEM WITH ZEROES
670F  37        1290                    SCF
6710  78        1291  CB_NB1           LD      A, B            ;TEST NEXT BIT
6711  CB11      1292                    RL      C
6713  A1        1293                    AND     C
6714  20FA      1294                    JR      NZ, CB_NB1      ;OTHERWISE, FOUND FIRST ZERO
6716  78        1295  CB_NB2           LD      A, B            ;TEST NEXT BIT
6717  CB11      1296                    RL      C
6719  A1        1297                    AND     C
671A  2804      1298                    JR      Z, CB_EXIT      ;FOUND LAST ZERO
671C  A8        1299                    XOR     B               ;OTHERWISE, UPDATE BITMAP
671D  47        1300                    LD      B, A
671E  18F6      1301                    JR      CB_NB2
6720  78        1302  CB_EXIT          LD      A, B            ;RETURN BITMAP
6721  C9        1303                    RET
                1304  ;
                1305  ;
                1306  ;  XFER_BYTES (DIRECTION, LENGTH, DEST_ADDR, SRC_ADDR, DEST_BANK,
                1307  ;              SRC_BANK) PASSED ON STACK IN ORDER SHOWN; STATUS_CODE: A)
                1308  ;
                1309  ;     ALL PARAMETERS ON STACK HAVE OFFSETS DEFINED ABOVE.
                1310  ;
                1311  ;
6722  F5        1312  XFER_BYTES      PUSH    AF              ;SAVE REGS
6723  C5        1313                  PUSH    BC
6724  D5        1314                  PUSH    DE
6725  E5        1315                  PUSH    HL
6726  210000    1316                  LD      HL, 0
6729  39        1317                  ADD     HL, SP
672A  110A00    1318                  LD      DE, 10
672D  19        1319                  ADD     HL, DE
672E  EB        1320                  EX      DE, HL          ;DE POINTS TO START OF PARMS
672F  3A1563    1321                  LD      A, (BS_MAX_BANK)
```

```
6732  4F          1322                    LD      C, A
6733  0600        1323                    LD      B, 0
6735  210000      1324                    LD      HL, 0
6738  39          1325                    ADD     HL, SP
6739  A7          1326                    AND     A
673A  ED42        1327                    SBC     HL, BC
673C  2B          1328                    DEC     HL
673D  2B          1329                    DEC     HL              ;HL = SP - MAX_BANK - 2
673E  E5          1330                    PUSH    HL
673F  DDE1        1331                    POP     IX              ;IX POINTS TO LOCATION TO SAVE STATUS
6741  DDF9        1332                    LD      SP, IX
6743  CD1E65      1333                    CALL    SAVE_STATUS     ;SAVE BANK'S' STATUS
6746  D5          1334                    PUSH    DE
6747  DDE1        1335                    POP     IX              ;IX NOW POINTS TO PARMS
6749  DD6E04      1336                    LD      L, (IX+SRC_ADDR)
674C  DD6607      1337                    LD      H, (IX+SRC_ADDR+1)
674F  CDE866      1338                    CALL    CREATE_BITMAP   ;GET SRC BITMAP
6752  F5          1339                    PUSH    AF              ;SAVE ON STACK TEMPORARILY
6753  DD6E04      1340                    LD      L, (IX+DEST_ADDR)
6756  DD6605      1341                    LD      H, (IX+DEST_ADDR+1)
6759  CDE866      1342                    CALL    CREATE_BITMAP   ;GET DEST BITMAP
675C  4F          1343                    LD      C, A            ;C = DEST BITMAP
675D  F1          1344                    POP     AF
675E  47          1345                    LD      B, A            ;B = SRC BITMAP
675F  DD7E09      1346                    LD      A, (IX+SRC_BANK)
6762  DD5608      1347                    LD      D, (IX+DEST_BANK)
6765  BA          1348                    CP      D               ;COMPARE SRC AND DEST BANK NUMBERS
6766  2005        1349                    JR      NZ, XB_DIFF_BANKS
6768  78          1350                    LD      A, B            ;HERE IF BANK NUMBERS ARE DIFFERENT
6769  A1          1351                    AND     C
676A  47          1352                    LD      B, A            ;B = UNION OR SRC AND DEST BITMAPS
676B  180B        1353                    JR      XB_DO_MOVE
676D  78          1354  XB_DIFF_BANKS     LD      A, B            ;CHECK FOR OVERLAP BETWEEN SRC AND
676E  B1          1355                    OR      C               ; DEST CHUNKS
676F  FEFF        1356                    CP      0FFH
6771  202D        1357                    JR      NZ, XB_OVERLAP
6773  58          1358                    LD      E, B            ;HERE IF NO OVERLAP
6774  42          1359                    LD      B, D
6775  CD9964      1360                    CALL    BANK_ENABLE     ;SELECT DEST BANK
6778  DD4609      1361  XB_DO_MOVE        LD      B, (IX+SRC_BANK)
677B  4B          1362                    LD      C, E
677C  CD9964      1363                    CALL    BANK_ENABLE     ;SELECT SRC BANK
677F  DD6E06      1364                    LD      L, (IX+SRC_ADDR)
6782  DD6607      1365                    LD      H, (IX+SRC_ADDR+1)
6785  DD5E04      1366                    LD      E, (IX+DEST_ADDR)
6788  DD5605      1367                    LD      D, (IX+DEST_ADDR+1)
678B  DD4E02      1368                    LD      C, (IX+LENGTH)
678E  DD4603      1369                    LD      B, (IX+LENGTH+1)
6791  DD7E00      1370                    LD      A, (IX+DIRECTION)
6794  07          1371                    RLCA
6795  0F          1372                    RRCA
6796  3804        1373                    JR      C, XB_REVERSE   ;IF A < 0
6798  EDB0        1374                    LDIR
679A  1852        1375                    JR      XB_EXIT
679C  EDB8        1376  XB_REVERSE       LDDR
679E  184E        1377                    JR      XB_EXIT
67A0  21C05C      1378  XB_OVERLAP       LD      HL, MSTBOT
67A3  C5          1379                    PUSH    BC
67A4  06FF        1380                    LD      B, 255
67A6  CD1663      1381                    CALL    GET_WORD
67A9  C1          1382                    POP     BC
67AA  110002      1383                    LD      DE, STKSZ
67AD  A7          1384                    AND     A
67AE  ED52        1385                    SBC     HL, DE          ;HL = ADDRESS OF STACK LIMIT
67B0  112000      1386                    LD      DE, FREE_BYTES
67B3  19          1387                    ADD     HL, DE
67B4  EB          1388                    EX      DE, HL          ;DE = SP_NEW
67B5  210000      1389                    LD      HL, 0
67B8  39          1390                    ADD     HL, SP          ;HL = SP_OLD
67B9  13          1391                    INC     DE              ;COMPARE SP_OLD WITH SP_NEW
67BA  A7          1392                    AND     A
67BB  ED52        1393                    SBC     HL, DE
67BD  3004        1394                    JR      NC, XB_SPACE    ;IF SP_OLD - SP_NEW > 0
67BF  3E01        1395                    LD      A, 1            ;RETURN ERROR CODE
67C1  182B        1396                    JR      XB_EXIT
67C3  1B          1397  XB_SPACE         DEC     DE
67C4  EB          1398                    EX      DE, HL
67C5  F9          1399                    LD      SP, HL          ;SET SP TO SP_NEW
67C6  13          1400                    INC     DE              ;DE = BUF_SZ
67C7  DD7E00      1401                    LD      A, (IX+DIRECTION)  ;HL = BUF_PTR
67CA  DD7500      1402                    LD      (IX+BUF_PTR), L    ;SAVE BUF_PTR ON STACK
67CD  DD7401      1403                    LD      (IX+BUF_PTR+1), H
67D0  DD6E02      1404                    LD      L, (IX+LENGTH)
67D3  DD6603      1405                    LD      H, (IX+LENGTH+1)
67D6  A7          1406  XB_MOVE_LOOP     AND     A               ;HL = BYTES LEFT TO MOVE
67D7  ED52        1407                    SBC     HL, DE          ;DE = BYTES TO MOVE THIS TIME
67D9  3805        1408                    JR      C, XB_LAST_MOVE ;IF LESS THAN BUF_SZ BYTES LEFT
67DB  CD8C66      1409                    CALL    MOVE_BYTES
67DE  18F6        1410                    JR      XB_MOVE_LOOP
67E0  19          1411  XB_LAST_MOVE     ADD     HL, DE
67E1  EB          1412                    EX      DE, HL
67E2  CD8C66      1413                    CALL    MOVE_BYTES
67E5  EB          1414                    EX      DE, HL
67E6  DD6E00      1415                    LD      L, (IX+BUF_PTR)
67E9  DD6601      1416                    LD      H, (IX+BUF_PTR+1)
67EC  19          1417                    ADD     HL, DE          ;HL = BUF_PTR+BUF_SZ
67ED  F9          1418                    LD      SP, HL          ;RESTORE STACK POINTER
67EE  AF          1419  XB_EXIT           XOR     A               ;RETURN CODE FOR SUCCESSFUL COMPLETION
67EF  DD210000    1420                    LD      IX, 0
67F3  DD39        1421                    ADD     IX, SP
67F5  CD4A65      1422                    CALL    RESTORE_STATUS  ;RESTORE STATE AND RETURN ZERO CODE
```

