TECHNICAL MANUAL

TIMEX SINCLAIR 2068 PERSONAL COLOR COMPUTER

Published by The Time Designs Magazine Co.

TIMEX SINCLAIR 2068
PERSONAL COLOR COMPUTER

TECHNICAL REFERENCE MANUAL

Prepared by

V. C. Corcoran
and
M. H. Branigin

TIMEX COMPUTER CORPORATION
Waterbury CT 06720

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.

TABLE 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
RAM - 48K Random Access Memory
ROM - 24K System Read-Only Memory
(16K plus 8K Extension)

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

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

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

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

FIGURE 1.1-1

TS 2068 SYSTEM BLOCK DIAGRAM

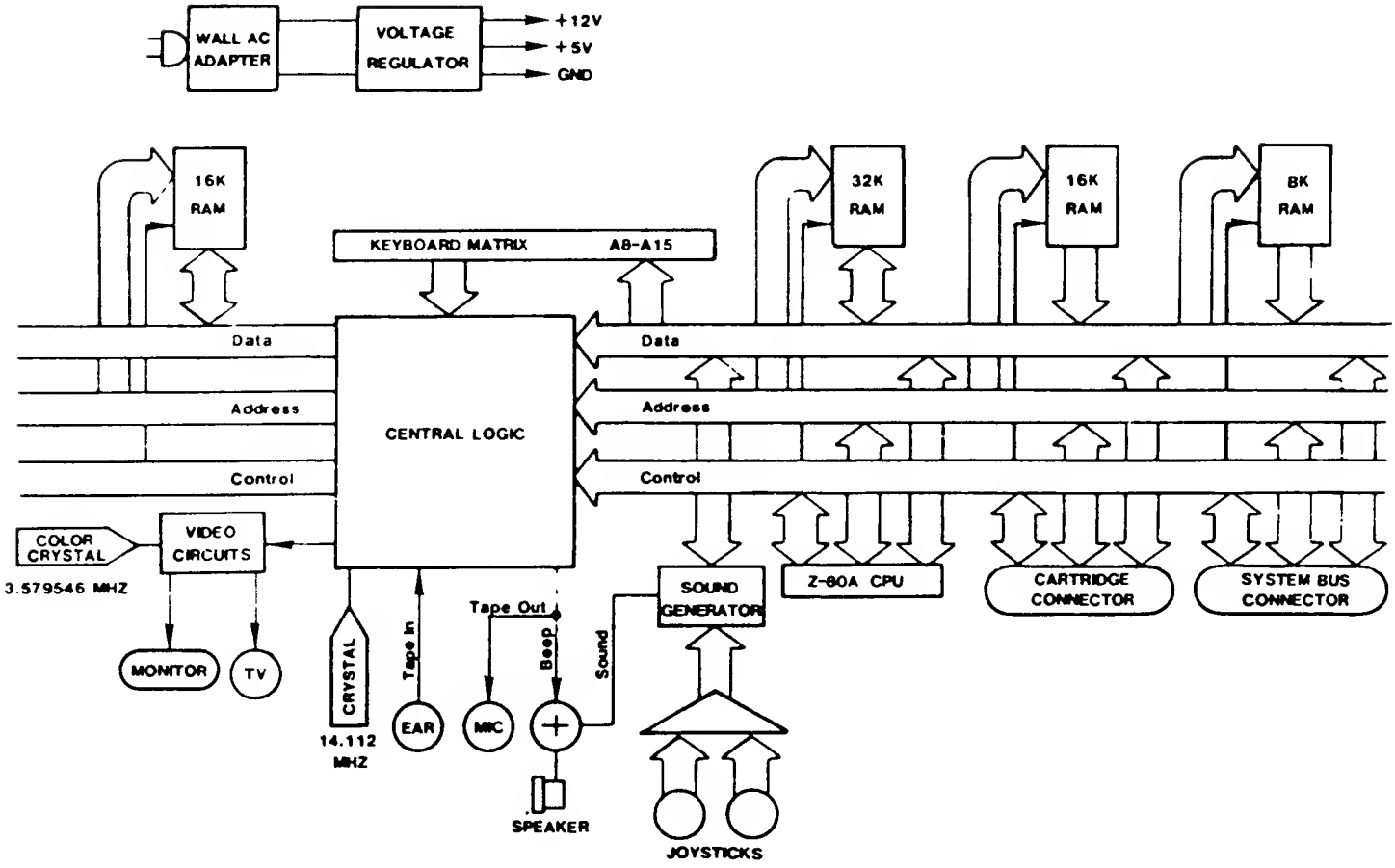
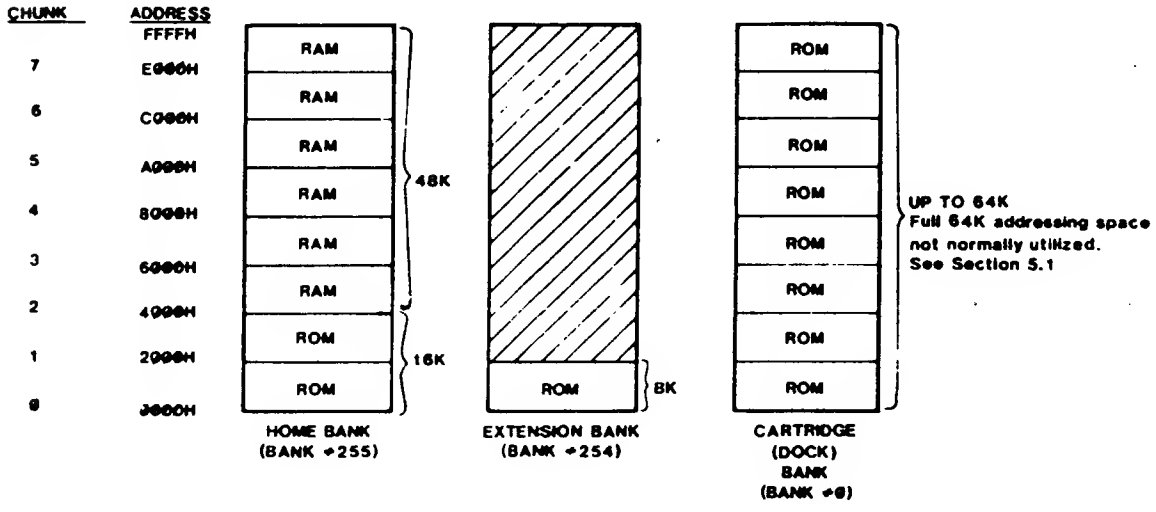


FIGURE 1.1-2

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

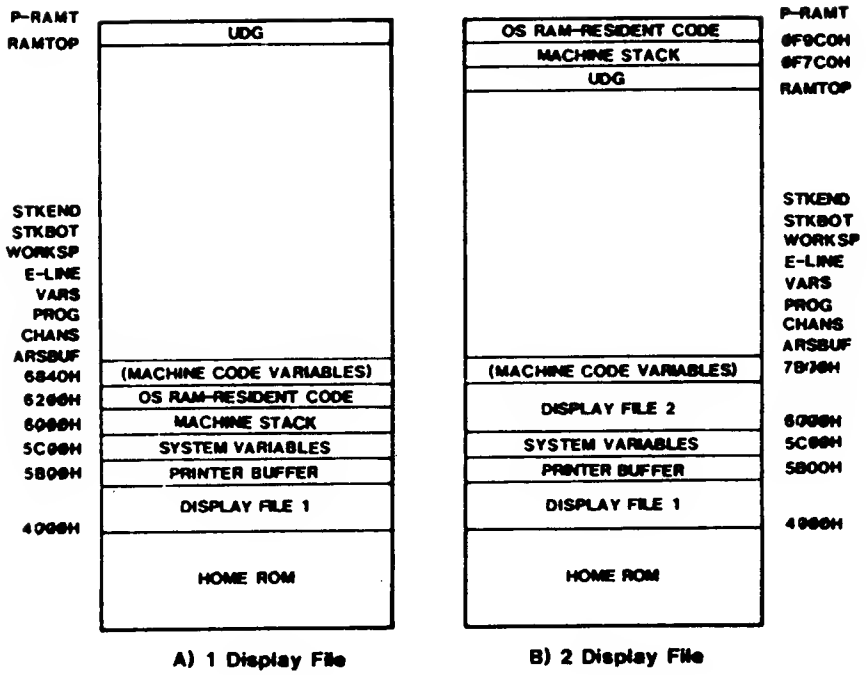
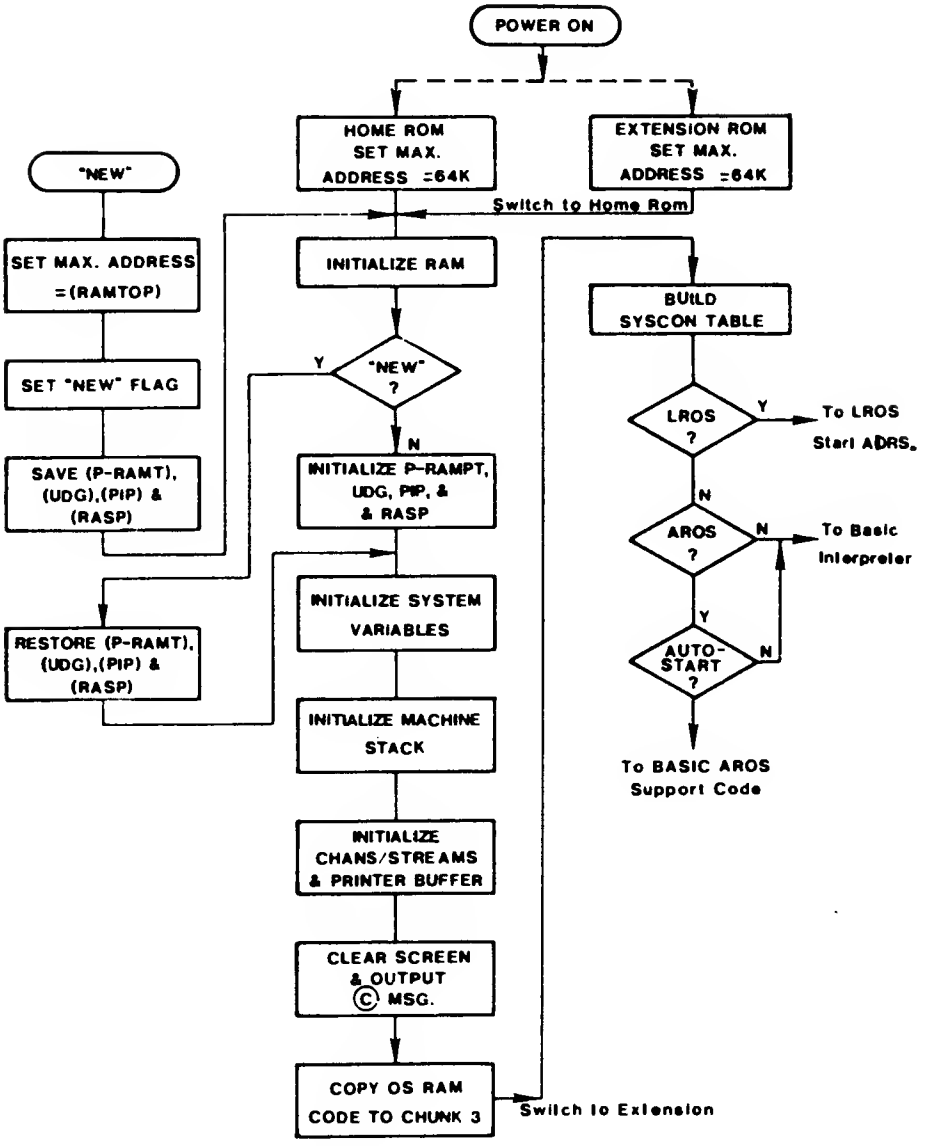


FIGURE 1.1-4

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

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

2.0 HARDWARE GUIDE

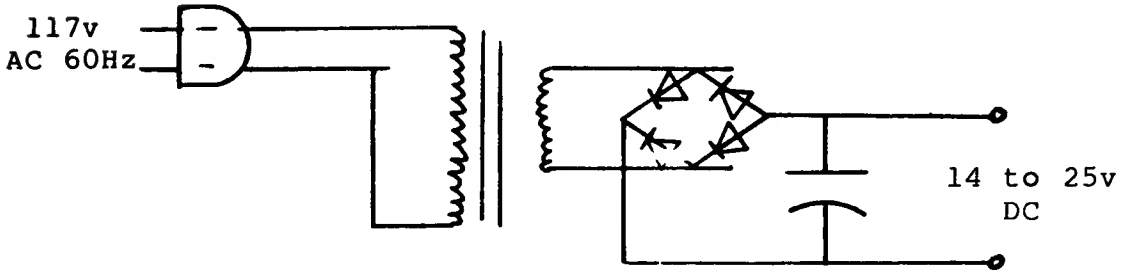
2.1 Description of Major Hardware Functions

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

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
2.1.3.5	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

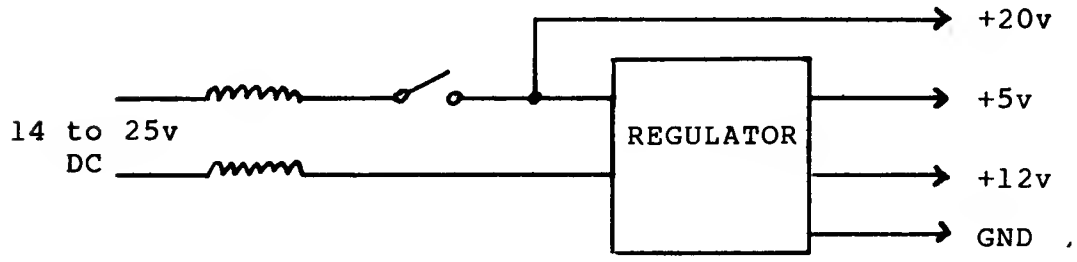
The AC Adapter transforms 117V AC (Nominal) to filtered DC via a step down transformer, full-wave bridge rectifier, and filter capacitor to supply from 14 to 25 volts at 1 amp over the AC voltage variation range of 105 to 130 V AC. Transformer isolation exceeds 1500 volts.



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

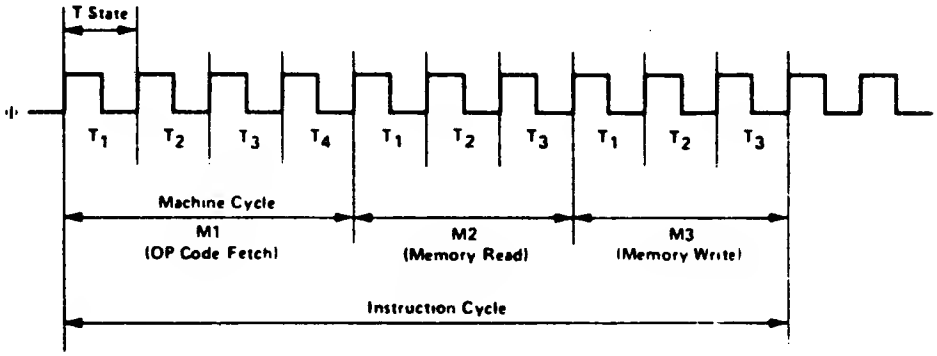
The Z-80 CPU executes instructions by proceeding through a sequence of operations that include:

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, D0 - D7, constitute an 8-bit bi-directional, high-active, tri-state data bus used for data exchanges with memory and I/O devices.

2.1.3.3 Control Bus

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

2.1.3.4 Op Code Fetch

The timing during an M1 cycle (OP Code Fetch) is shown in Figure 2.1.3-2. At the beginning of the M1 cycle the PC (Program Counter) is placed onto the address bus, then one-half clock time later the $\overline{\text{MREQ}}$ signal goes active indicating that the memory address is stable. The $\overline{\text{RD}}$ 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 $\overline{\text{RD}}$ and $\overline{\text{MREQ}}$ 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 $\overline{\text{RFSH}}$ signal indicating a refresh read is to begin when $\overline{\text{MREQ}}$ 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 $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$ signals used as in the fetch cycle. During a write cycle the $\overline{\text{WR}}$ 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 $\overline{\text{WR}}$ signal goes active. Figure 2.1.3-3 illustrates.