```
67F8  DD23      1423                    INC     IX
67FA  DDF9      1424                    LD      SP, IX
67FC  E1        1425                    POP     HL              ;RESTORE REGS
67FD  D1        1426                    POP     DE
67FE  C1        1427                    POP     BC
67FF  F1        1428                    POP     AF
6800  DDE1      1429                    POP     IX              ;CLEAN UP PARMS
6802  DDE3      1430                    EX      (SP), IX
6804  DDE1      1431                    POP     IX
6806  DDE3      1432                    EX      (SP), IX
6808  DDE1      1433                    POP     IX
680A  DDE3      1434                    EX      (SP), IX
680C  DDE1      1435                    POP     IX
680E  DDE3      1436                    EX      (SP), IX
6810  DDE1      1437                    POP     IX
6812  DDE9      1438                    EX      (SP), IX
6814  C9        1439                    RET
                1440  ;
                1441  ;
                1442  ;  GOTO_EXT_INIT (ADDR: HL)
                1443  ;
                1444  ;
6815  DDE1      1445  GOTO_EXT          POP     IX              ;TRASH RET ADDR
6817  F5        1446                    PUSH    AF
6818  DBFF      1447                    IN      A, (HREXPT)
681A  CBFF      1448                    SET     7, A
681C  D3FF      1449                    OUT     (HREXPT), A
681E  3E01      1450                    LD      A, 1
6820  D3F4      1451                    OUT     (DKHSPT), A
6822  F1        1452                    POP     AF
6823  E9        1453                    JP      (HL)
                1454                    END
```

```
                                FIXTBL
LOC   OBJ CODE M STMT SOURCE STATEMENT                                  ASM 5.9

                    1  DISPATCH         EQU     6200H
                    2  INT              EQU     62AEH
                    3  GET_WORD         EQU     6316H
                    4  PUT_WORD         EQU     633BH
                    5  GET_STATUS       EQU     6405H
                    6  GET_NUMBER       EQU     645EH
                    7  BANK_ENABLE      EQU     6499H
                    8  SAVE_STATUS      EQU     651EH
                    9  RESTORE_STATUS   EQU     654AH
                   10  BS_STACK         EQU     658EH
                   11  BS_SP            EQU     65CEH
                   12  GOTO_BANK        EQU     6572H
                   13  CALL_BANK        EQU     65D0H
                   14  MOVE_BYTES       EQU     668CH
                   15  CREATE_BITMAP    EQU     66E8H
                   16  XFER_BYTES       EQU     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-VERSA.  THE ADDRESSES IN THE TABLE ARE DEFINED AS CHUNK 3 ADDRESSES
                   21
                   22
1D00               23                  ORG     1D00H
                   24
                   25
1D00  3262         26  FIXTBL          DEFW    DISPATCH+32H
1D02  4D62         27                  DEFW    DISPATCH+4DH
1D04  7262         28                  DEFW    DISPATCH+72H
1D06  AB62         29                  DEFW    DISPATCH+0ABH
                   30
1D08  B862         31                  DEFW    INT+0AH
1D0A  CD62         32                  DEFW    INT+1FH
1D0C  D362         33                  DEFW    INT+25H
1D0E  DC62         34                  DEFW    INT+2EH
1D10  FB62         35                  DEFW    INT+4DH
                   36
1D12  1A63         37                  DEFW    GET_WORD+4H
1D14  2063         38                  DEFW    GET_WORD+0AH
1D16  2463         39                  DEFW    GET_WORD+0EH
1D18  2A63         40                  DEFW    GET_WORD+14H
1D1A  3563         41                  DEFW    GET_WORD+1FH
                   42
1D1C  3E63         43                  DEFW    PUT_WORD+3H
1D1E  4463         44                  DEFW    PUT_WORD+9H
1D20  4863         45                  DEFW    PUT_WORD+0DH
1D22  4E63         46                  DEFW    PUT_WORD+13H
1D24  5763         47                  DEFW    PUT_WORD+1CH
                   48
1D26  1764         49                  DEFW    GET_STATUS+12H
1D28  1E64         50                  DEFW    GET_STATUS+19H
1D2A  2864         51                  DEFW    GET_STATUS+23H
                   52
1D2C  6164         53                  DEFW    GET_NUMBER+3H
1D2E  6564         54                  DEFW    GET_NUMBER+7H
1D30  6D64         55                  DEFW    GET_NUMBER+0FH
```

```
1D36  B364        59              DEFW    BANK_ENABLE+1AH
1D38  0E65        60              DEFW    BANK_ENABLE+75H
1D3A  1665        61              DEFW    BANK_ENABLE+7DH
                  62
1D3C  3265        63              DEFW    SAVE_STATUS+14H
1D3E  3A65        64              DEFW    SAVE_STATUS+1CH
                  65
1D40  5C65        66              DEFW    RESTORE_STATUS+12H
1D42  6665        67              DEFW    RESTORE_STATUS+1CH
                  68
1D44  CE65        69              DEFW    BS_SP
                  70
1D46  8565        71              DEFW    GOTO_BANK+13H
                  72
1D48  D365        73              DEFW    CALL_BANK+3H
1D4A  ED65        74              DEFW    CALL_BANK+1DH
1D4C  F965        75              DEFW    CALL_BANK+29H
1D4E  1E66        76              DEFW    CALL_BANK+4EH
1D50  2D66        77              DEFW    CALL_BANK+5DH
1D52  3A66        78              DEFW    CALL_BANK+6AH
1D54  4266        79              DEFW    CALL_BANK+72H
1D56  5066        80              DEFW    CALL_BANK+80H
1D58  6666        81              DEFW    CALL_BANK+96H
1D5A  7266        82              DEFW    CALL_BANK+0A2H
1D5C  8066        83              DEFW    CALL_BANK+0B0H
                  84
1D5E  9466        85              DEFW    MOVE_BYTES+8H
1D60  C066        86              DEFW    MOVE_BYTES+34H
                  87
1D62  FD66        88              DEFW    CREATE_BITMAP+15H
1D64  0367        89              DEFW    CREATE_BITMAP+1BH
                  90
1D66  3067        91              DEFW    XFER_BYTES+0EH
1D68  4F67        92              DEFW    XFER_BYTES+2DH
1D6A  5067        93              DEFW    XFER_BYTES+2EH
1D6C  5A67        94              DEFW    XFER_BYTES+38H
1D6E  7667        95              DEFW    XFER_BYTES+54H
1D70  7D67        96              DEFW    XFER_BYTES+5BH
1D72  A767        97              DEFW    XFER_BYTES+85H
1D74  DC67        98              DEFW    XFER_BYTES+0BAH
1D76  E367        99              DEFW    XFER_BYTES+0C1H
1D78  F667       100              DEFW    XFER_BYTES+0D4H
                 101
1D7A  0000       102              DEFW    0       ;THIS IS THE TABLE TERMINATOR
```

APPENDIX B

System Variables Definition File

2068 HOME ROM

```
TS2000 HOME ROM                BASIC
 LOC   OBJ CODE M STMT SOURCE STATEMENT                              ASM 5.9

                    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  ; SYSTEM VARIABLES
                    28
                    29  L_LEN   EQU 32          ; # CHARS PER LINE ON THE DISPLAY
                    30  TV_LNS: EQU 24          ; NO. OF LINES ON TV SCREEN
                    31  D_FILE: EQU 4000H       ; ADDRESS OF DISPLAY FILE
                    32  ATTRS:  EQU D_FILE+L_LEN*TV_LNS*8       ; SCREEN ATTRIBUTES
                    33  PRBUFF: EQU ATTRS+TV_LNS*L_LEN  ; PRINTER BUFFER
                    34          ASSERT PRBUFF.AND.0FFH=0
                                COND .NOT.(PRBUFF.AND.0FFH=0)
                                ERROR IN ASSERTION PRBUFF.AND.0FFH=0
                                ENDC
```