FIGURE 2.1.3-2

INSTRUCTION OP CODE FETCH

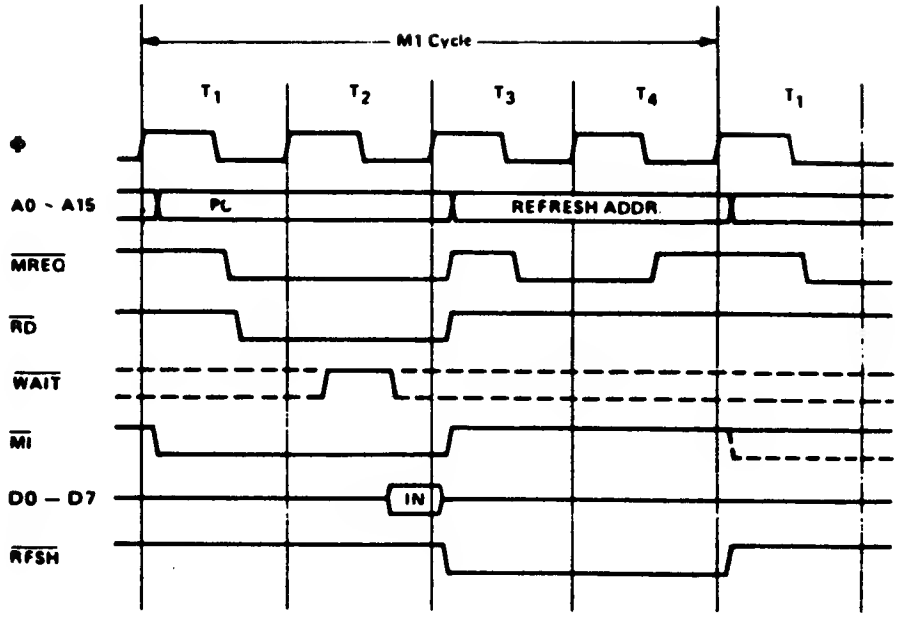
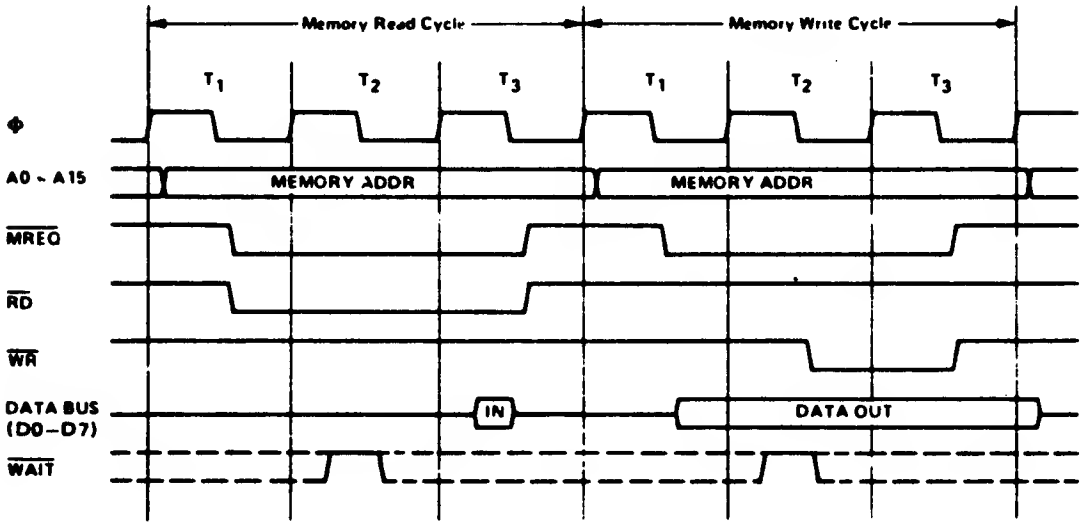


FIGURE 2.1.3-3

MEMORY READ OR WRITE CYCLES

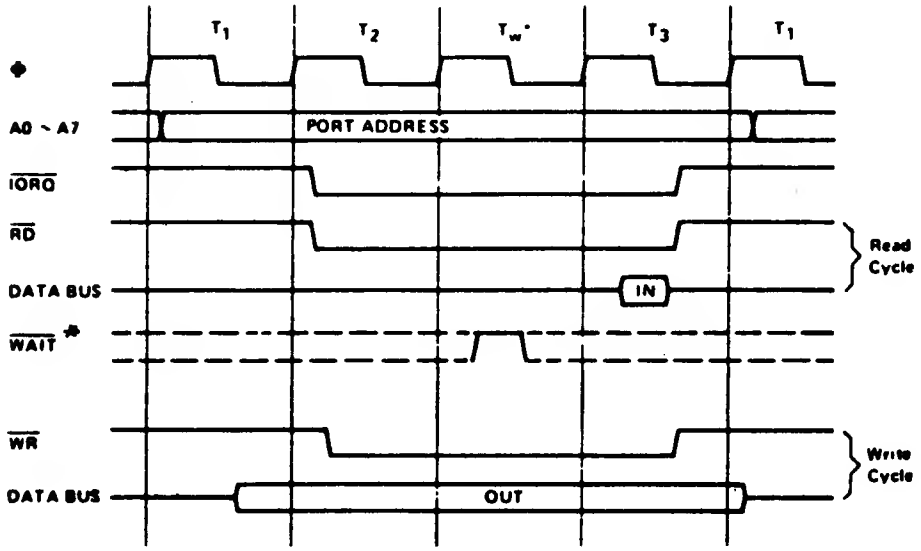


2.1.3.6 I/O READ/WRITE

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 $\overline{Wait}$ 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



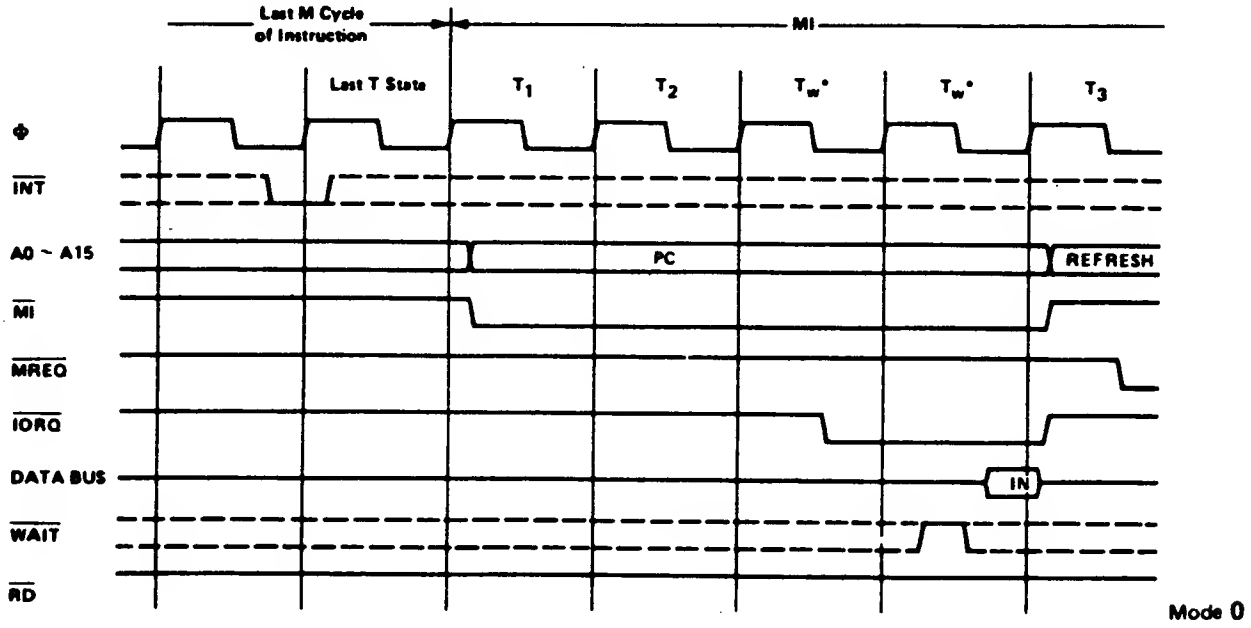
*Inserted by Z80 CPU

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 M1 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 2-byte boundary address, with the 8-bit I Register contents to form a 16-bit pointer to a memory table entry containing the 16-bit service routine address - the first byte in the table

being the low order portion of the address. Once the interrupting device supplies the lower portion of the pointer (for concatenation), the CPU automatically pushes the PC onto the stack, obtains the starting address from the table, and does a jump to that address. 19 clock periods are required to complete this sequence.

2.1.3.8 Non-Maskable Interruption (NMI)

A pulse on the $\overline{\text{NMI}}$ 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.
	$\overline{RFSH}$	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 $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories. A7 is a logic zero and the upper 8 bits of the Address Bus contain the contents of the I Register.

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.

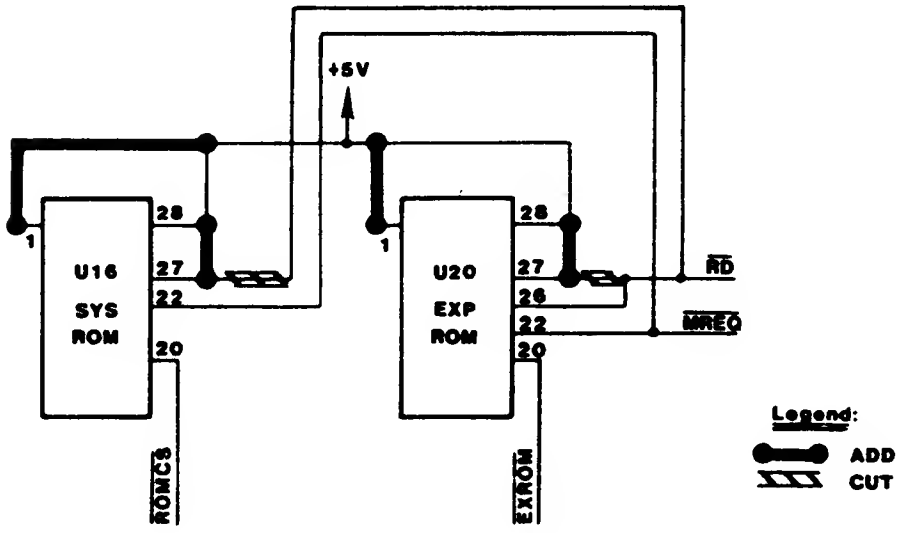
TABLE 2-1

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 go to a high impedance state permitting other devices to control these buses. The CPU sets these buses to a high impedance state at the termination of the current Machine cycle.
	$\overline{\text{BUSAK}}$	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 $\overline{\text{BUSRQ}}$.

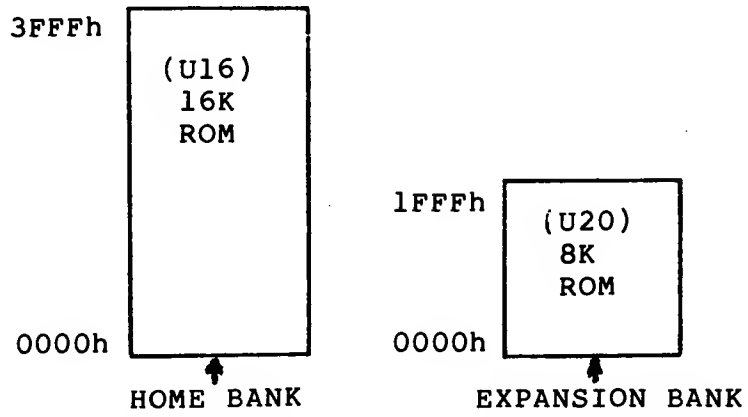
Figure 2.1.4-1

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 0F5H (Address) and 0F6H (Data). The basic registers in the PSG which produce the programmed sounds include:

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

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

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

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

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

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

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

2.1.6.1 Tone Generator Control (Registers R0-R5)

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

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

Note also that due to the design technique used in the Tone Period countdown, the lowest period value is 000000000001 (divide by 1) and the highest period value is 111111111111 (divide by 4095).

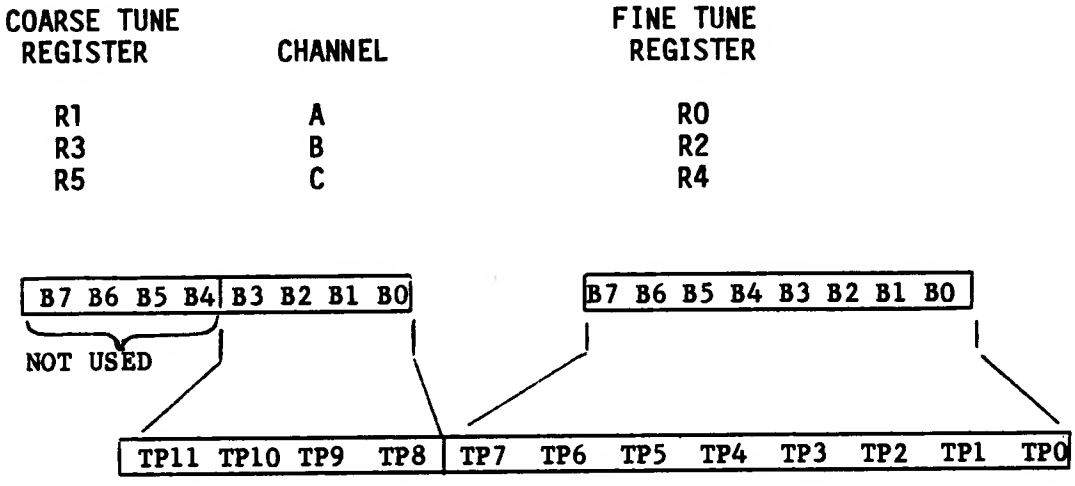
FIGURE 2.1.6-1

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	R12	Ch.C Amplitude	////////////////			M	L3	L2	L1	L0
R11	RB	R13	Envelope	8 Bit Fine Tune E							
R12	RC	R14	Period	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



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:

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

Where:

- fT = Desired tone frequency
- $fCLOCK$ = Input clock frequency
- TP_{10} = Decimal equivalent of the Tone Period bits TP11 to TP0
- CT_{10} = Decimal equivalent of the Coarse Tune register bits B3 to B0 (TP11 to TP8)
- FT_{10} = Decimal equivalent of the Fine Tune register bits B7 to B0 (TP7 to TP0)

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

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

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:

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

Example 1: $fT = 1$ kHZ $\qquad fCLOCK = 1.76475$ MHz

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

Substituting this result into equation (b):

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

resulting in:

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

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

Example 2: $fT = 100$ Hz $\qquad fCLOCK = 1.76475$ 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:

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

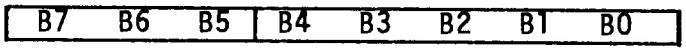
$$FT_{10} = 79_{10} = 01001111_{2}\ (B7\text{-}B0)$$

2.1.6.2 Noise Generator Control (Register R6)

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

FIGURE 2.1.6-3

NOISE PERIOD REGISTER R6



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 $NP_{10} = 31_{10}$) 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 B0 of R7.