```
KSTATE: EQU PRBUFF+L_LEN*8      ; SEE KB DOCUMENTATION
 KS_A:  EQU 0                   ;    1ST BYTE IS A CHAR KEY PRESSED
 KS_C:  EQU 1                   ;    2ND IS TIME TILL COUNTS AS RELEASED
 KS_B:  EQU 2                   ;    3RD IS TIME (IN FRAMES) TILL REPEAT
 KS_D:  EQU 3                   ;    4TH IS CODE WHEN REPEATS
                                ;    5TH - 8TH ARE A SECOND SET OF 1ST FOUR
LAST_K: EQU KSTATE+8            ; NEWLY PRESSED KEY
REPDEL: EQU LAST_K+1            ; DELAY BEFORE 1ST REPEAT (INITIALIZED TO 35)
REPPER: EQU REPDEL+1            ; DELAY BEFORE SUBSEQUENT REPEATS (INITIALIZED TO 5)
DEFADD: EQU REPPER+1            ; -> CHAR AFTER '(' IN FORMAL PARAMETER LIST: MUST BE
                                ;    0 WHEN NO USER-DEFINED FN BEING EVALUATED
K_DATA: EQU DEFADD+2            ; DATA BYTE IN COMPOSITE CHAR FROM KEYBOARD
TVDATA: EQU K_DATA+1            ; USED FOR STORING BYTES IN COMPOSITE CHARACTERS:
                                ;    (TVDATA) = KEY BYTE,
                                ;    (TVDATA+1) = 1ST DATA BYTE FOR AT OR TAB.
STRMS:  EQU TVDATA+2            ; STREAM DATA: POINTERS (OFFSETS FROM (CHANS)-1) TO
                                ;    CHANNELS.  0 = STREAM NOT OPEN.
 HIDSTR:EQU 3                   ; NO. STREAMS HIDDEN FROM USER. THESE ARE TIED
                                ;   UNALTERABLY TO SPECIFIC CHANNELS.
 HID_K: EQU -3                  ; KEYBOARD
 HID_S: EQU -2                  ; TV, UPPER HALF OF SCREEN
 HID_R: EQU -1                  ; INSERTION IN RAM
 COM_ST:EQU 0                   ; STREAM FOR COMMANDS
 INP_ST:EQU 1                   ; STREAM FOR INPUT DATA
 PR_ST: EQU 2                   ; STREAM FOR PRINT
 LPR_ST:EQU 3                   ; STREAM FOR LPRINT
CHARS:  EQU STRMS+(HIDSTR+16)*2 ; -> 8*20H BYTES BEFORE CHARACTER SET
FART:   EQU CHARS+2             ; NO. CYCLES OF ERROR NOISE (2 8VES BELOW MIDDLE C)
PIP:    EQU FART+1              ; NO. CYCLES OF KEYBOARD NOISE (3 8VES ABOVE MIDDLE C)
Y:      EQU PIP+1               ; VALUE ALWAYS HELD IN IY
ERR_NR  EQU Y                   ; [RUN TIME ERROR #] - 1
FLAGS:  EQU ERR_NR+1            ; VARIOUS FLAGS
 SPC:   EQU 0                   ;    SUPPRESS SPACE BEFORE TOKENS
 PR:    EQU 1                   ;    PRINTING TO PRINTER, NOT TV
 LMODE1:EQU 2                   ;    L MODE, NOT K, AT CURRENT CHARACTER
 LMODE: EQU 3                   ;    L MODE, NOT K, AT CURSOR
 KEYHIT:EQU 5                   ;    KEYHIT FOUND
 NO:    EQU 6                   ;    EXPRESSION IS NUMERICAL, NOT STRING
 INTPT: EQU 7                   ;    REQ INTERPRET RATHER THAN CHECK SYNTAX
TVFLAG: EQU FLAGS+1             ; FLAGS ASSOCIATED WITH THE TV
 LHS:   EQU 0                   ;    PRINTING TO LOWER HALF OF SCREEN
 EDIT:  EQU 1                   ;    OUTPUTTING LINE FOR EDIT OR NO. FOR STRING
 ECHREQ:EQU 3                   ;    ECHO REQUESTED IF INPUTTING FROM KEYBOARD
 LIST:  EQU 4                   ;    OUTPUTTING AN AUTOMATIC LISTING
 CLHS:  EQU 5                   ;    CLEAR LOWER HALF WHEN KEY PRESSED
ERR_SP: EQU TVFLAG+1            ;-> BOTTOM ITEM ON MACHINE STACK.
LISTSP: EQU ERR_SP+2            ; -> RETURN ADDRESS FROM AUTOMATIC LISTING
MODE:   EQU LISTSP+2            ; 0 = K OR L, 1 = F, 2 = G.
NEWPPC: EQU MODE+1              ; LINE TO BE JUMPED TO
NSPPC:  EQU NEWPPC+2            ; SUBLINE TO BE JUMPED TO (BIT 7 OFF FORCES JUMP)
PPC     EQU NSPPC+1             ; LINE # OF INSTR BEING INTERPRETED
SUBPPC: EQU PPC+2               ; NO. WITHIN LINE OF INSTR BEING INTERPRETED
BORDCR: EQU SUBPPC+1            ; BORDER COLOUR (SHIFTED LEFT BACKG BITS WITH 0S IN
                                ;    BITS 0-2 & 6-7)
E_PPC   EQU BORDCR+1            ; LINE # OF "CURRENT" LINE IN LISTING
                                ;THE VARIABLES FROM (VARS) UP TO & INCLUDING (STKEND)
                                ;    ARE 'MOVABLE' IN THE SENSE THAT THEY ARE ADJUSTED
                                ;    (BY REMGSZ IN MODULE EDIT) WHENEVER STUFF IS
                                ;    INSERTED IN OR DELETED FROM RAM.
VARS    EQU E_PPC+2             ; -> 1ST RECORD FOR A VARIABLE (LAST IS 1 BYTE 80H)
DEST    EQU VARS+2              ; -> VAR MATCHED BY TEMPL CODE 1 OR 4 (TEXT OR RECORD)
CHANS_: EQU DEST+2              ; -> CHANNEL DATA (INCLUDING FLOPPY BUFFERS).
                                ;    EACH ITEM COMPRISES:
                                ;    THE ADDRESS OF AN OUTPUT ROUTINE FOR WRCH,
                                ;    THE ADDRESS OF AN INPUT ROUTINE FOR INCH,
                                ;    A 1-BYTE CODE FOR THE DEVICE TYPE,
                                ;    &, WHERE APPROPRIATE, A FILE NAME, ADDITIONAL
                                ;    DATA & A BUFFER.
CURCHL: EQU CHANS_+2            ; -> DATA FOR CURRENT CHANNEL
PROG:   EQU CURCHL+2            ; -> BASIC PROGRAM
NXTLIN: EQU PROG+2              ; -> NEXT LINE OF SOURCE CODE
```

```
106  DATADD: EQU NXTLIN+2     ; -> TERMINATOR OF LAST DATA ITEM
107  E_LINE  EQU DATADD+2     ; -> LINE BEING EDITED
108  K_CUR:  EQU E_LINE+2     ; -> CURRENT CHAR IN INPUT BUFFER
109  CH_ADD  EQU K_CUR+2      ; -> CURRENT CHAR WHEN SYNTAX CHECKING ETC
110  X_PTR   EQU CH_ADD+2     ; -> 1ST CHAR NOT SYNTACTICALLY OK (0 IF ALL OK)
111                           ;    ALSO STORES (CH_ADD) DURING READ & INPUT
112  WORKSP: EQU X_PTR+2      ; -> TEMPORARY WORK SPACE
113  STKBOT: EQU WORKSP+2     ; -> BOTTOM OF CALCULATOR STACK
114  STKNXT: EQU STKBOT+2     ; -> NEXT FREE PLACE ON CALCULATOR STACK
115  STKEND: EQU STKNXT       ; ALTERNATIVE NAME
116
117  BREG:   EQU STKEND+2     ; KEEPS VALUE OF CALCULATOR B REGISTER
118  MEM:    EQU BREG+1       ; -> AREA USED BY CALCTR INSTRS MEMORY & COPY
119  FLAGS2: EQU MEM+2        ; MORE FLAGS
120   ALOS:  EQU 0            ;    AUTOMATIC LISTING ON SCREEN
121   PRLEFT:EQU 1            ;    PRINTER BUFFER NOT EMPTY
122   L_STR: EQU 2            ;    INSIDE STRING WHEN DOING KB MODE IN LISTCH
123   CAPS_L:EQU 3            ;    CAPITALS SHIFT LOCK ON
124   RETPOS:EQU 4            ;    RETYPE POSSIBLE AFTER SYNTAX ERROR
125   DELREP:EQU 5            ;    DELETE KEY REPEAT (KEY HELD DOWN)
126  DF_SZ   EQU FLAGS2+1     ; # LINES IN 2ND HALF OF SCREEN INC SEP'G BLANK LINE
127  S_TOP   EQU DF_SZ+1      ; LINE # (IN PROGRAM) OF TOP LINE ON SCREEN
128  OLDPPC  EQU S_TOP+2      ; LINE # OF E.G. INTERRUPTED STMT
129  OSPPC:  EQU OLDPPC+2     ; (OLD SUB PPC) STATEMENT NO. WITHIN LINE FOR OLDPPC
130  FLAGX:  EQU OSPPC+1      ; FLAGS ASSOCIATED WITH ASSIGNMENT
131   FLEX:  EQU 0            ;    FLEXIBLE LENGTH ASSIGNMENT REQUIRED
132   UNFND: EQU 1            ;    DESTINATION OF ASSIGNMENT NOT FOUND
133   INPLN: EQU 5            ;    REQ INPUT VALUE RATHER THAN LINE OF PROGRAM
134  ;NO:    EQU 6            ;    REQD TYPE IS NUMERIC
135   LINPLN:EQU 7            ;    LINPUT (INPUT LINE) RATHER THAN STRAIGHT INPUT
136  STRLEN: EQU FLAGX+1      ; LENGTH OF DESTINATION WHEN STRING TYPE
137  T_ADDR  EQU STRLEN+2     ; -> NEXT BYTE IN TEMPLATE
138  SEED    EQU T_ADDR+2     ; LAST RANDOM # BEFORE SCALING
139  FRAMES: EQU SEED+2       ; LS 2 BYTES OF 3-BYTE FRAME COUNTER
140  FRAME2: EQU FRAMES+2     ; MS BYTE OF 3-BYTE FRAME COUNTER
141  UDG:    EQU FRAME2+1     ; -> 1ST USER DEFINED GRAPHIC
142  COORDS: EQU UDG+2        ; COORDINATES OF LAST PLOT ETC.: (COORDS) = X-COORD.,
143                           ;    (COORDS+1) = Y-COORD.
144  P_POSN: EQU COORDS+2     ; COLUMN NO. OF PRINTER POSN
145  PR_CC:  EQU P_POSN+1     ; LS BYTE OF ADDRESS OF NEXT CHAR FOR PRINTER
146  ECHO_E: EQU PR_CC+2      ; COORDS IN LOWER HALF OF END OF KEYBOARD INPUT BUFFER
147  DF_CC   EQU ECHO_E+2     ; -> SCREEN CHAR UNDER PRINT CURSOR
148  DFCCL:  EQU DF_CC+2      ; LIKE DF_CC FOR LOWER HALF
149  S_POSN  EQU DFCCL+2      ; SCREEN POSN (COL & LINE) OF NEXT CHAR TO BE OUTPUT
150  SPOSNL: EQU S_POSN+2     ; LIKE S_POSN FOR LOWER HALF
151  SCR_CT: EQU SPOSNL+2     ; (SCROLL COUNT) DECREMENTED FOR EACH SCROLL
152  ATTR_P: EQU SCR_CT+1     ; CURRENT PERMANENT PRINTING ATTRIBUTES
153   FOREG: EQU 0            ;    LS BIT OF FOREGROUND COLOUR
154    BLUE: EQU 0            ;        (INK)
155    RED:  EQU 1
156    GREEN:      EQU 2
157   BACKG: EQU 3            ;    LS BIT OF BACKGROUND COLOUR
158    BLUEB:  EQU 3          ;        (PAPER)
159    REDB:   EQU 4
160    GREENB: EQU 5
161   HILITE: EQU 6           ;    BRIGHT
162   FLASH:  EQU 7           ;    FLASH
163  MASK_P: EQU ATTR_P+1     ; CURRENT PERMANENT PRINTING ATTRIBUTES MASK:
164                           ;    0 FOR NEW, 1 FOR OLD
165  ATTR_T: EQU MASK_P+1     ; CURRENT TEMP. PRINTING ATTRIBUTES (BITS AS ATTR_P)
166  MASK_T: EQU ATTR_T+1     ; CURRENT TEMPORARY PRINTING ATTRIBUTES MASK
167  P_FLAG: EQU MASK_T+1     ; ADDITIONAL FLAGS FOR PRINTING: TEMPORARY FLAGS IN
168                           ;    EVEN BITS, PERMANENT FLAGS IN ODD BITS
169   XOR_CH:EQU 0            ;    NEW CHARS XOR'D INTO OLD RATHER THAN BEING LOADED
170   INV_CH:EQU 2            ;    NEW CHARS INVERTED
171   F_CB:  EQU 4            ;    FOREGROUND := COMPLEMENT OF BACKGROUND
172   B_CF:  EQU 6            ;    BACKGROUND := COMPLEMENT OF FOREGROUND
173  MEMBOT: EQU P_FLAG+1     ; BOTTOM OF CALCULATOR MEMORY (6 NUMBERS)
174  NMIADD: EQU MEMBOT+30    ; -> USER'S NMI SERVICE ROUTINE
175  RAMTOP: EQU NMIADD+2     ; LAST ADDRESS OF BASIC SYSTEM AREA
176  P_RAMT: EQU RAMTOP+2     ; -> LAST BYTE OF PHYSICAL RAM
```