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

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

FIGURE 2.1.6-4

MIXER CONTROL - I/O ENABLE REGISTER R7

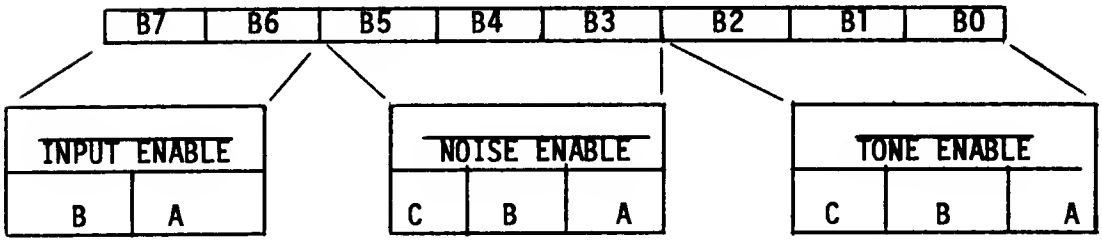


TABLE 2.1.6-1

I/O ENABLE TRUTH TABLE

NOISE ENABLE TRUTH TABLE				TONE ENABLE TRUTH TABLE			
R7 BITS B5 B4 B3			Noise Enabled on Channel	R7 BITS B2 B1 B0			Tone Enabled 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	I/O Port Status
B6	IOA
0	Input
1	Output

NOTE

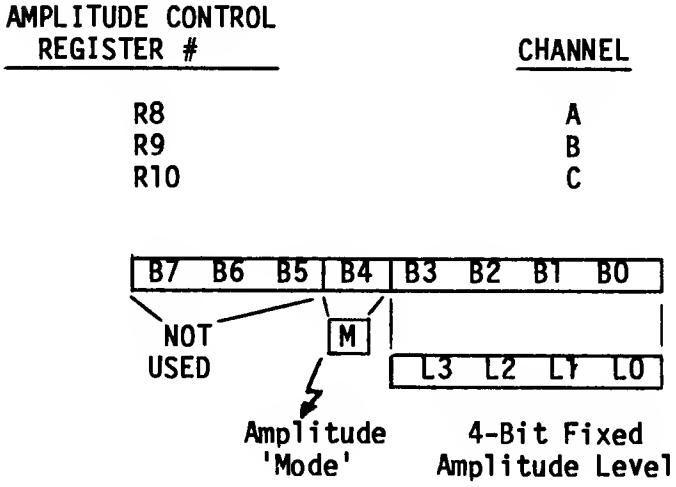
Disabling noise and tone does not turn off a channel. Turning a channel off can only be accomplished by writing all zeroes into the corresponding Amplitude Control register, R8, R9 or R10 (refer to Paragraph 2.1.6.4).

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)								

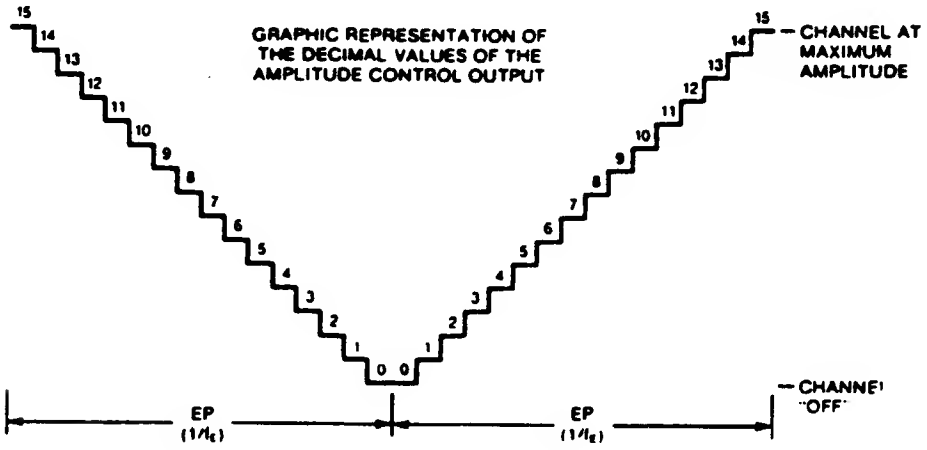
*The all zeros code is used to turn a channel "off".

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)

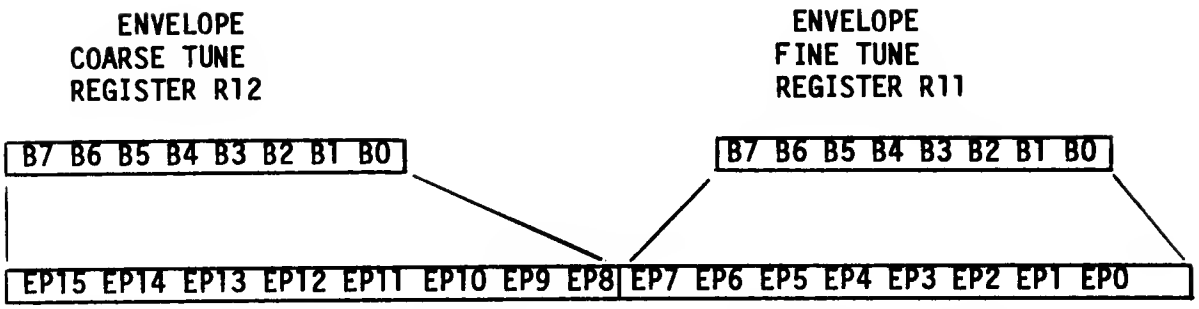
To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG; first, it is possible to vary the frequency of the envelope using registers R11 and R12; and second, the relative shape and cycle pattern of the envelope can be varied using register 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 programmed 16-bit Envelope Period value. This 16-bit value is obtained in the PSG by combining the contents of the Envelope Coarse and Fine Tune registers, as illustrated by Figure 2.1.6-8.

FIGURE 2.1.6-8

16-BIT ENVELOPE PERIOD (EP) TO
ENVELOPE GENERATOR



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

Note also that, as with the Tone Period, the lowest period value is 0000000000000001 (divide by 1); the highest period value is 1111111111111111 (divide by 65,535).
$_{2}$ $_{10}$

The envelope frequency equations are:

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

(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 $_{10}$) 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):

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

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

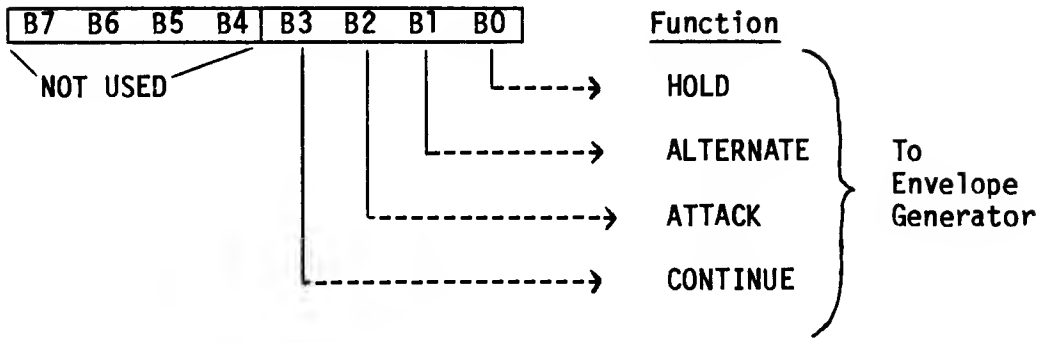
$FT_{10} = 219 = 11011011_{2}$ (B7-B0)

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

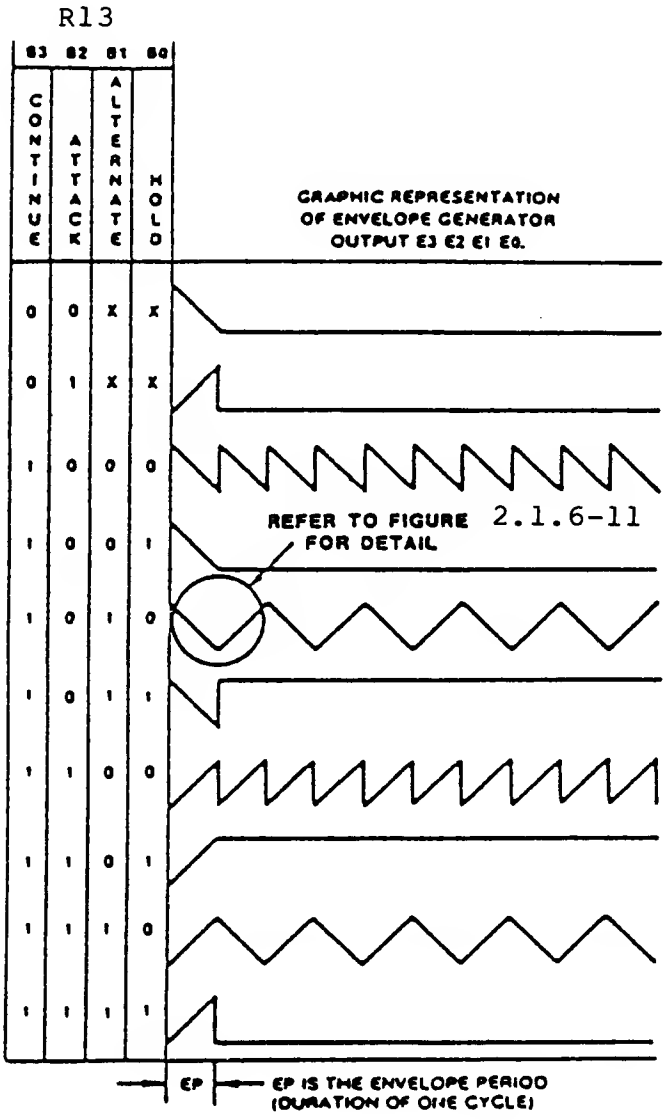
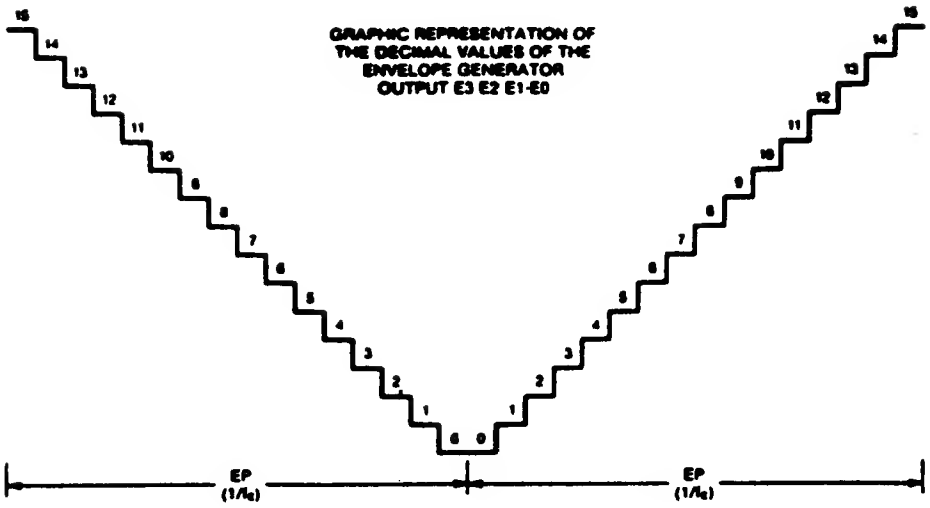


FIGURE 2.1.6-11

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-1. In this state, a read of port F6H with address bit A8 high, causes actions as follows:

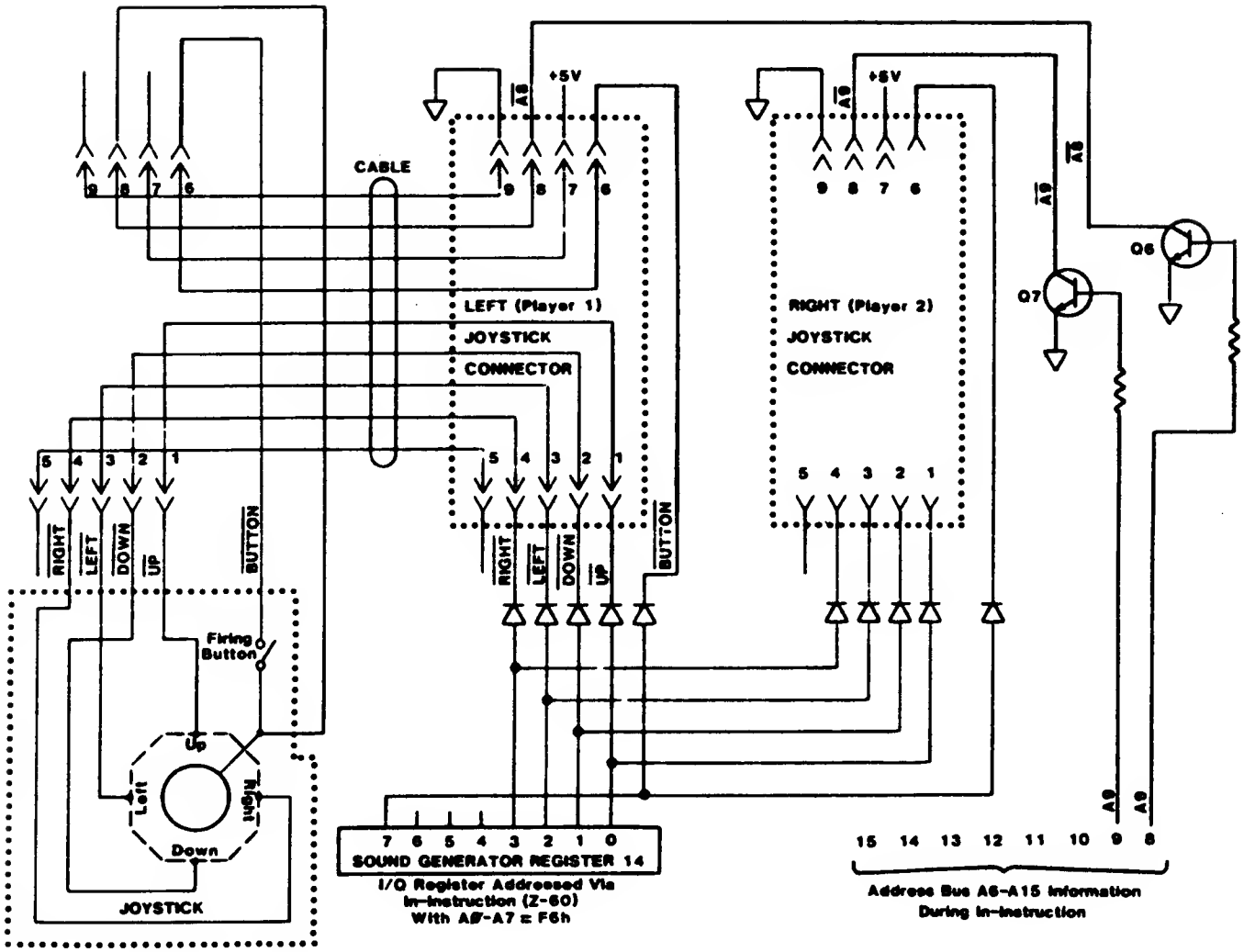
(1) Address A8 high turns on transistor Q8
(2) Q8 drives cable pin-8 low
(3) The movable center stick of the joystick in contact with the up-contact results in a conductive path from cable pin-8 to cable pin-1.
(4) Pin-1 low results in a 0 in bit position 0 of the I/O register via the isolation diode.

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

Note that +5 volts and ground are available on the connector so +5V logic could be attached to the joystick port.