```
177
178                     ;**** ADDITIONAL
179  ERR_LN: EQU P_RAMT+2     ;POINTER TO ON ERROR LINE NUMBER FOR A GO-TO.
180  ERR_C:  EQU ERR_LN+2     ;STORE LINE NUMBER IN WHICH ERROR OCCURRED
181  ERR_S:  EQU ERR_C+2      ;STORES STATEMENT NUMBER IN WHICH ERROR OCCURRED
182  ERR_T:  EQU     ERR_S+1  ;STORE FOR 'ERROR TYPE' AFTER A 'ON ERR'
183  SYSCON: EQU ERR_T+1      ;SYSTEM CONFIGURATION TABLE.
184  MAX_BANK: EQU SYSCON+2   ;LARGEST BANK NUMBER ASSIGNED
185  CURCBN: EQU MAX_BANK+1   ;BANK NUMBER OF THE CURRENT CHANNEL
186  MSTBOT: EQU CURCBN+1     ;ADDRESS OF LOCATION ABOVE MACHINE STACK
187  VIDMOD: EQU MSTBOT+2
188  ;                        ; NOTE:  UNUSED BYTE AFTER VIDMOD
189  ;
190  ARSBUF: EQU VIDMOD+2     ;POINTER TO AROS BUFFER.
191  ARSFLG: EQU ARSBUF+2     ;AROS FLAG - BIT 7 SET INDICATES AROS PRESENT.
192                           ;BIT 4 SET INDICATES NXTLIN POINTING TO AROS.
193                           ;BIT 3 SET INDICATES DATADD POINTING TO AROS.
194                           ;THESE BITS BECOME IMPORTANT FOR THE INSERT ROUTINE
195                           ;(POINTERS POINTING TO AROS SHOULD NOT BE UPDATED
196                           ;FOR AN INSERTION INTO RAM).
197  ADATLN: EQU ARSFLG+1     ;POINTER TO THE START OF THE CURRENT DATA LINE
198                           ; (AROS ONLY)
199  DTLNLN: EQU ADATLN+2     ;LENGTH OF THE CURRENT DATA LINE (AROS ONLY).
200  STRMNM: EQU DTLNLN+2     ;CURRENT STREAM NUMBER, USED FOR BUS EXPANSION
201                           ; UNIT DEVICES.
202  MSTACK: EQU     6200H    ;LOCATION ABOVE MACHINE STACK
203  DRIVES: EQU     6840H    ;START OF 'DRIVES' AREA
204  BANK_ENABLE     EQU     6499H
205  CALL_BANK       EQU     65D0H
206  MOVE_SZ         EQU     DRIVES-6000H
207  DEST7           EQU     0FFFFH-MOVE_SZ+1
208  FIX             EQU     DEST7-6000H
209  CALL_VBANK      EQU     CALL_BANK+FIX
210  GOTO_BANK       EQU     6572H    ;ADDRESS OF "GO TO BANK" BANK SWITCHING
211                                   ; AWARD.
212  XFER_BYTES      EQU     6722H    ;INDIRECT DATA TRANSFER BETWEEN BANKS.
213  GOTO_EXT        EQU     6815H    ;FOR INITIALIZATION CODE IN HOME BANK
214                                   ; EXTENTION.
215  SLVM            EQU     01ABH    ;ADDRESS OF TAPE ROUTINES FOR SAVE, LOAD
216                                   ; VERIFY AND MERGE COMMANDS.
217  BLDSCT          EQU     09F4H    ;ADDRESS OF INITIALIZATION ROUTINE TO
218                                   ; BUILD THE SYSTEM CONFIGURATION TABLE.
219  RESSCI          EQU     0C4CH    ;ADDRESS OF RESET ROUTINE TO ADD DEVICES.
220  PASSING         EQU     0F09H    ;ADDRESS OF ROUTINE TO PUSH PARAMETERS TO
221                                   ; THE BEU ROUTINES ONTO THE MACHINE STACK.
222                     ;****
223
224
225  ; OTHER EQUATES
226
227                     ;RESTARTS
228
229  ERROR:  EQU 8
230  WRCH:   EQU 16
231  IGN_SP: EQU 24
232  NXT_IS: EQU 32
233  CALCTR: EQU 40
234  COPYUP: EQU 48
235
236  NOSIZE  EQU 5            ; # OF BYTES IN A FLOATING POINT NUMBER
237  DIGIT   EQU '0'          ; DIGIT+N IS CODE FOR DIGIT N
238  LETTER  EQU 0            ; LETTER+'ALPHA' IS CODE FOR LETTER ALPHA
239  DEBDEL: EQU 5            ; NO. CONSECUTIVE TIMES KB SWITCH FOUND OPEN BEFORE
240                           ;    KEY RECKONED RELEASED.
241
242                     ;CONTROL CHARACTERS (APPEARING ON STREAM)
243
244  COM_CC: EQU  6           ; PRINT COMMA
245  EDT_CC: EQU  7           ; EDIT
246  BS_CC:  EQU  8           ; BACKSPACE (CURSOR LEFT)
247  CRT_CC: EQU  9           ; CURSOR RIGHT
248  CD_CC:  EQU 0AH          ; CURSOR DOWN
249  CU_CC:  EQU 0BH          ; CURSOR UP
```

```
250  RUB_CC: EQU OCH          ; RUBOUT
251  CR_CC:  EQU ODH          ; CARRIAGE RETURN (NEWLINE)
252  NL:     EQU CR_CC
253  SLUG:   EQU OEH          ; PRECEDES 5 BYTES OF SLUG
254  FORECC: EQU 10H          ; FOREGROUND
255                           ; THE CONTROL CHARS FOR FORE, BACK, FLASH, BRIGHT,
256                           ;    INVERT & OVER ARE CONSECUTIVE IN THAT ORDER.
257  AT_CC:  EQU 16H          ; PRINT AT
258  TAB_CC: EQU 17H          ; PRINT TAB
259
260                    ; CONTROL CHARACTERS (RECEIVED FROM KEYBOARD)
261
262  STY_KC: EQU  0           ; STEADY
263  FSH_KC: EQU  1           ; FLASH
264  LOL_KC: EQU  2           ; LOWLIGHT
265  HIL_KC: EQU  3           ; HIGHLIGHT
266  NLV_KC: EQU  4           ; NORMAL VIDEO
267  INV_KC: EQU  5           ; INVERSE VIDEO
268  CSL_KC: EQU  6           ; CAPS SHIFT LOCK TOGGLE
269  TM_KC:  EQU OEH          ; TOKEN MODE
270  GRM_KC: EQU OFH          ; GRAPHICS MODE
271  FG_KC:  EQU 10H          ; FOREGROUND BLACK
272  BG_KC:  EQU 18H          ; BACKGROUND BLACK
273
274  SPACE:  EQU ' '
275  QUOTE   EQU '"'          ; STRING QUOTE
276  DOLLAR  EQU '$'          ; DOLLAR SIGN
277  COLON:  EQU ':'
278  COMMA   EQU ','
279  KET     EQU ')'
226
227                    ;RESTARTS
228
229  ERROR:  EQU 8
230  WRCH:   EQU 16
231  IGN_SP: EQU 24
232  NXT_IS: EQU 32
233  CALCTR: EQU 40
234  COPYUP: EQU 48
235
236  NOSIZE  EQU 5            ; # OF BYTES IN A FLOATING POINT NUMBER
237  DIGIT   EQU '0'          ; DIGIT+N IS CODE FOR DIGIT N
238  LETTER  EQU 0            ; LETTER+'ALPHA' IS CODE FOR LETTER ALPHA
239  DEBDEL: EQU 5            ; NO. CONSECUTIVE TIMES KB SWITCH FOUND OPEN BEFORE
240                           ;    KEY RECKONED RELEASED.
241
242                    ;CONTROL CHARACTERS (APPEARING ON STREAM)
243
244  COM_CC: EQU  6           ; PRINT COMMA
245  EDT_CC: EQU  7           ; EDIT
246  BS_CC:  EQU  8           ; BACKSPACE (CURSOR LEFT)
247  CRT_CC: EQU  9           ; CURSOR RIGHT
248  CD_CC:  EQU OAH          ; CURSOR DOWN
249  CU_CC:  EQU OBH          ; CURSOR UP
250  RUB_CC: EQU OCH          ; RUBOUT
251  CR_CC:  EQU ODH          ; CARRIAGE RETURN (NEWLINE)
252  NL:     EQU CR_CC
253  SLUG:   EQU OEH          ; PRECEDES 5 BYTES OF SLUG
254  FORECC: EQU 10H          ; FOREGROUND
255                           ; THE CONTROL CHARS FOR FORE, BACK, FLASH, BRIGHT,
256                           ;    INVERT & OVER ARE CONSECUTIVE IN THAT ORDER.
257  AT_CC:  EQU 16H          ; PRINT AT
258  TAB_CC: EQU 17H          ; PRINT TAB
259
260                    ; CONTROL CHARACTERS (RECEIVED FROM KEYBOARD)
261
262  STY_KC: EQU  0           ; STEADY
263  FSH_KC: EQU  1           ; FLASH
264  LOL_KC: EQU  2           ; LOWLIGHT
265  HIL_KC: EQU  3           ; HIGHLIGHT
266  NLV_KC: EQU  4           ; NORMAL VIDEO
267  INV_KC: EQU  5           ; INVERSE VIDEO
```

```
268  CSL_KC: EQU  6          ; CAPS SHIFT LOCK TOGGLE
269  TM_KC:  EQU 0EH         ; TOKEN MODE
270  GRM_KC: EQU 0FH         ; GRAPHICS MODE
271  FG_KC:  EQU 10H         ; FOREGROUND BLACK
272  BG_KC:  EQU 18H         ; BACKGROUND BLACK
273
274  SPACE:  EQU ' '
275  QUOTE   EQU '"'         ; STRING QUOTE
276  DOLLAR  EQU '$'         ; DOLLAR SIGN
277  COLON:  EQU ':'
278  COMMA   EQU ','
279  KET     EQU ')'
280  BRA     EQU '('
281  GT:     EQU '>'
282  MINUS   EQU '-'
283  EQUAL   EQU '='
284  PLUS:   EQU '+'
285  STROKE: EQU '/'
286  POWER:  EQU '^'
287  POINT:  EQU '.'
288  SHARP:  EQU  5FH        ; PRESTEL CODE FOR '#'
289  STD_GR: EQU  80H        ; 1ST STANDARD GRAPHIC
290  UD_GR:  EQU  90H        ; 1ST USER-DEFINED GRAPHIC
291
292                  ;TOKENS
293
294  TOK0:   EQU 0A5H        ; 1ST TOKEN
295  RNDTOK: EQU 0A5H        ; 'RND'
296  INKEY:  EQU 0A6H        ; 'INKEY$'
297  PI:     EQU 0A7H        ; 'PI'
298  FN_TK:  EQU 0A8H        ; 'FN'
299  PNT_TK: EQU 0A9H        ; 'POINT'
300  SCRNTK: EQU 0AAH        ; 'SCREEN$'
301  ATTRTK: EQU 0ABH        ; 'ATTRT'
302  AT:     EQU 0ACH        ; 'AT'
303  TOK_FN: EQU FN_TK       ; 1ST TOKEN TO REQUIRE A SPACE AFTER
304  TAB:    EQU 0ADH        ; 'TAB'
305  VALSTK: EQU 0AEH        ; 'VAL$'
306  LO_MON: EQU 0AFH        ; TOKEN FOR 1ST MONADIC OPTR AFTER VAL$ (CODE)
307  BIN_TK: EQU 0C4H        ; 'BIN'
308  OR_TK:  EQU 0C5H        ; 'OR' NB THE TOKENS FOR OR, AND, <=, >=, <> ARE
309                          ;    CONSECUTIVE IN THAT ORDER.
310  LINETK: EQU 0CAH        ; 'LINE'
311  THEN:   EQU 0CBH        ; 'THEN'
312  TO:     EQU 0CCH        ; 'TO'
313  STEP:   EQU 0CDH        ; 'STEP'
314  DEF_TK: EQU 0CEH        ; 'DEF'
315  MIN_KW: EQU DEF_TK      ; 1ST TOKEN THAT IS A KEYWORD RATHER THAN OPERATOR
316  CAT_TK: EQU 0CFH        ; 'CAT'
317  FORMTK: EQU 0D0H        ; 'FORMAT'
318  MOVETK: EQU 0D1H        ; 'MOVE'
319  DEL_TK: EQU 0D2H        ; 'DELETE'
320  OPN_TK: EQU 0D3H        ; 'OPEN'
321  CLO_TK: EQU 0D4H        ; 'CLOSE'
322  MGE_TK: EQU 0D5H        ; 'MERGE'
323  VFY_TK: EQU 0D6H        ; 'VERIFY'
324  BEEPTK: EQU 0D7H        ; 'BEEP'
325  ARC_TK: EQU 0D8H        ; 'ARC'
326  FGTOK:  EQU 0D9H        ; 'FOREGROUND' NB THE TOKENS FOR FORE, BACK, FLASH,
327                          ;    BRIGHT, INVERT & OVER ARE CONSECUTIVE IN THAT
328                          ;    ORDER.
329  INVTOK: EQU FGTOK+5     ; 'INVERT'
330  OUT_TK: EQU 0DFH        ; 'OUT'
331  LPR_TK: EQU 0E0H        ; 'LPRINT'
332  LL_TK:  EQU 0E1H        ; 'LLIST'
333  STOPTK: EQU 0E2H        ; 'STOP'
334  READTK: EQU 0E3H        ; 'READ'
335  DATATK: EQU 0E4H        ; 'DATA'
336  RESTTK: EQU 0E5H        ; 'RESTORE'
337  NEXTOK  EQU 0F3H        ; 'NEXT'
338  DUMPTK: EQU 0FFH        ; 'COPY'
339
340  BORDPT: EQU 0FEH        ; OUTPUT PORT FOR SETTING BORDER COLOUR
```

```
PR_IN:   EQU OFBH          ; FOR INPUT FROM PRINTER
PR_OUT:  EQU OFBH          ; FOR OUTPUT TO PRINTER
KB_PT:   EQU OFEH          ; INPUT PORT FOR READING KEYBOARD
O_PORT:  EQU OFEH          ; OUTPUT PORT FOR TAPE
I_PORT:  EQU OFEH          ; INPUT PORT FOR TAPE
 TAPE_I:EQU 6              ;    TAPE INPUT BIT IN (I_PORT)
                    ;***ADDITIONAL
DKHSPT:  EQU OF4H          ; DOCK HORIZONTAL SELECT PORT
BDATPT:  EQU OFCH          ; EXPANSION BANK DATA PORT
BCMDPT:  EQU OFDH          ; EXPANSION BANK COMMAND PORT
HREXPT:  EQU OFFH          ; HOME ROM EXPANSION BANK PORT
                    ;***

                    ;OFFSETS FROM (CHANS) OF PERMANENT CHANNELS

CHAN_K:  EQU 0             ; KEYBOARD
CHAN_S:  EQU 5             ; TV SCREEN (UPPER HALF)
CHAN_R:  EQU 10            ; RAM INSERTION
CHAN_P:  EQU 15            ; ZX PRINTER

CH_SET:  EQU 4000H-96*8    ; ADDRESS OF CHARACTER SET (STARTING WITH SPACE)
*EJECT

                   ;CALCULATOR COMMANDS. IN 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 STACK POINTERS HL & DE (BUT
                   ; NOT (STKNXT)) DECREMENTED FOR THEM BY CALCTR BEFORE THEY
                   ; ARE CALLED (STKDWN).