FIGURE 2.1.7-1

JOYSTICK PORT OPERATION



2.1.8 Control Logic

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

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

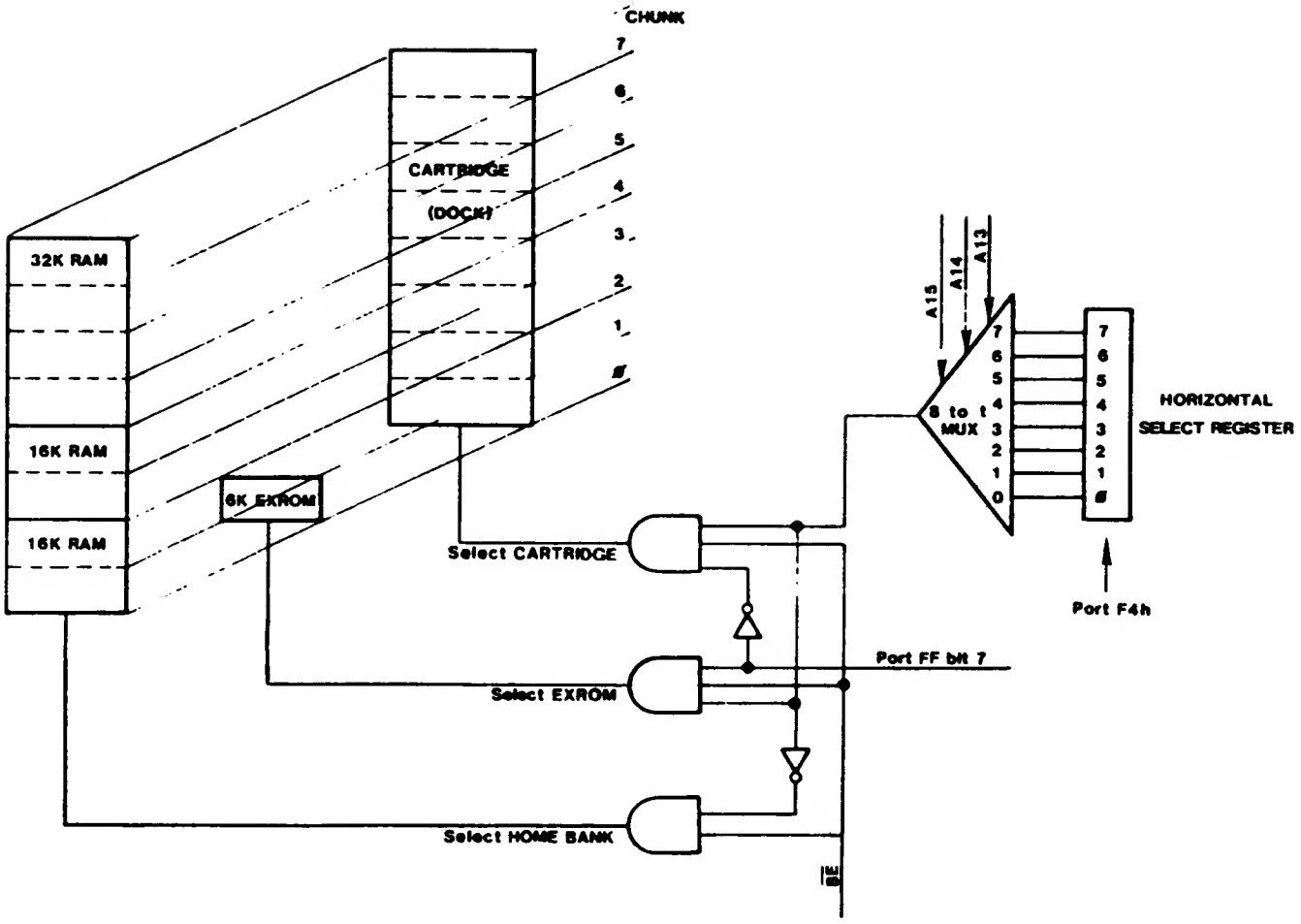
(2) SELECT EXROM of Figure 2.1.8-1 involves activating $\overline{EXROM}$ to its low active state

(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



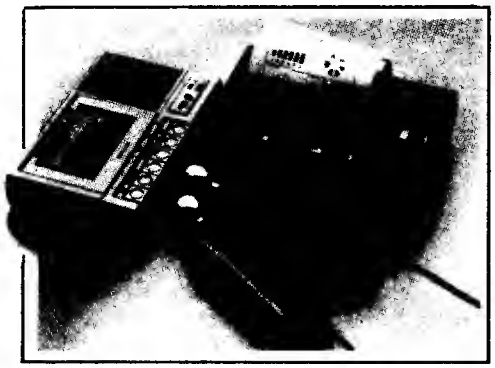


TABLE 2.1.8-1

SCLD I/O PIN FUNCTION DEFINITIONS

SYMBOL	NAME	DIRECTION OF SCLD IN/OUT	FUNCTION
A0-A7 A13-A15	Address Bus	In	Address Bus lines Input from Z80A
D0-D7	Data Bus	In/Out	Data Bus inputs/outputs from/to Z80A through U9-74LS245 or inputs from display RAM (16K) - U6 and U7
KB0-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)
MA0-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
0CPU	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
$\overline{RAS1}$	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 #1	Out	To activate column address strobe for display memory only (2nd 16K) during memory read/write and display read
$\overline{CAS2}$	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 & U11 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 (brightness) 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 2.1.8-1

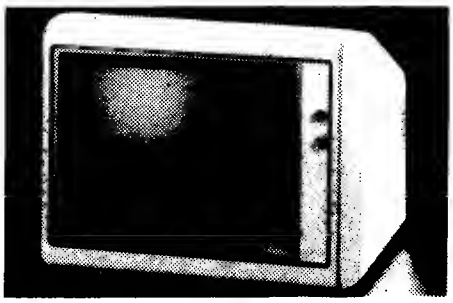
SCLD I/O PIN FUNCTION DEFINITIONS
(continued)

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 @1.764 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
BC1	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
OSC Out	Oscillator 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 (J1 - 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.

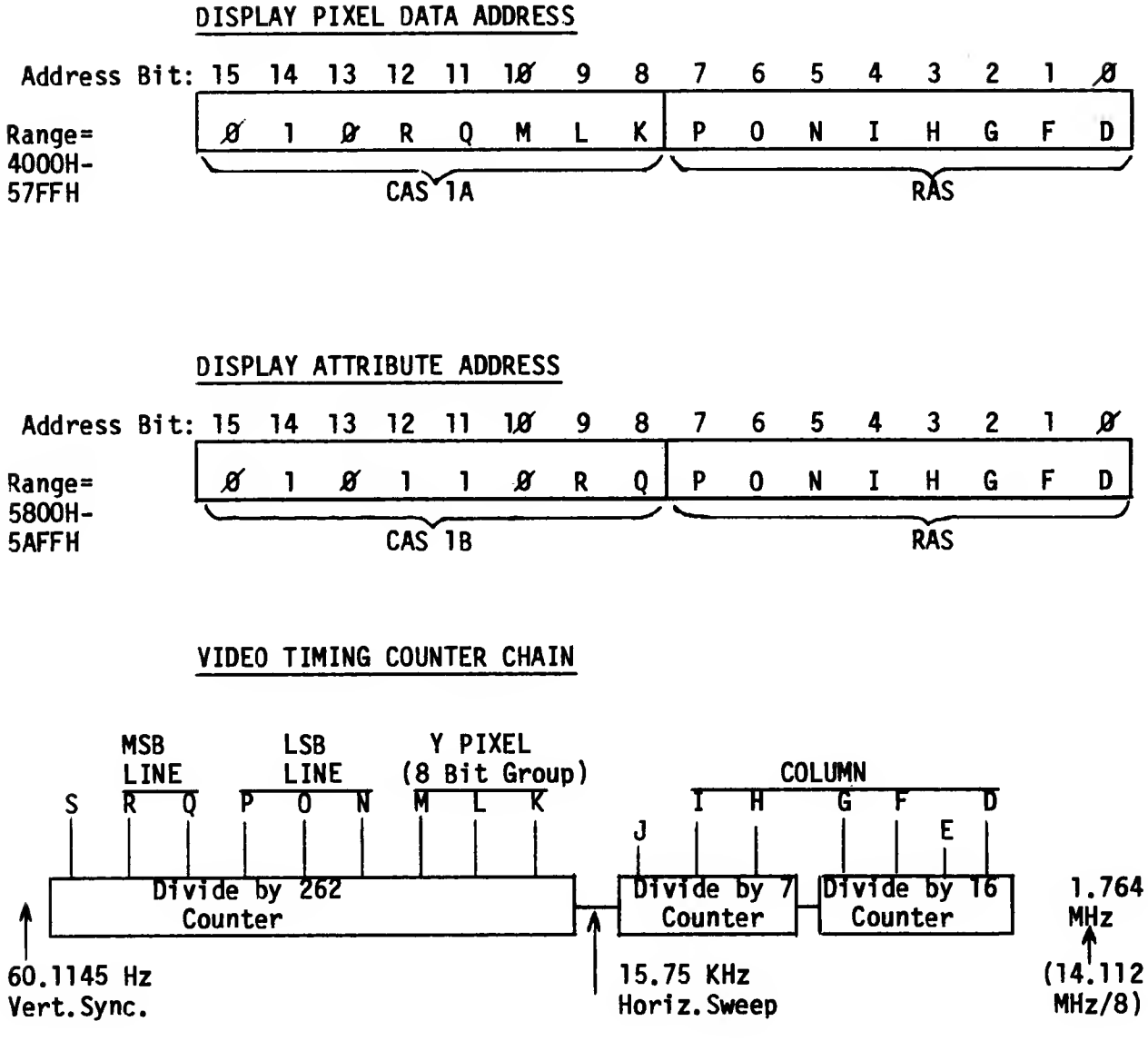
During each horizontal scan the video display processor accesses, in the standard video mode, 32 bytes of pixel data plus 32 bytes of attributes by 32 memory accesses reading 2 bytes per access in RAM page mode, i.e. the low order address bits are provided to the RAM once via RAS activation, then the 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 Fig. 2.1.8-2.)

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

The address information provided to the RAM's during RAS and CAS times is as shown in Figure 2.1.8-2. This address generation logic explains the non-sequential nature of the video display as described in Section 2.1.10.

FIGURE 2.1.8-2

VIDEO DISPLAY PROCESSOR RAM ADDRESS GENERATION
(Normal Video Mode)



2.1.8.4 Interruption Generation (17 ms)

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

2.1.9 Keyboard

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

Each switch closure connects one of the eight high order address lines (by going low through a diode) to one of the five input lines to the SCLD (KB0 through KB4).

Scanning is by software algorithm as described in Section 4.1.1. During the IN instruction, address bits A0-A7=FEH select the Keyboard I/O port while bits A8-A15 select the particular 5 keys to be sampled during the particular IN instruction execution. For example, an IN instruction directed at the keyboard I/O port with address bit A8 low and A9-A15 high will supply 0's on KB0, KB1, KB2, KB3, and/or KB4 if the CAP SHIFT, Z, X, C, and/or V keys are respectively depressed.

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

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

2.1.10 16K Video Display RAM

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

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

Figure 2.1.9. KEYBOARD SCHEMATIC

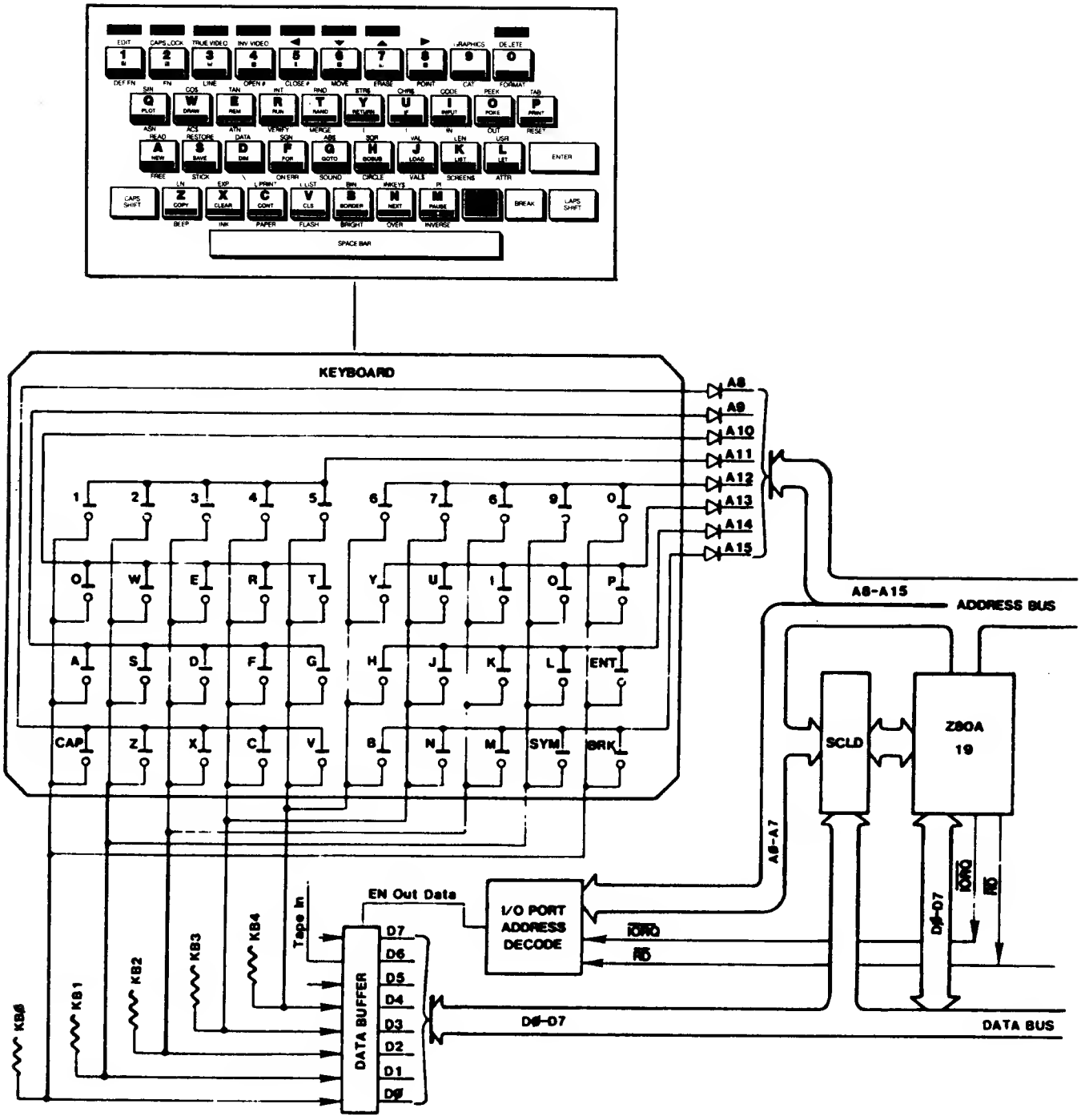


FIG. 2.1.10-1
DISPLAY FILE ORGANIZATION (NORMAL MODE)

		32 BYTES LINE 0			32 BYTES LINE 1				32 BYTES LINE 7		
BLOCK 0	Scan 0	4000	4001.....401F		4020.....403F		.....		40E0	40E1	40FF
	1	4100	4101.....411F		4120.....413F		.....		41E0	41E1	41FF
	2	4200	4201.....421F		4220.....423F		.....		42E0	42E1	42FF
	3	4300	4301.....431F		4320.....433F		.....		43E0	43E1	43FF
	4	4400	4401.....441F		4420.....443F		.....		44E0	44E1	44FF
	5	4500	4501.....451F		4520.....453F		.....		45E0	45E1	45FF
	6	4600	4601.....461F		4620.....463F		.....		46E0	46E1	46FF
	7	4700	4701.....471F		4720.....473F		.....		47E0	47E1	47FF
		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.....503F		.....		50E0	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	LINE 0	LINE 1	LINES 2 - 6	LINE 7
0	5800.....581F	5820.....583F	5840.....58DF	58E0.....58FF
BLOCK	LINE 8	LINE 9	LINES 10-14	LINE 15
1	5900.....591F	5920.....593F	5940.....59DF	59E0.....59FF
BLOCK	LINE 16	LINE 17	LINES 18-22	LINE 23
2	5A00.....5A1F	5A20.....5A3F	5A40.....5ADF	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.