IFJUMP:  EQU 0    ;S,T -> S: RELATIVE JUMP CONDITIONAL ON VALUE OF T.
EXCH:    EQU IFJUMP+1     ;(EXCHANGE) S,T -> T,S
LOSE:    EQU EXCH+1       ;S,T -> S
SUB:     EQU LOSE+1       ;(SUBTRACT) S,T -> S-T
TIMES:   EQU SUB+1        ;S,T -> S*T
DIV:     EQU TIMES+1      ;(DIVIDE) S,T -> S/T
POWER:   EQU DIV+1        ;S,T -> S**T
OR:      EQU POWER+1      ;S,T -> S OR T (SEE OR).
AND:     EQU OR+1         ;S,T -> NUMERICAL S AND T (SEE NOAND).
GT:      EQU AND+4        ;S,T -> NUMERICAL S>T
                          ;5 NUMERIC COMPARISON OPERATIONS HAVE NOT BEEN GIVEN
                          ; MNEMONICS. S,T -> S^T WHERE ^ IS <=,>=,<>,>,< OR =
                          ; SEE CMPRSN.
ADD:     EQU AND+7        ;S,T -> S+T
STGAND:  EQU ADD+1        ;S$,T -> S$ AND$ T (SEE STGAND).
                          ;6 STRING COMPARISON OPERATIONS WITHOUT MNEMONICS.
CONCAT:  EQU STGAND+7     ;S$,T$ -> S$ +$ T$

                   ;ORDINARY OPERATIONS WITHOUT STKDWN.

VALS:    EQU CONCAT+1     ;T$ -> VAL$ T$
USRS:    EQU VALS+1       ;T$ -> ADDRESS OF BIT PATTERN FOR CORRESPONDING
                          ;    USER-DEFINED GRAPHIC
INKEY:   EQU USRS+1       ;T -> INKEY$ #T
NEGATE:  EQU INKEY+1      ;T -> -T
CODE:    EQU NEGATE+1     ;T$ -> CODE T$
LO_MON:  EQU CODE         ;OPERATION CODE FOR LO_MON
VAL:     EQU CODE+1       ;T$ -> VAL T$
LEN:     EQU VAL+1        ;T$ -> LEN T$
SIN:     EQU LEN+1        ;T -> SIN T
COS:     EQU SIN+1        ;T -> COS T
TAN:     EQU COS+1        ;T -> TAN T
ASN:     EQU TAN+1        ;T -> ARCSIN T
ACS:     EQU ASN+1        ;T -> ARCCOS T
ATN:     EQU ACS+1        ;T -> ARCTAN T
LN:      EQU ATN+1        ;T -> LN T
EXP:     EQU LN+1         ;T -> EXP T
INT:     EQU EXP+1        ;(INTEGER PART) T -> INT T
ROOT:    EQU INT+1        ;T -> SQUARE ROOT OF T
SGN:     EQU ROOT+1       ;T -> SGN T
```

```
413  ABS:     EQU SGN+1          ;(ABSOLUTE) T -> \T\
414  PEEK:    EQU ABS+1          ;T -> PEEK T
415  IN:      EQU PEEK+1         ;T -> IN T
416  USR:     EQU IN+1           ;T -> USR T
417  STR:     EQU USR+1          ;T -> STR$ T
418  CHR:     EQU STR+1          ;T -> CHR$ T
419  NOT:     EQU CHR+1          ;T -> BOOLEAN (T = 0)
420  ZERO?:   EQU NOT
421  DUP:     EQU NOT+1          ;(DUPLICATE) T -> T,T
422  INTDIV:  EQU DUP+1          ;(INTEGER DIVISION) S,T -> S MOD T, INT(S/T)
423  JUMP:    EQU INTDIV+1       ;PROGRAMME CONTROL - RELATIVE JUMP BY FOLLOWING BYTE
424  LITERAL:EQU JUMP+1          ;STACKS FOLLOWING NUMBER.
425  LOOP:    EQU LITERAL+1      ;LIKE ZILOG DJNZ
426  MINUS?:  EQU LOOP+1         ;T -> BOOLEAN (T < 0)
427  PLUS?:   EQU MINUS?+1       ;T -> BOOLEAN (T > 0)
428  QUIT:    EQU PLUS?+1        ;RETURNS CONTROL TO Z80
429  ANGLE:   EQU QUIT+1         ;T -> Y WHERE -1 <= Y <= +1 & SIN T = SIN (PI/2*Y)
430                              ;  MEMORY 0 := TRUE IF T IN 2ND OR 3RD QUADRANT
431  TRUNC:   EQU ANGLE+1        ;(TRUNCATE) T -> INTEGER TRUNCATION OF T TOWARDS 0.
432  XEQTB:   EQU TRUNC+1        ;EXECUTES (BREG) AS A CALCULATOR INSTRUCTION
433  XEY:     EQU XEQTB+1        ;S,T -> S * 10**T
434  FLOAT:   EQU XEY+1          ;T 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   MINUS1:EQU CONST+6         ;CALCTR CONSTANT EQUAL TO -1
441  COPY:    EQU CONST+20H      ;T -> T; T COPIED TO NTH CALCULATOR MEMORY
442  MEMORY:  EQU COPY+20H       ;T -> 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.0C0H         ;OPERATION CODE FOR LO_MON, TOP 2 BITS SET.
447  LONOMO:  EQU OP_TK+SIN      ; TOKEN FOR 1ST (NUMBER) NUMBER OPTR AFTER -
448  HINOMO:  EQU OP_TK+USR      ; TOKEN FOR LAST (NUMBER) NUMBER OPTR
449
450  #LIST ON
```

APPENDIX C

The entirety of Appendix C (pages 158 to 287) has been excluded primarily because of its length and because of the poor print quality. My OCR software would not accept it and including these pages as images would unacceptably expand the girth of this file.

Appendix C-1: Assembly source to support the 64 column mode
Appendix C-2: Assembly source to support 80 columns in the 64 column mode
Appendix C-3: Assembly source to support 40 columns in the 32 column mode
Appendix C-4: Assembly source to support the dual screen mode
Appendix C-5: Assembly source for sprite graphics in the 32 column mode

Much of this software is still bugged. Appendix C-5 was debugged and eventually released as "Sprites 2068" by a third party. Timex of Portugal also released "Basic 64" which supported 64, 80, 128 column text and BASIC graphics commands (CIRCLE, DRAW, etc.) in the 64 column mode, though written for the TC2048 and therefore must be run using a Spectrum emulator on the TS2068. A third party released OS64 on cartridge, an expansion to BASIC that allowed it to operate in the 64 column mode.

```
IOLOW   CRI80/11 version 13.36.14       7-May-84  15:9:0
Low level I/O module   IOLOW.SRC

A       Reserved        ATTBTT   E89A           ATTMSK   E89C           ATTRSP   5C8D           S       Reserved
C       Reserved        CALCAT   EA14           CALCA1   EA22           CALCPO   E9AP           CMRSET   3C00
CMTBL   E890            CONVPM   E990           CURPCS   E895           D       Reserved        DPADRS   E897
E       Reserved        ERRRET   E908           GCYCEX   E99A           GETBCM   E91P           GOODRE   E906
GRPMST  E93A            GRPST    EA3A           GRTBC    E892           GTCM1    E92P           GTCM2    E934
GTCM23  E946            GTCM3    E948           GTCM31   E94E           GTCM32   E969           GTCM4    E96E
GTCM5   E989            GTCM6    E99B           GTINDR   E898           H       Reserved        L       Reserved
LDATTR  EA2A            LDPOSN   E9DP           LINLEN   E894           LMBU     E9BC           M       Reserved
MASKB   E899            MASKBP   5C8E           PSW     Reserved        PSPLAG   9C91           SCRSZ    0318
SETSPR  E9DE            SP      Reserved        STPGSW   E9D7           UDG      5C78           UPDATT   E9E7
UPDAT1  EA06            UPDAT2   EA12           UPDPCB   E9A8           VIOMOD   5CC2           WACHRT   E903
WACH01  E8A3            WACH11   E9B8           WACH1E   E8C0           WACH13   E8C4           WACH14   E8DA
WACH15  E8D9            WACH2    E8DB           WACH3    E8F6           WACH5    E8EE           WACH7    E8FA
WRITE8  E890