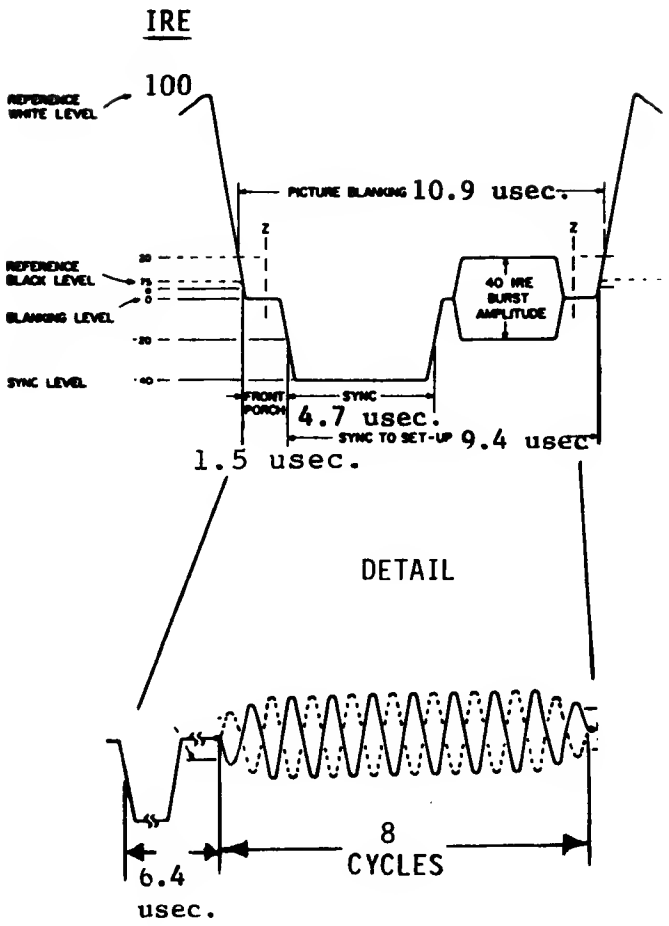
1. The color burst is not synchronous with the waveform since it is generated from the 3.579545 MHz crystal and the waveform is derived from the 14.112 MHz crystal. The result is observed ripples at color boundaries, e.g. green to magenta.

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)

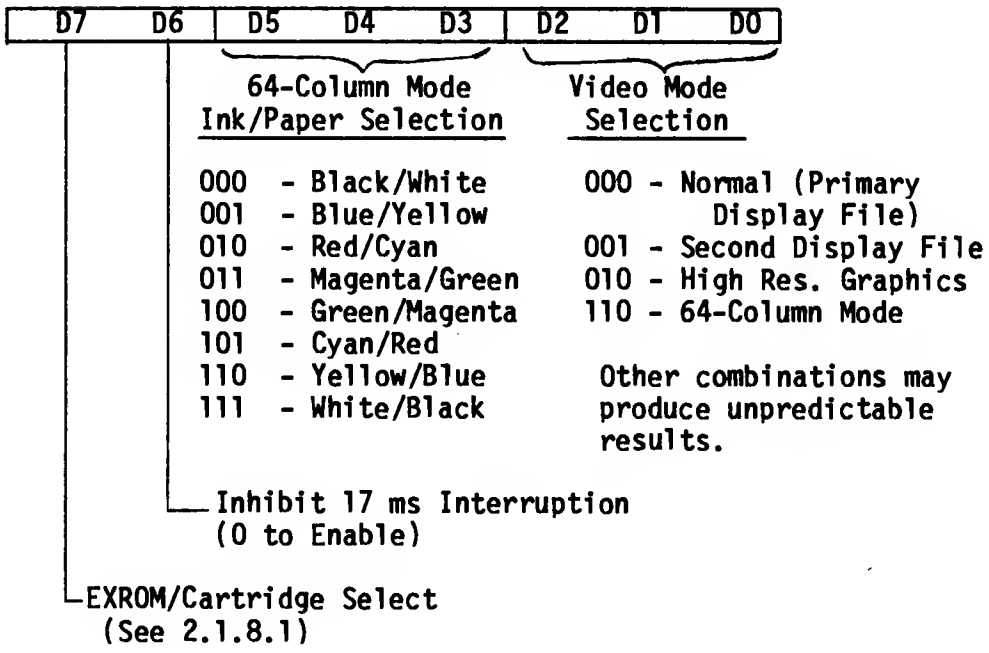


TABLE 2.1.13-1

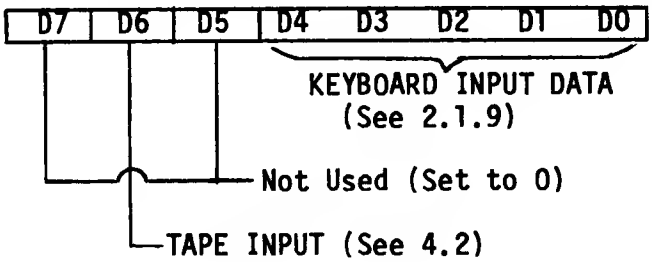
I/O PORT MAP

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

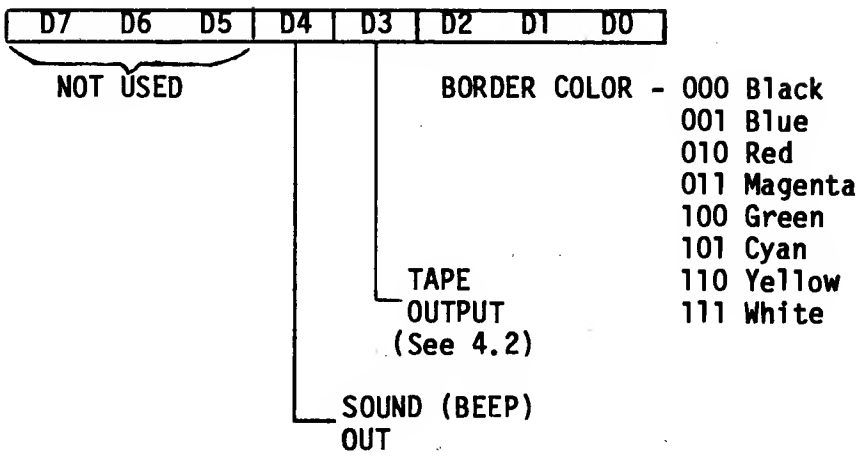
2.1.13.2 Keyboard/Tape I/O (Port FEH)

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

READ (IN)



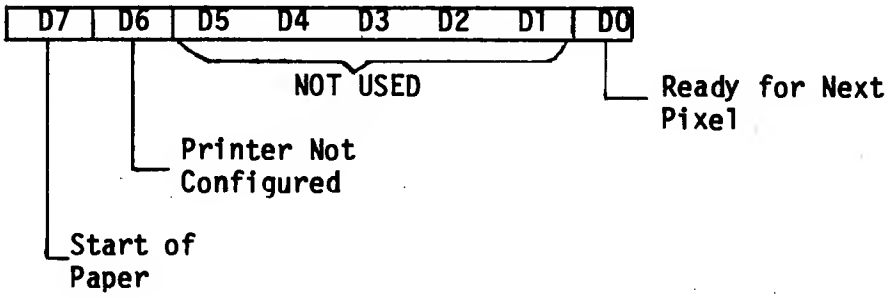
WRITE (OUT)



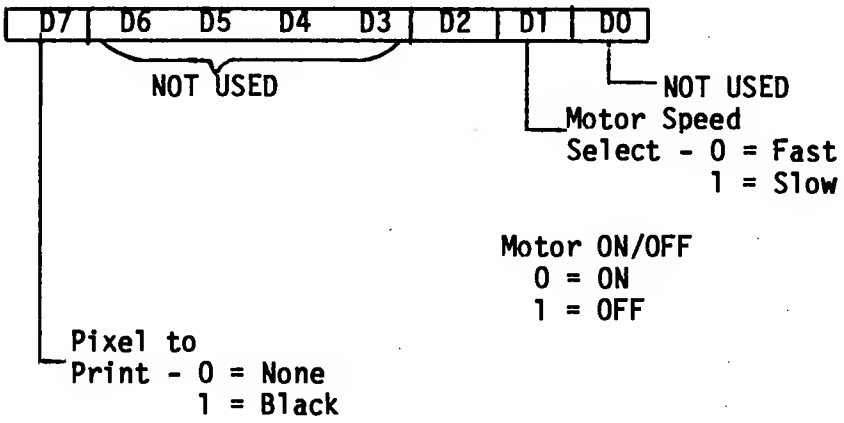
2.1.13.3 TS 2040 Printer (Port 1XXXX0XX)

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 in Sections 2.1.6 and 2.1.7.

2.1.13.5 Horizontal Select Register (Port F4H)

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

2.2 Schematic Diagram

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

2.3 Unit Absolute Ratings

FUNCTION	DESCRIPTION	MIN	MAX
TS	Storage Temperature	-40°C	+65°C
VAC	AC Line Voltage	105V	130V
Ta	Operating Ambient Temp	0°C	40°C
Vin	Voltage on any Logic Pin	-0.3V	+5.3V
Vin (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 via 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

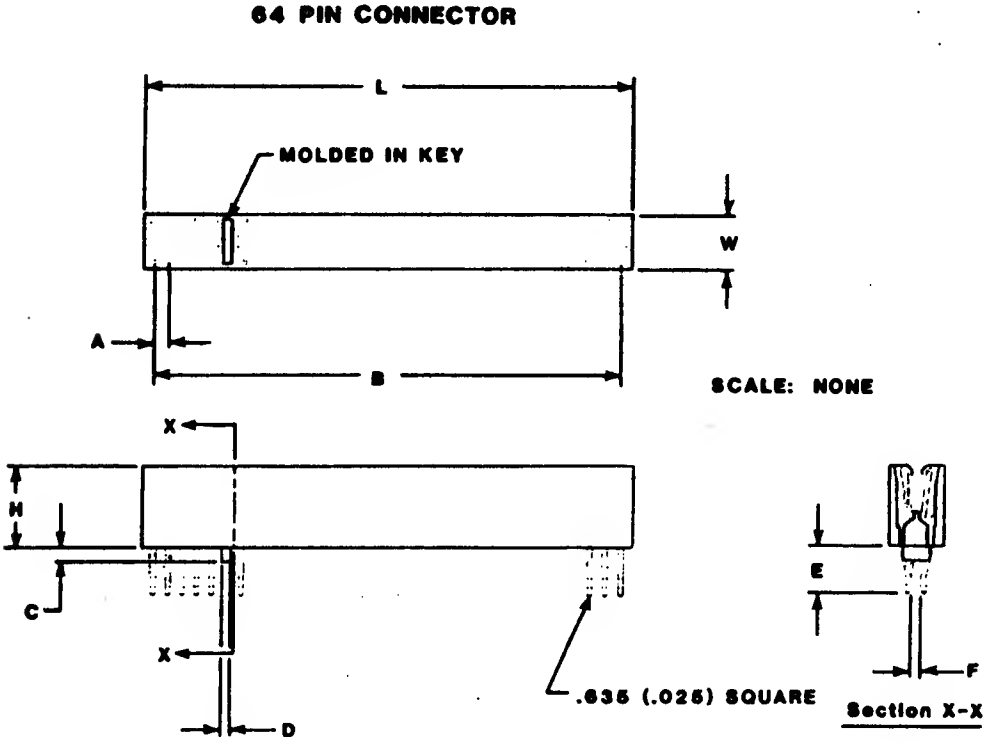
2.4.1 System Bus Connector - P1

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	8.826±0.127 (.378±.005)
H	13.87±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

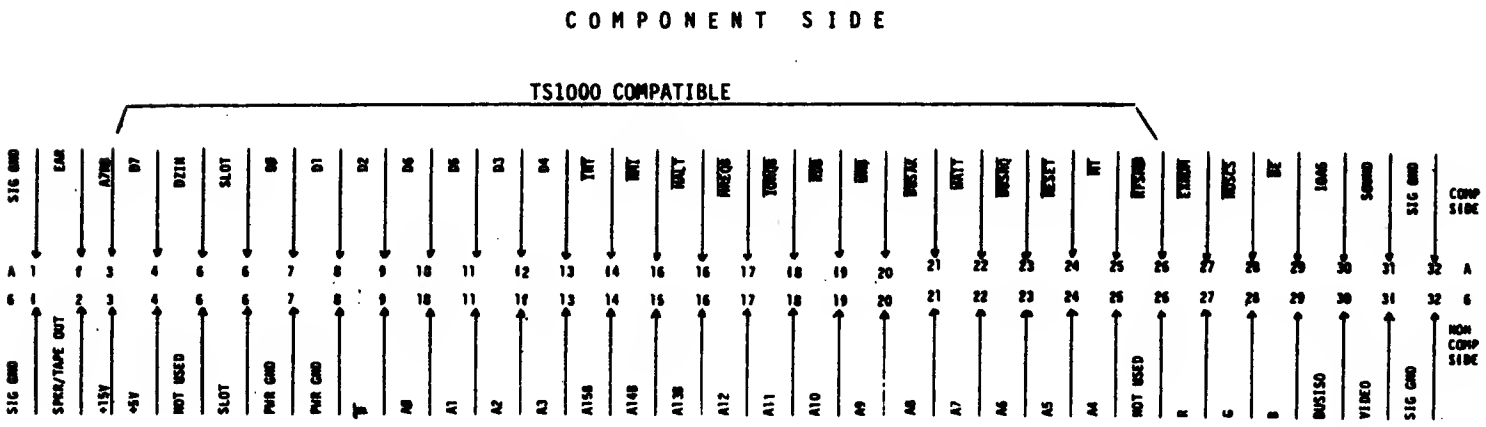
*All dimensions are in millimeters, dimensions shown (XX.X) are in inches.

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 70.1-283.5 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



(VIEW FROM FRONT OF COMPUTER)