    No errors detected
```

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.

VRl: U3-33 goes to VRl/Q5

Q4: Connect base to R55/R54

Solder dots on horizontal lines below keyboard:

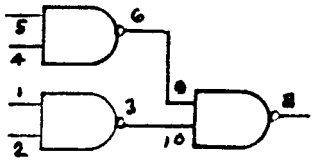
U12-4 to U3-65 ($\overline{WR}$)

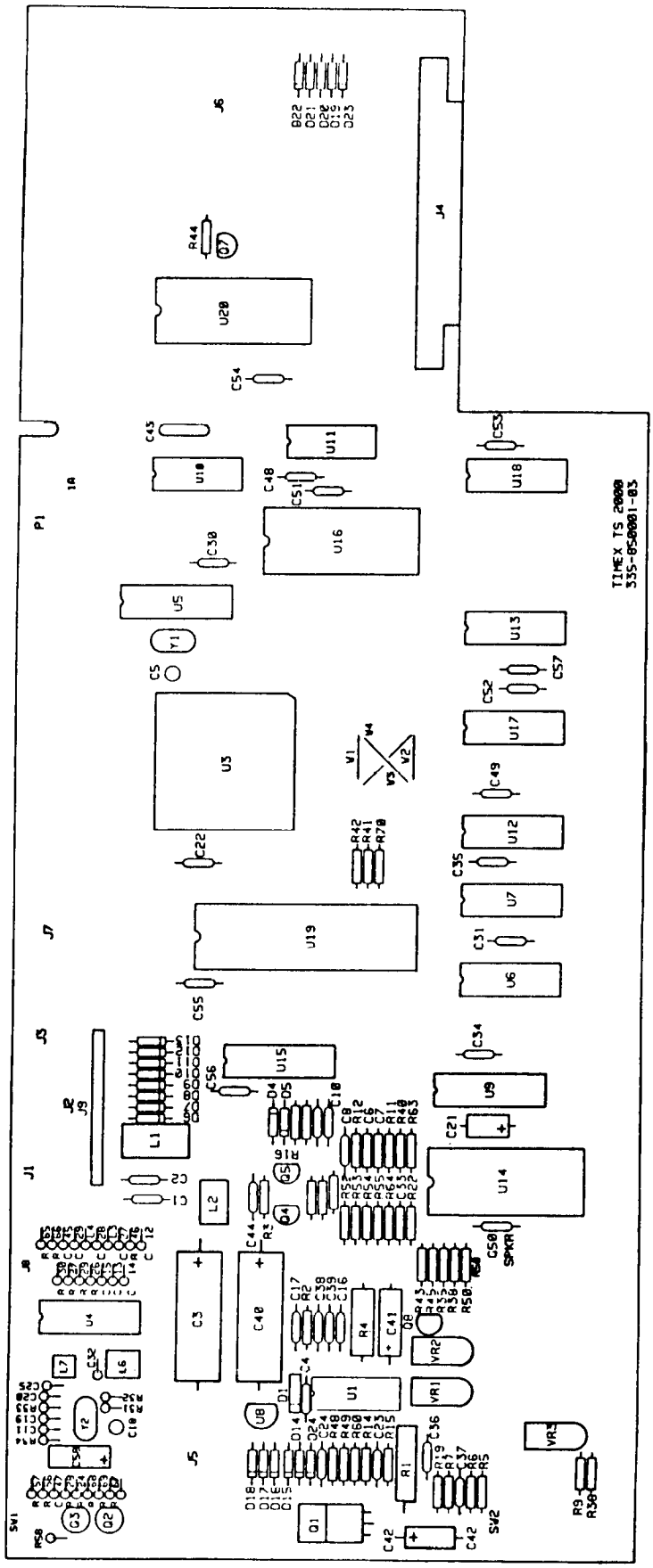
U12-5 to U3-66 ($\overline{MREQ}$)

u5: U5-2 to U3-38 (A7R not A7RB)

Pl: Pl-4B +15V (not -15V)

u21:





TS2068 PC BOARD COMPONENT LAYOUT

APPENDIX D

TS2068 PARTS LIST

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	COMMENTS
. . . (Fabrication and Artwork)			REV 3A
CAP. 0.1 uf, Ceramic, Axial TEMP Z5U	C2,7,9,16,24,30 31,34,35,37,39,43 44,48,49,50,51,52 53,54,55,56,57	23	-20 +80% or GMV
CAP. 0.01 uf, Ceramic,Axial	C11,12,14,33,61 62,68,69	8	-20 +80% or GMV TEMP Z5U
CAP. 0.001 uf, Ceramic, Axial	C8,45,46,47	4	-20 +80% or GMV TEMP Z5U
CAP. 0.047 uf, Ceramic, Axial	c10,15,74,75	4	-20 +80% or GMV TEMP Z5U
CAP. 20pf Ceramic Axial	C23	1	-20 +80% or GMV TEMP Z5U
CAP. 39pf Ceramic Axial	c20	1	NPO
CAP. 43pf Ceramic Axial	C19	1	NPO
CAP. 56pf Ceramic Axial	C25	1	NPO
CAP. 75pf Ceramic Axial	C32	1	NPO
CAP.120pf Ceramic Disc	C59,63,64,65,72 73	6	-20 +80% or GMV TEMP Z5U
CAP.470uf, 25V AL Electro-lytic Axial	c3	1	
CAP. 1 uf, 16V MIN AL Electro-lytic Axial	c21	1	
CAP. 47 uf, 16V MIN AL Elec-trolytic Axial or Radial	c41	1	
CAP. 1000 uf, 12V MIN AL Electrolytic Axial	c40	1	LOW ESR
CAP. 1000 pf, 50V MIN FILM MYLAR	C36	1	+/- 20%
CAP. 100 uf, 10V MIN AL Elec-trolytic Axial	C58,67	2	
CAP. 6-50 pf, TRIMMER	C5,18	2	NPO
CAP. 0.47 uf Ceramic Axial	C60	1	-20 +80% or GMV TEMP Z5U
CAP. 33 uf TANTALUM	c71	1	+/- 20%

APPENDIX D

TS2068 PARTS LIST
(continued)

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	-20 +80% COMMENTS
68 pf Ceramic Axial	c/o	1	TEMP Z5U 20 80% or GMV
CAP. 24 pf Ceramic Axial	c29,27	2	-20 or GMV TEMP Z5U
CAP. 47 pf Ceramic Axial	C28	1	-20 +80 or GMV TEMP Z5U
RES. 300 OHM, 1/4W, +/-5%, CF	R23	1	
RES. 200 OHM, 1/4W, +/-5%, CF	R19,50,54,55	4	
RES. 100 OHM, 1/4W, +/-5%, CF	R58	1	
RES. 240 OHM, 1/4W, +/-5%, CF	R24,28,56,57	4	
RES. 68 OHM, 1/4W, +/-5%, CF	R2	1	
RES. 680 OHM, 1/4W, +/-5%, CF	R13 68	2	
RES. 390 OHM, 1/4W, +/-5%, CF	R74'	1	
RES. 1K OHM, 1/4W, +/-5%, CF	R11,33,34,35,36 38,42,62	8	
RES.1.5K OHM, 1/4W, +/-5%, CF	R41	1	
RES.1.8K OHM, 1/4W, +/-5%, CF	R29,30	2	
RES. 620 OHM, 1/4W, +/-5%, CF	R52	1	
RES. 2K OHM, 1/4W, +/-5%, CF	R22	1	
RES. 3K OHM, 1/4W, +/-5%, CF	R32	1	
RES.2.2K OHM, 1/4W, +/-5%, CF	R61	1	
RES. 110 OHM, 1/4W, +/-5%, CF	R53	1	
RES. 510 OHM, 1/4W, +/-5%, CF	R69	1	
RES.5.1K OHM, 1/4W, +/-5%, CF	R31	1	
RES. 10K OHM, 1/4W, +/-5%, CF	R16,40,60,70	4	
RES. 13K OHM, 1/4W, +/-5%, CF	R26,27	2	
RES. 20K OHM, 1/4W, +/-5%, CF	R44,45	2	
RES 62K OHM, 1/4W, +/-5%, CF	R9,73	2	
RES.lOOK OHM, 1/4W, +/-5%, CF	R15,49	2	
RES.220K OHM, 1/4W, +/-5%, CF	R43	1	
RES. 75 OHM, 1/4W, +/-5%, CF	R46,67	2	
RES.l.lOK OHM 1/4W, +/-l%, MF	R6	1	
RES.3.32K OHM 1/4W, +/-l%, MF	R5	1	
RES. 10K OHM. VARIABLE. LINEAR	VR1.2,3	3	
RES. 330 OHM; 0.5W, +/-5%, CF	R4	1	
RES. 56 OHM, 1/4W, +/-5%, CF	R65,71	2	
RES. 0.110 OHM, 3W, +/-5%, Wire Wound	Rl	1	
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)

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	COMMENTS
DIODE 1N4148	R4,5,6,7,8,9,10 11,12,13,14,15,16 17,18,19,20,21,22 23,24,25,26,27,28	25	
DIODE, Schottky 1N5821 or equivalent	CR1		
IC, UA 78S40 NPC, Switching Regulator	Ul	1	
IC, SCLD	u3	1	
IC, LM1889N, Video Modulator	u4	1	
IC, 74LS244N	u5	1	
IC, TMS4416-15 (150NS) MOS Dynamic RAM	U6,7	2	
IC, UA 78L12 Regulator	U8		
IC, 74LS245	u9,15	2	
IC,74LS157N	U10,11	2	
IC,TMS4416-20 (200NS) MOS Dynamic RAM	U12,13,17,18	4	
IC,AY-3-8912, Sound Gen.and I/O Port	u14	1	
IC,23128 Mask ROM (16K X 8)	U16	1	
IC, CPU Z80A	u19	1	
IC, 2364 Mask ROM (8K X 8)	u20	1	
IC, 74LSOO	u21	1	
TRAN. PNP D43Cl	Ql	1	
TRAN. PNP 2N2907	Q3	1	
TRAN. PNP 2N3904	Q7,8	2	
TRAN. PNP 2N2222	Q5,4,2	3	

APPENDIX D

TS2068 PARTS LIST
(continued)

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	COMMENTS
EMI Filter(Bifiler) 2.2mh	Ll	1	
Inductor 230 uh	L2	1	
Inductor .33uh Axial	L3,4	2	
Inductor .12uh	L6,7	2	
Crystal Oscillator 14.112 MHz	Yl	1	
Crystal Oscillator 3.579545 MHz	Y2	1	
Switch SPDT, Rocker	SW2	1	
Switch Channel Select, SPDT Slide	SW1	1	
Video Jack Insulation Pad		1	Under J7
Jack, Right Angle RCA Video Jack	J7	1	Monitor
Jack, Mini Phone, EAR & MIC	J2,3	2	Tape
Jack, COAX, DC Power, 2 l/2 MM Pin	Jl	1	
Jack, Phono	J8	1	Assembled to Shield, R.F.
Connector,Cartridge 2 X 18 Pin 0.1" Space	J4	1	Key between Contact 4&6
Connector,Flex Cable 14 Pin	J9	1	Keyboard
Connector,Joystick 9-Pin Male (D Type)	J5,6	2	Joysticks
Shield, R.F. Button		1	
Shield, R.F. Top		1	
Heat Sink	HSl	1	
Heat Sink Insulation Pad			

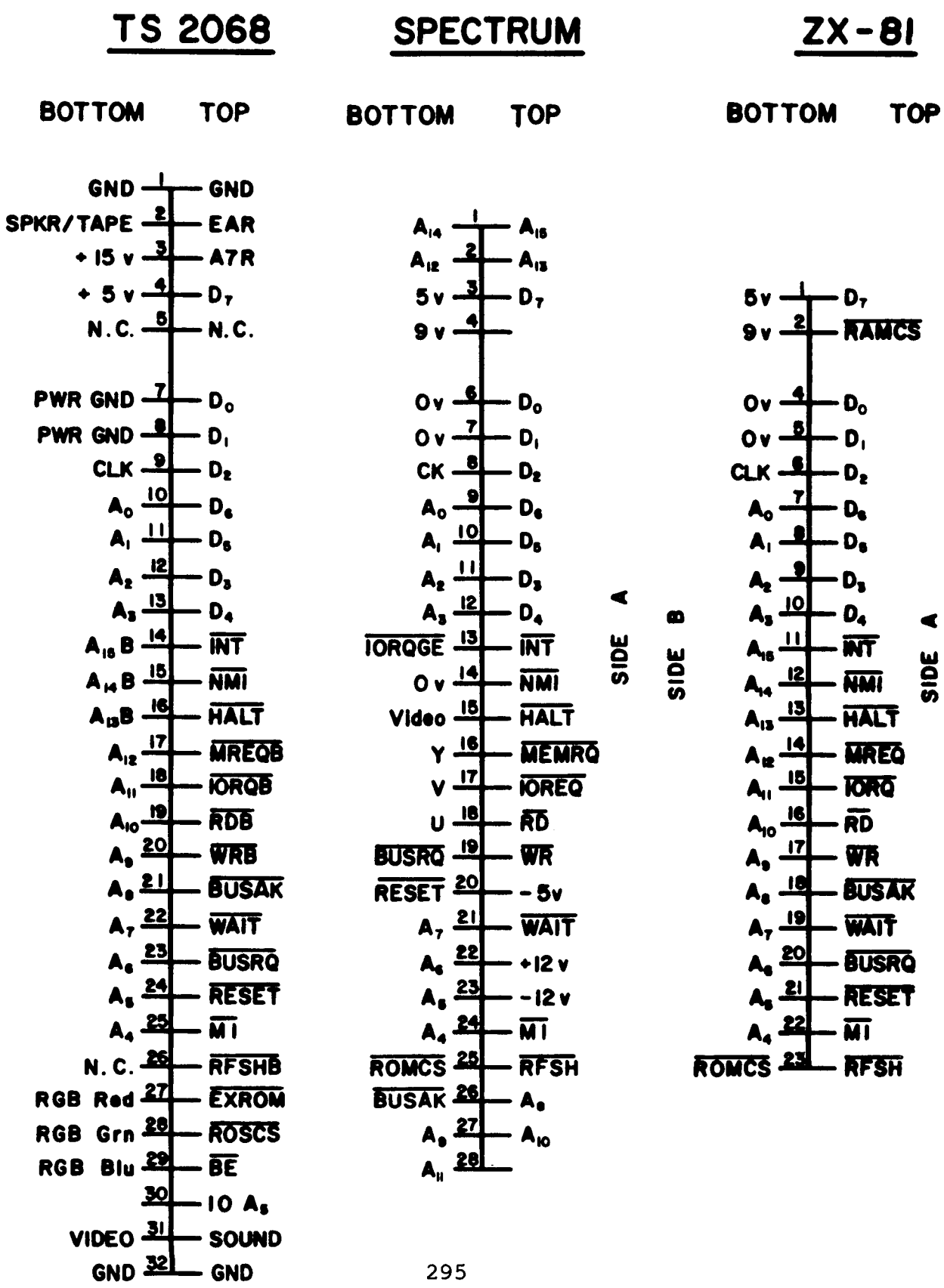
APPENDIX D

TS2068 PARTS LIST
(continued)

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	COMMENTS
Socket, XC, 28 Pin		2	
Socket, IC, 40 Pin		1	
Speaker, 45 OHM, Mylar Cone		1	
Jumper Wire	Wl, 2, 50	3	
Ferrite Bead	L5,8	2	
PC Board Assembly,Daughter		1	

APPENDIX E

Expansion Buss Comparison of
TS2068, Sinclair Spectrum and ZX81

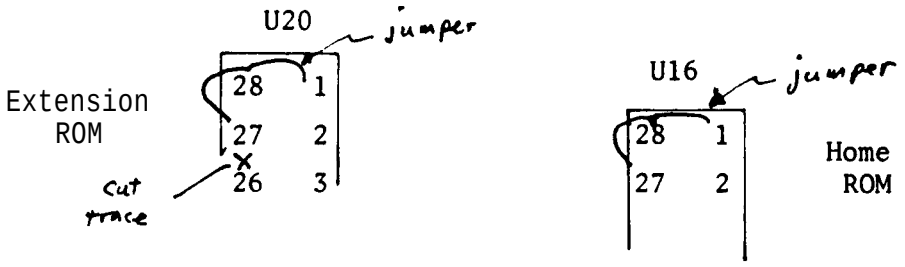


APPENDIX F

August 1985
Bob Orrfelt

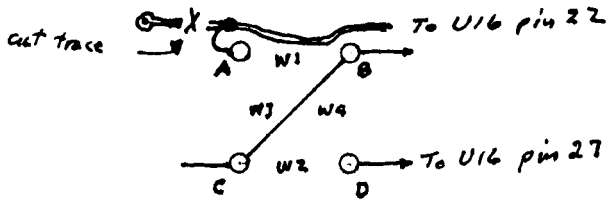
TS2068 MODIFICATIONS FOR EPROMS

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-comnonent 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 Wl 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 $\overline{\text{MREQ}}$ to U16 pin 22.
6. Add a jumper from hole C to hole B. This connects $\overline{\text{ROMCS}}$ 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.
006D 2801       JR    Z,0070H

DELETE delay timing.
0351 010100     LD    BC,0001H
0354 0B         DEC   BC
0355 79         LD    A,C
0356 B0         OR    B
0357 20FB       JR    NZ,0354H
0359 F1         POP   AF
035A 18D2       JR    032EH
```

Optional turn an message.
(Last. character add 80H)

```
1118 Property of Bob
1128 Orrfelt .........
1138 ..................
1138 ..
```

```
INT -65536 etc. errors.
33F1 F5         PUSH AF
33F2 3C         INC  A
33F3 B3         OR   E
33F4 B2         OR   D
33F5 C2E435     JP   NZ,35E4H
33F8 C3EF35     JP   35EFH

35E2 181A       JR   35FEH
35E4. F1        POP  AF
35E5 77         LD   (HL),A
35E6 23         INC
35E7 73         LD   (HL)
35E8 23         INC  HL
35E9 72         LD   (HL),D
35EA 2B         DEC  HL
35EB 2B         DEC  HL
35EC 2B         DEC  HL
35ED Dl         POP  DE
35EE C9         RET

35EF   F1       POP  AF
35F0 2B         DEC
35F1 3691       LD   (HL),91H
35F3 23         INC
35F4 3680       LD   (HL),80H
35F6 3C         INC  A
35F7 18ED       JR   35E6H

35F9            blanks
35FD
```

```
USR chunk selection.
389F E660       AND  60H
38A1 281B       JR   Z,38BEH
38A3 D640       SUB  A,40H
38A5 FAB738     JP   M,38B7H
```

```
Fix for Oliger EPROM programer.
(see May 85 Syncware, page 14)
002B 84         DB   84H
002C 87         DB   87H
002D 8B         DB   8BH
002E 8D         DB   8DH
002F 92         DB   92H
```

```
     for EPROM programmer.
     D9         EXX
37B9 212B00     LD   002BH
37BC 85         ADD  A,L
37BD 6F         LD   L,A
37BE 6E         LD   (HL)
37BF 2636       LD   36H

37C1D9          EXX
37C2AF          XORA
37C3C9
37C4            NOP
```

NOTES

RESTRICTED INFORMATION

DWG NO

"USE OF THE TECHNICAL INFORMATION CONTAINED HEREIN IS AT USER'S SOLE RISK. WHILE THE INFORMATION WAS BELIEVED TO BE CORRECT AT THE TIME IT WAS PREPARED, TIMEX COMPUTER CORPORATION OR ITS AFFILIATES, 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."

NOTES: (UNLESS OTHERWISE SPECIFIED)

1) ALL RESISTORS ARE 1/4 W, 5%, RATED IN OHMS.
2) ALL CAPACITORS ARE RATED IN MICROFARADS.
3) FOR D28128C, INSERT W1 & W2.
   FOR INSERT W3 & W4 (NOT USED)

CHANGES

REVISION NO	DESCRIPTION	DATE	APPROVED

REFERENCE DESIGNATIONS	
LAST USED	NOT USED
C75	C1, 13, 26
CR28	CR2, 3
L8	
Q8	Q6
U21	U2
VR3	
R75	R8, 10, 17, 18, 20, 21, 25, 37 39, 47, 47, 51, 59
W4	
SW2	
Y2	

MATERIAL		DO NOT SCALE DRAWING	
SPEC		CHK	DWN N PARSONS
SIZE & TOL		REF	DATE FEB 84
TEMPER		FIRST USED ON M2068	SCALE NONE
PHY SPEC			SUPERSEDES
			SUPERSEDED BY

TIMEX
COMPUTER CORPORATION
WATERBURY CONN 06720

NAME TS2000 SYSTEM SCHEMATIC DIAGRAM

DWG NO 335-870500

BRUNING 44131 22870 3