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	D0	Data Bus Bit 0
7B	GND	Power Ground
8A	D1	Data Bus Bit 1
8B	GND	Power Ground
9A	D2	Data Bus Bit 2
9B	$\overline{0}$	CPU Clock (Inverted)
10A	D6	Data Bus Bit 6
10B	A0	Address Bus Bit 0
11A	D5	Data Bus Bit 5
11B	A1	Address Bus Bit 1
12A	D3	Data Bus Bit 3
12B	A2	Address Bus Bit 2
13A	D4	Data Bus Bit 4
13B	A3	Address Bus Bit 3
14A	$\overline{\text{INT}}$	Interrupt Request (Active Low)
14B	A15B	Address Bus Bit 15, Buffered
15A	$\overline{\text{NMI}}$	Non-Maskable Int.(Active Low)
15B	A14B	Address Bus Bit 14, Buffered
16A	$\overline{\text{HALT}}$	CPU HALT Indicator (Active Low)
16B	A13B	Address Bus Bit 13, Buffered
17A	$\overline{\text{MREQB}}$	Memory Request (Active Low),Bfrd.
17B	A12	Address Bus Bit 12
18A	$\overline{\text{IORQB}}$	I/O Request (Active Low), Bfrd.
18B	A11	Address Bus Bit 11
19A	$\overline{\text{RDB}}$	Read (Active Low), Buffered
19B	A10	Address Bus Bit 10
20A	$\overline{\text{WRB}}$	Write (Active Low), Buffered
20B	A9	Address Bus Bit 9
21A	$\overline{\text{BUSAK}}$	Bus Acknowledge (Active Low)
21B	A8	Address Bus Bit 8
22A	$\overline{\text{WAIT}}$	CPU WAIT (Active Low)
22B	A7	Address Bus Bit 7
23A	$\overline{\text{BUSRQ}}$	Bus Request (Active Low)
23B	A6	Address Bus Bit 6
24A	$\overline{\text{RESET}}$	CPU Reset (Active Low)
24B	A5	Address Bus Bit 5
25A	$\overline{\text{M1}}$	CPU M1 State (Active Low)
25B	A4	Address Bus Bit 4
26A	$\overline{\text{RFSHB}}$	Refresh (Active Low),Buffered
26B	DZOUT	Daisy Out (Not Connected)
27A	$\overline{\text{EXROM}}$	Extension ROM Enable (Active Low)
27B	R	Color Signal - Red
28A	$\overline{\text{ROSCS}}$	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(LOAO) MAX (MA)	V(OH) MIN VOLTS	V(IL) MAX VOLTS	V(IH) MIN VOLTS	I IN (MAX) µA	INPUT CAPACITIVE LOADING MAX (PF)
A15B	30	0.5	1.8	2.4	0.8	2.0	1800	40
A14B	30	0.5	1.8	2.4	0.8	2.0	1800	40
A13B	30	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	72
A3	30	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
$\overline{\text{IORQB}}$	30	0.5	12	2.4	0.8	2.0	20	10
$\overline{\text{WRB}}$	30	0.5	12	2.4	0.8	2.0	20	10
$\overline{\text{RFSHB}}$	30	0.5	12	2.4	0.8	2.0	20	10
$\overline{\text{EXROM}}$	30	0.5	12	2.4	---	---	----	--
$\overline{\text{ROSCS}}$	30	0.5	12	2.4	---	---	----	--
$\overline{\text{MREQB}}$	30	0.5	12	2.4	0.8	2.0	20	10
$\overline{\text{ROB}}$	30	0.5	12	2.4	0.8	2.0	20	10
$\overline{\text{MI}}$	30	0.4	1.8	2.4	0.8	2.0	20	10
$\overline{\text{8E}}$	--	---	---	---	0.8	2.0	10	12
$\overline{\text{BUSAK}}$	30	0.4	1.8	2.4	---	---	----	--
$\overline{\text{WAIT}}$	--	---	---	---	0.8	2.0	----	10
$\overline{\text{HALT}}$	30	0.4	1.8	2.4	0.8	2.0	----	10
$\overline{\text{NMI}}$	--	---	---	---	0.8	2.0	----	10
$\overline{\text{INT}}$	---------- 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	---------- TO 75 ohm COAX ----------							
D0	30	0.4	1.8	2.4	0.8	2.0	20	120
01	30	0.4	1.8	2.4	0.8	2.0	20	120
02	30	0.4	1.8	2.4	0.8	2.0	20	120
03	30	0.4	1.8	2.4	0.8	2.0	20	120
04	30	0.4	1.8	2.4	0.8	2.0	20	120
05	30	0.4	1.8	2.4	0.8	2.0	20	120
D6	30	0.4	1.8	2.4	0.8	2.0	20	120
07	30	0.4	1.8	2.4	0.8	2.0	20	120
SPKP/TAPE OUT	500	0.2	0.04	0.3-0.5	---	---	----	--
EAR	15	0.5	1.6	2.4	+/- 1.3	+/- 5.0	----	--
SOUND	100	0	---	2.5	-0.3	+5.0	----	--
$\overline{\text{BUSRQ}}$	--	---	---	---	0.8	2.0	----	10
$\overline{\text{RESET}}$	---------- 1 µF 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.

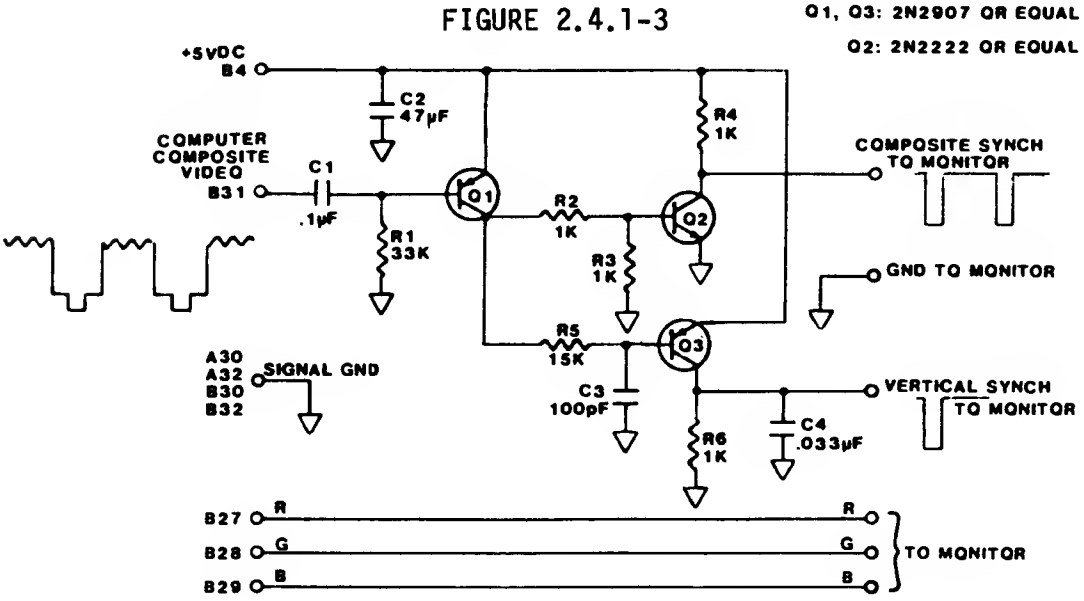
The Schematic of Figure 2.4.1-3 shows the required connections and electronics. Attachment is via the 64-pin keyed P1 connector. Shielding should not normally be required, but ferrite beads are recommended on each wire to minimize EMI, TVI, etc.

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 B27, B28, 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

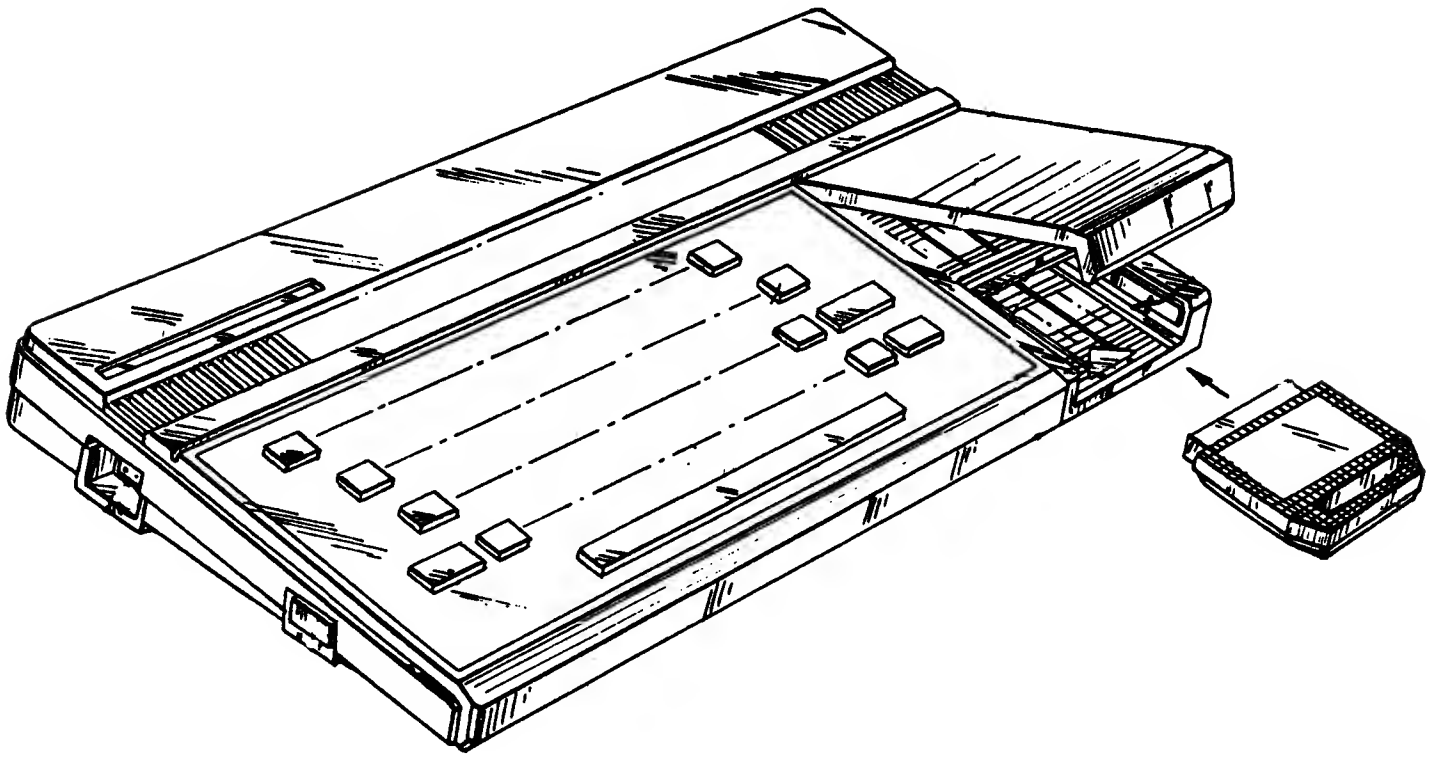
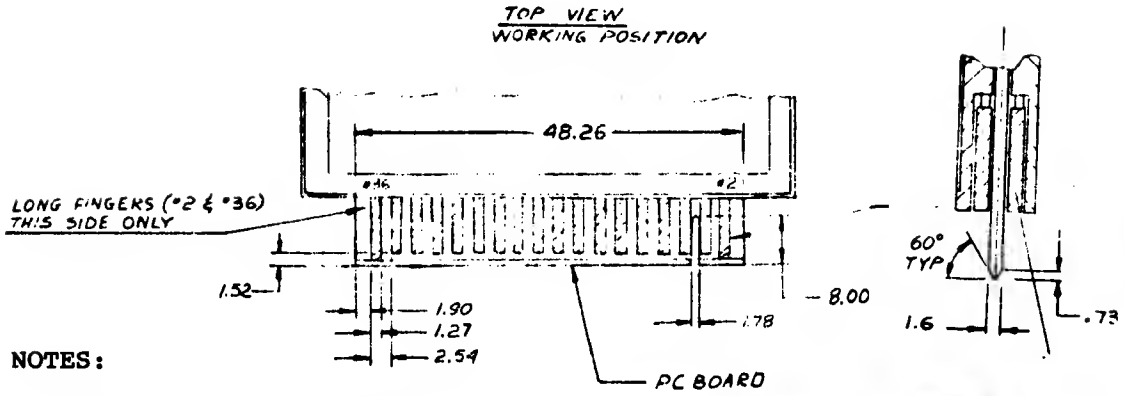


FIGURE 2.4.2-1

J4 MATING PCB MECHANICAL REQUIREMENTS



NOTES:

(1) Circuit Board Material:
FLGFN C62
C1/1A2A (94V-0)
Copper 1 or 2 sides

(2) Contact Fingers: Min. 10 millionth MIL-G - 45204 Gold over .00005 to .00010 inch low stress nickel.

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

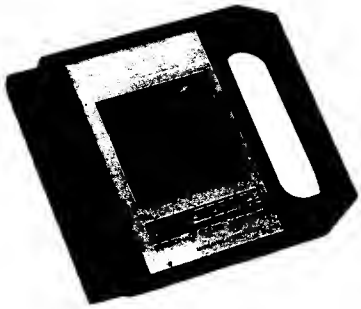


FIGURE 2.4.2-2

J4 SIGNAL LAYOUT

(View from Front)

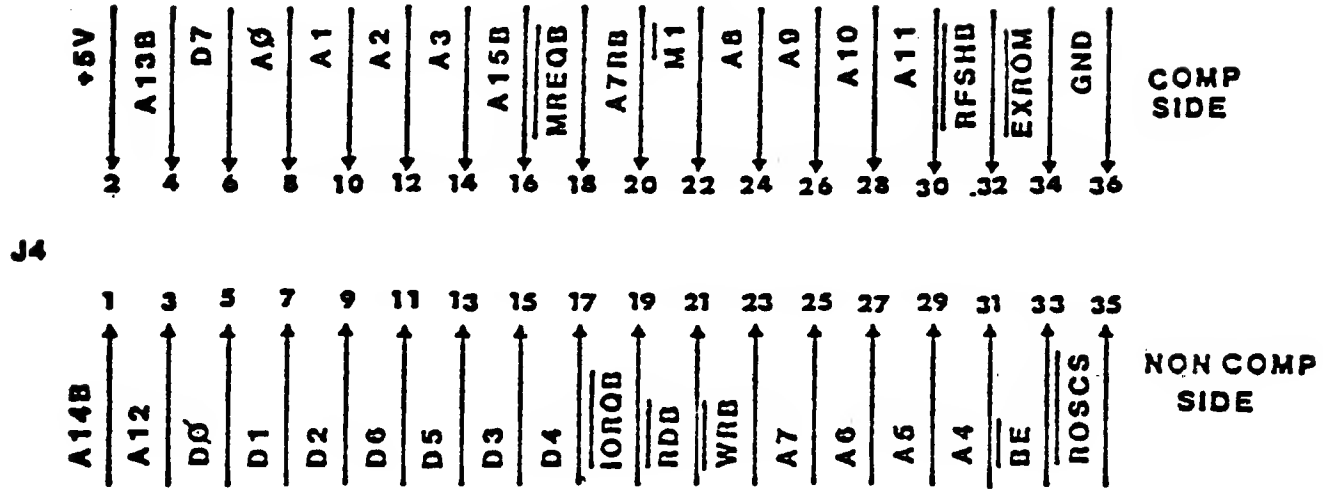


TABLE 2.4.2-1

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	D2	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) MIN (uA)	V(IL) MAX VOLTS	V(IH) MIN VOLTS	I IN (MAX) μA	INPUT CAPACITIVE LOADING MAX (PF)
A15	30	0.5	1.8	2.4	10				
A14	30	0.5	1.8	2.4	10				
A13	30	0.5	1.8	2.4	10				
A12	30	0.4	1.8	2.4	10				
A11	30	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				
A1	30	0.4	1.8	2.4	10				
A0	30	0.4	1.8	2.4	10				
A7RB	30	0.5	0.35	2.7					
$\overline{\text{ROSCS}}$	30	0.4	1.8	2.4	10				
$\overline{\text{MREQ8}}$	30	0.5	1.8	2.4	10				
$\overline{\text{RD8}}$	30	0.5	1.8	2.4	10				
$\overline{\text{IORQ8}}$	30	0.5	12	2.4	10				
$\overline{\text{WR8}}$	30	0.5	12	2.4	10				
$\overline{\text{RFSH8}}$	30	0.5	12	2.4	10				
$\overline{\text{EXROM}}$	30	0.5	12	2.4	10				
$\overline{\text{MI}}$	30	0.5	12	2.4	10				
D0	30	0.4	1.8	2.4		0.8	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 voltages should be between 4.0 and 10.0 V p-p.

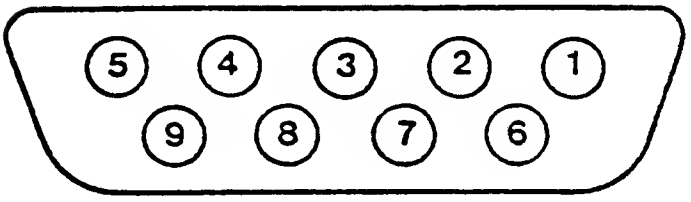
2.4.4 Joystick

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

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

FIGURE 2.4.4-1

JOYSTICK CONNECTOR



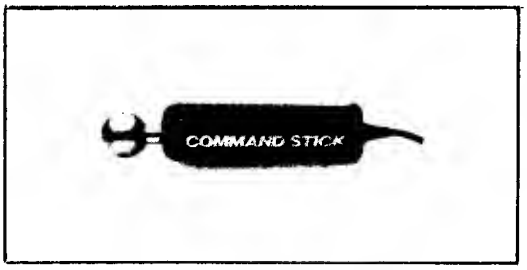


TABLE 2.4.4-1

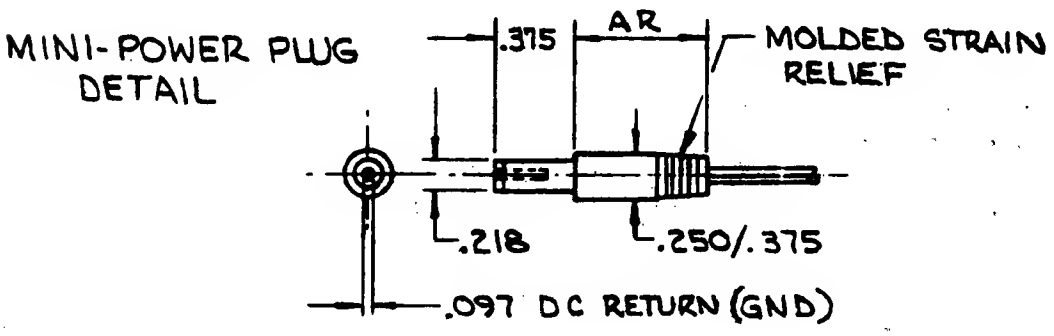
JOYSTICK CONNECTOR SIGNAL ASSIGNMENT

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

3.2 ROM Organization and Services

3.2.1 Home ROM

3.2.1.1 Fixed Entry Points

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

RESTART	FUNCTION
8	ERROR - Error exit from BASIC (Address on Stack points to Error Number)
16	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=1st Byte of Space; HL=Last.

Location 56 (38H) is the entry to service the hardware generated interruption which occurs approximately every 1/60 of a second (16.67 ms). 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 for search and copy operations for both program lines and DATA statements, and when executing a USR function, otherwise the entire Home Bank is enabled while executing in the BASIC Interpreter. There is no support for User-Defined Functions which insert the expanded definition parameters directly into the program and then require search of the program area to find these parameters whenever a function is invoked.

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

3.2.1.3 General

The balance of the Home ROM contains the BASIC Interpreter and standard I/O routines with the exception of the cassette I/O which is in the Extension ROM. The bit map table for the standard character set is located at the end of the Home ROM from location 15616 to 16383 (3D00H to 3FFFH). The address of this table minus 256 (100H) is contained in the System Variable CHARS (=3C00H).

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

3.2.2 Extension ROM

3.2.2.1 Fixed Entry Points

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

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

3.2.2.2 General

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

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

FIGURE 3.2.2-1

Extension ROM Interruption Fielder

```
LOCATION   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       F1            POP   AF             Restore AF
0042       C36EFA        JP    INT7           Chunk 7 if Vidmod not 0
0045       F1       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 Register A which designates the desired video mode as shown in Table 3.2.2-1. All non-zero values involve access to the second display file located at 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 0FFH, 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

```
                    ;                                   ; EXTENSION ROM INTERFACE ROUTINE
=00FC               R_TAPE           EQU     00FCH      ; READ TAPE ROUTINE
=0068               W_TAPE           EQU     0068H      ; WRITE TAPE ROUTINE
=0E8E               CHNG_VIO         EQU     0E8EH      ; CHANGE VIDEO MODE ROUTINE
=5CC2               VIOMOD           EQU     5CC2H      ; VIDEO MODE SYSTEM VARIABLE
                    ;
                    ;
                    ;                                   ; CALL READTP WITH REGISTERS SET
                    ;                                   ; UP FOR R_TAPE ROUTINE
                    ;
0000'  21 00FC      READTP  LD       HL,R_TAPE          ; ADDRESS TO HL
0003'  CD 002D'             CALL     IFRTN              ; ENABLE EXT./EXECUTE RTN
0006'  18 17                JR       EXIT               ; RESTORE HOME BANK AND RETURN
                    ;
                    ;                                   ; CALL WRITTP WITH REGISTERS SET
                    ;                                   ; UP FOR W_TAPE ROUTINE
                    ;
0008'  21 0068      WRITTP  LD       HL,W_TAPE          ; ADDRESS TO HL
000B'  CD 002D'             CALL     IFRTN              ;
000E'  18 0F                JR       EXIT
                    ;
                    ;                                   ; CALL CHGVID WITH DESIRED VIDEO
                    ;                                   ; MODE IN A
                    ;
0010'  21 0E8E      CHGVID  LD       HL,CHNG_VID        ; ADDRESS TO HL
0013'  F5                   PUSH     AF                 ; SAVE VIDEO MODE
0014'  CD 002D'             CALL     IFRTN
                    ;
                    ;                                   ; COMPENSATE FOR 'BUG' IN
                    ;                                   ; CHNG_VID RTN.WHICH SETS
                    ;                                   ; VIDMOD=0 INSTEAD OF 80
                    ;                                   ; WHEN BOTH DISPLAY FILES
                    ;                                   ; ARE OPEN
                    ;
                    ;
0017'  F1                   POP      AF                 ; TEST VIDEO MODE
0018'  FE 80                CP       80H                ; TEST IF 80
001A'  20 03                JR       NZ,EXIT
001C'  32 5CC2              LD       (VIDMOD),A         ; SET VIDMOD=80H
001F'  3A 002C'     EXIT    LD       A,(HSSAVE)         ; GET PREV. HOR.SEL.
0022'  D3 F4                OUT      (0F4H),A           ; RESTORE
0024'  DB FF                IN       A,(0FFH)           ; READ PORT FF
0026'  CB BF                RES      7,A                ; TURN OFF ROM SEL.
0028'  D3 FF                OUT      (0FFH),A
002A'  FB                   EI
002B'  C9                   RET
                    ;
002C'  00           HSSAVE  DEFB     0                  ; SAVE HOR.SEL. (PORT 0F4H)
                    ;
002D'  F3           IFRTN   DI                          ; MASK INTERRUPTIONS
002E'  F5                   PUSH     AF                 ; PRESERVE REG. A
002F'  DB FF                IN       A,(0FFH)           ; EXT.ROM SELECT BIT
0031'  CB FF                SET      7,A                ; SEL. EXT.ROM
0033'  D3 FF                OUT      (0FFH),A
0035'  DB F4                IN       A,(0F4H)           ; HORIZONTAL SELECT FOR DOCK/EXT
0037'  32 002C'             LD       (HSSAVE),A         ; SAVE
003A'  3E 01                LD       A,1                ; SELECT CHUNK 0 IN EXT.ROM
003C'  D3 F4                OUT      (0F4H),A           ;
003E'  F1                   POP      AF                 ; RESTORE REG. A
003F'  E9                   JP       (HL)               ; EXECUTE TARGET ROUTINE AND
                    ;                                   ; RETURN TO CALLER OF IFRTN
                            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.

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

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

3.3.4.1 RAM Interruption Handler

Chunk 3 Entry: 62AEH

Chunk 7 Entry: FA6EH

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

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

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

3.3.4.2 RAM Service Routines

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

TABLE 3.3.4-1

OS RAM SERVICE ROUTINES

LABEL	SERVICE CODE (Decimal)	LOCATION CH.3	LOCATION CH.7	DESCRIPTION
GET_WORD	_	6316	FAD6	Returns in HL the word from the address in HL in the bank specified in B.
PUT_WORD	_	633B	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	FC0D	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	FC1E	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)
GOTO_BANK	-	6572	FD32	Transfers control to the specified address after enabling the memory selected in the specified bank. Parameters passed on stack by pushing target address, then Bank #/Mem.Select prior to calling GOTO_BANK. (Return address is discarded.).
CALL_BANK	-	65D0	FD90	Like GOTO_BANK except saves current bank status, calls target address, and restores status prior to returning to user. Two additional parameters are passed on stack prior to doing call to CALL_BANK. These are PRM_OUT (16-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 CH.3	LOCATION CH.7	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 Scan (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.1.1-4)
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 # and 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 Command (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).
SUBLIN	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-1 statements, then the next line counts as the D'th.). If match on E is found, then returns NZ,NC and both HL and (CH_ADD) point to keyword. D is decremented by the number of statements looked at (e.g. D= -2 if two statements). If no match on E then returns NZ,C with both HL and (CH_ADD) pointing to End-of-Line byte (0DH).

TABLE 3.3.4-2

TS 2068 FUNCTION DISPATCHER SERVICES
(continued)

SERVICE	SERVICE CODE	DESCRIPTION
RECLEN	55 (37H)	Returns in BC the length of the record pointed to by HL. Sets DE to HL+BC. The record can be a program line, or a string or numeric variable or array.
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, otherwise contains Error Number-1.
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_U1	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 3.3.3-2

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 3.3.3-2

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 +1 and the SIN X = SIN (PI/2 * Y).

TABLE 3.3.3-2

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 2-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 following values:

LOCATION	VALUE	
5C10	0100 (Channel -3)	(Copied from SMINIT in module EDIT of the Home ROM)
5C12	0600 (Channel -2)	
5C14	0B00 (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 5C00H. The 2-byte value in the table is an index into the CHANS area of memory which contains the addresses of the I/O routines for the selected channel. If the 2-byte value is zero, the Channel is closed. The STRMS table is modified via the OPEN # and CLOSE # commands. When a Channel is OPENed, the device specification is used to obtain the 2-byte value to be inserted. This value is taken from the table STRMINIT in module EDIT of the Home ROM. When Channels 0 through 3 are CLOSEed, the values are restored to those used at power-on time. All others are cleared to zero.

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

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

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

Whenever an I/O operation is performed, the appropriate Channel is "selected", i.e. its number is used as an index into STRMS to obtain the offset into the CHANS table. This offset is added to

(CHANS)-1 and the resultant pointer is loaded into the System Variable CURCHL for use by the next character I/O operation (WRCH/RDCH). The device specification from CHANS is used to find and execute the 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

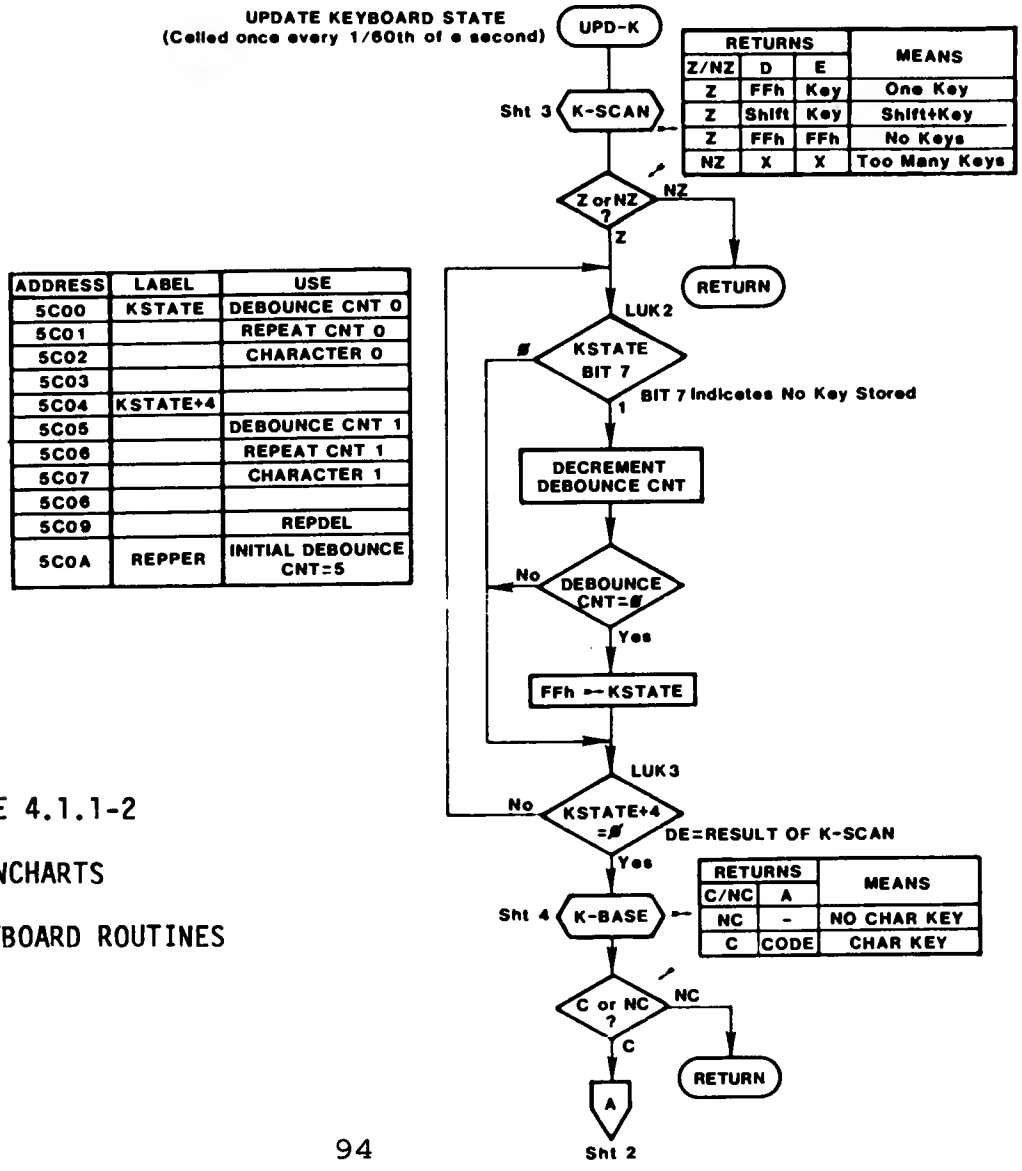
The low-level routines supporting keyboard input are executed every 1/60 of a second out of the Interruption Handler (Location 56 (38H)). The controlling routine is labelled UPD_K. This routine calls K_SCAN to determine if any key(s) are currently being depressed, controls the debouncing and repeat algorithms,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

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

FIGURE 4.1.1-1

TS 2068 MODE CONTROLS

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



RETURNS			MEANS
Z/NZ	D	E	
Z	FFh	Key	One Key
Z	Shift	Key	Shift+Key
Z	FFh	FFh	No Keys
NZ	X	X	Too Many Keys

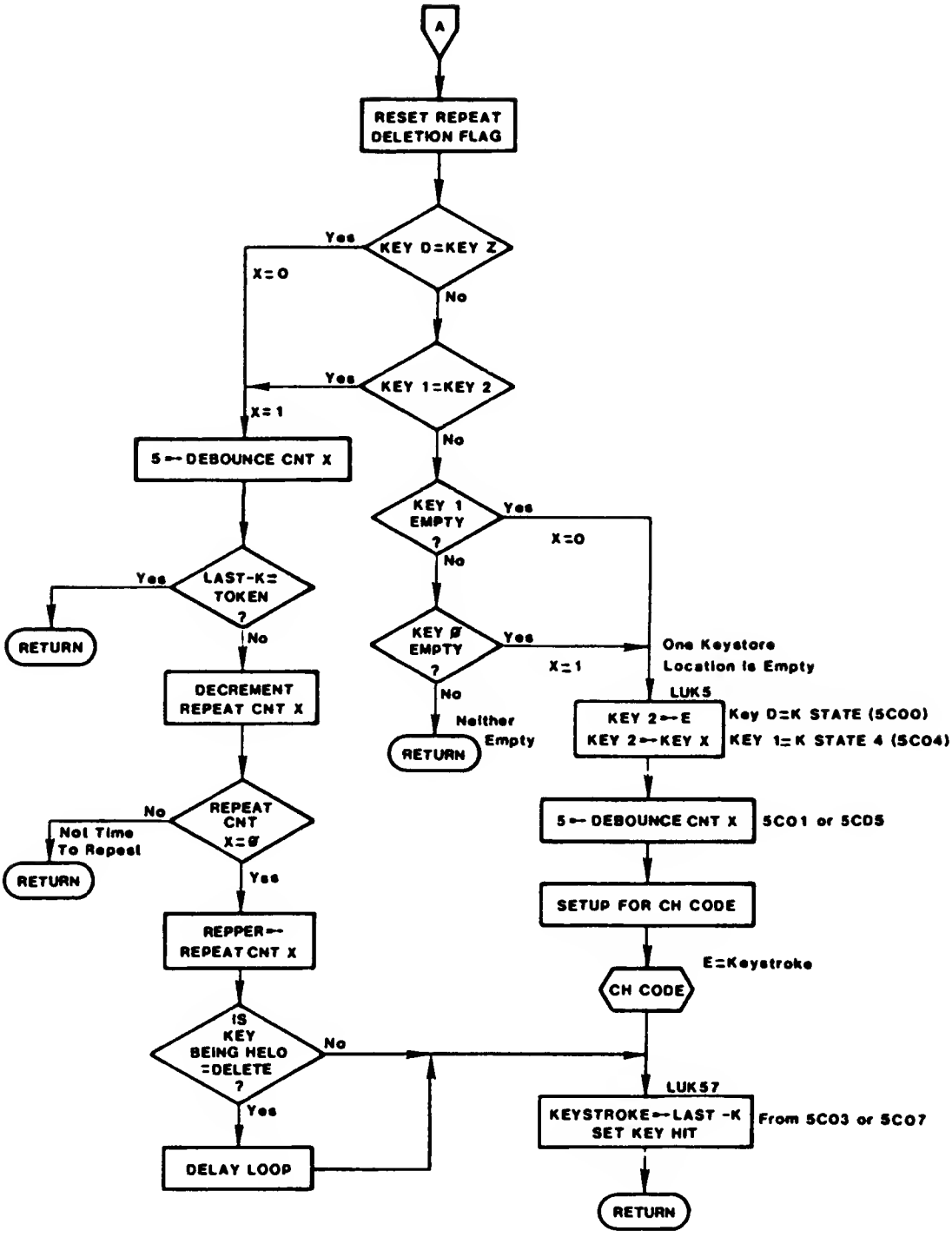
ADDRESS	LABEL	USE
5C00	KSTATE	DEBOUNCE CNT 0
5C01		REPEAT CNT 0
5C02		CHARACTER 0
5C03		
5C04	KSTATE+4	
5C05		DEBOUNCE CNT 1
5C06		REPEAT CNT 1
5C07		CHARACTER 1
5C08		
5C09		REPDEL
5C0A	REPPER	INITIAL DEBOUNCE CNT=5

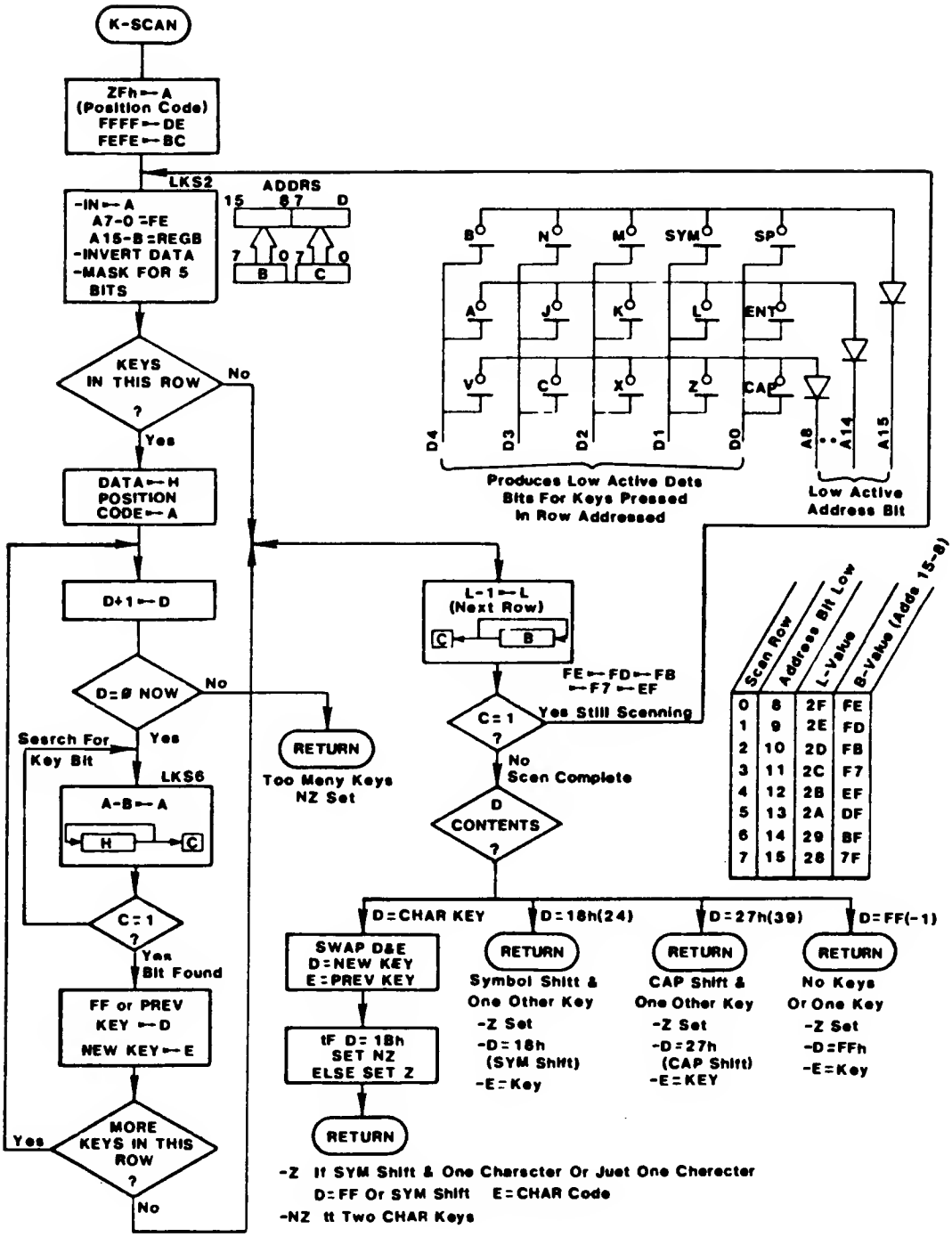
RETURNS		MEANS
C/NC	A	
NC	-	NO CHAR KEY
C	CODE	CHAR KEY

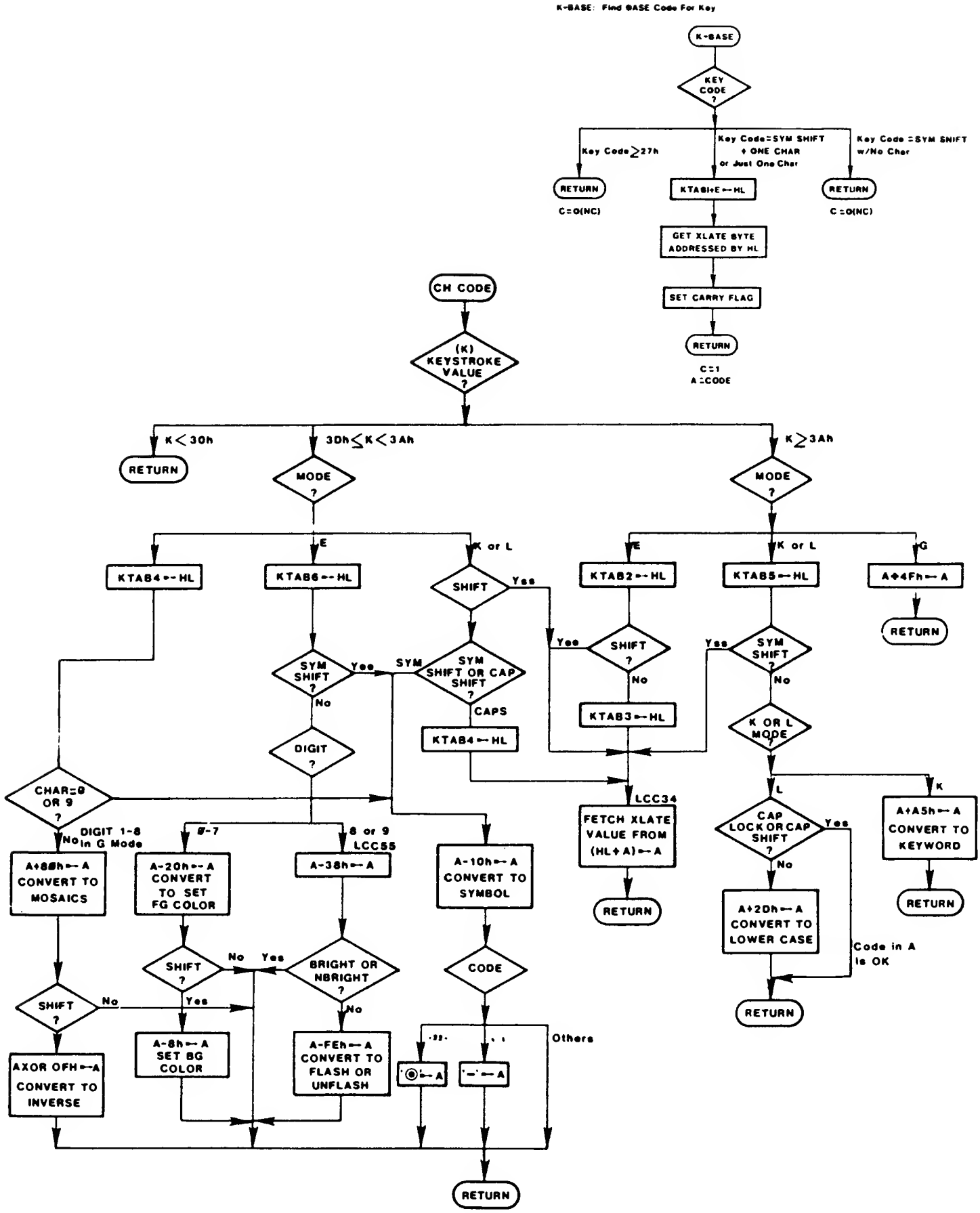
FIGURE 4.1.1-2

FLOWCHARTS

TS 2068 KEYBOARD ROUTINES


A
RESET REPEAT DELETION FLAG
KEY D=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 D=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
REPPER ← REPEAT CNT X
SETUP FOR CH CODE
E=Keystroke
CH CODE
IS KEY BEING HELD =DELETE ?
No
Yes
DELAY LOOP
LUK57
KEYSTROKE ← LAST -K
SET KEY HIT
From 5C03 or 5C07
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-B→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
Produces Low Active Dets
Bits For Keys Pressed
In Row Addressed
A8 A14 A15
Low Active
Address Bit
Scan Row
Address Bit Low
L-Value
B-Value (Addr 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
tF D=18h
SET NZ
ELSE SET Z
RETURN
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 Juet One Cherecter
D=FF Or SYM Shift E=CHAR Code
-NZ tt 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)
KTAB1+E←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
A+80h←A CONVERT TO MOSAICS
SHIFT ?
No
Yes
AXOR 0FH←A CONVERT TO INVERSE
0-7
A-20h←A CONVERT TO SET FG COLOR
SHIFT ?
No
Yes
A-8h←A SET BG COLOR
8 or 9
LCC55
A-38h←A
BRIGHT OR NBRIGHT ?
Yes
No
A-FEh←A CONVERT TO FLASH OR UNFLASH
A-10h←A CONVERT TO SYMBOL
CODE
'0'
'1'
Others
'0'←A
'-'←A
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 ?
L
K
CAP LOCK OR CAP SHIFT ?
Yes
No
A+20h←A CONVERT TO LOWER CASE
A+A5h←A CONVERT TO KEYWORD
RETURN
Code in A is OK
RETURN
LCC34
FETCH XLATE VALUE FROM (HL+A)←A
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 lines) and the lower screen (2 lines). The lower portion of the screen is used for output of system messages and to echo input from the keyboard of BASIC commands, BASIC program lines, or data. The lower screen expands as needed for multi-line input, scrolling the entire screen upwards. The variable DF SZ reflects the number of lines in the lower screen (default=2).

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

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

FIGURE 4.1.2-1

TS 2068 STANDARD CHARACTER TABLES

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 passes control to the SENDTV routine based on (CURCHL).

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

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

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

6. The attribute byte controlling the character position just written is updated based on the value in ATTR_T and other flags.

7. The variables S_POSN and DF_CC are updated to reflect the next screen position and return is made from the WRCH operation.

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

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

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

FIGURE 4.1.2-2

DISPLAY FILE ROW/COLUMN NOTATION

BASIC Parameters	Internal Representation	
Line/Row 0	24 (18H)	
1	23 (17H)	
.	.	UPPER
.	.	SCREEN
21	3	
22	2	LOWER
23	1	SCREEN
Column 0	33 (21H)	
1	32 (20H)	
.	.	
.	.	
31	2	

4.1.3 2040 Dot Matrix Printer

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

The Print Buffer is located at 23296 (5B00H) and is 256 (100H) bytes long, the data needed to print one line of 32 characters, each character comprised of 8 bytes (8 X 8 pixels/character). The buffer is cleared to zeros and the flag PRLEFT set to zero at power-on time (or execution of a NEW command). The PRLEFT flag is set to 1 whenever pixel data is written to the buffer. This flag is used when exit is made from a program to print any unprinted data prior to program termination. As the pixel data for a particular character is entered into the buffer, the buffer address is incremented by 32 (20H); the sequential data in the buffer therefore represents 8 complete scan lines of 32 characters. When the Print Buffer is full, or upon processing an End-of-Line (0DH), 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 0FBH, 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 0FA through 0F8 and Ports 0F3 through 0F0 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 4.2-1 shows the header formats for the various types of data.

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

1. Call W_TAPE with A=0. IX contains the address of the header and DE contains the length.

2. Delay loop approximately 1 second.

3. Call W_TAPE with A=FFH. IX contains the address of the data block and DE contains the length.

The R_TAPE routine performs either a LOAD (transfers data from tape to memory) or VERIFY (compare data from tape against data in memory) operation, based on the status at entry: Carry Set for Load and No Carry if Verify. As for the Write, A=Block Type (0 for Header and -1 (FFH) for Data Block). IX contains the memory address.

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

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

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

FIGURE 4.2-1

TAPE HEADER FORMATS

	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 0F5H and 0F6H respectively. The joysticks are read by first addressing Register 14 in the PSG by writing a 14 (0EH) to Port 0F5H. The data is then read by executing an IN from Port 0F6H, having the port address in Z80 Register C and the joystick (player) number in Register B (number = 1 or 2). Note that PSG Register 7, Bit 6 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.

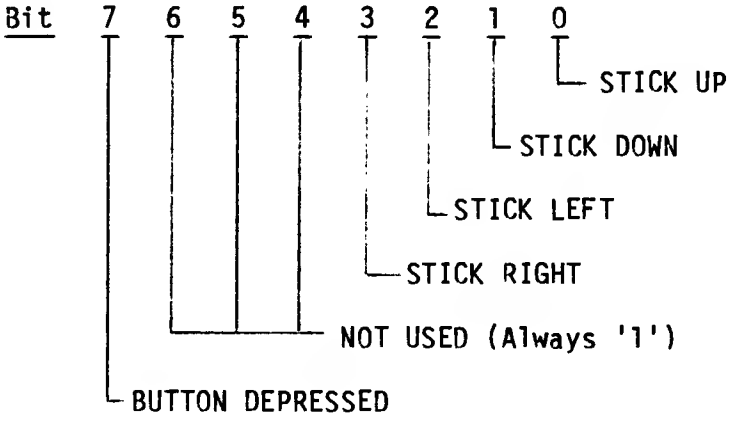
Sample routine:

```
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 0FEH to generate a signal of a calculated frequency and duration based on the command parameters. It uses the routine PARP which takes as input two parameters, one defining the period of the signal (HL) and the other defining the number of cycles to be generated (DE) and outputs DE+1 cycles of a tone having the period 8N+236 to 8N+246 T-States where (HL) = N. Both the BEEP and PARP routines are in the K_SCAN module of the Home ROM. The PARP routine is also used to generate the keyboard "click" and the "raspberry" which can be varied by modifying the values in the system variables PIP (23609/5C39H) and RASP (23608 5C38H).

4.5 Sound Chip (SOUND)

The SOUND command writes the first parameter (register number) to Port 0F5H (address to Programmable Sound Generator) and the second parameter (load data) to Port 0F6H (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 Bank Enable code, execution will continue from that point in Chunk 3 in the Dock Bank with the Stack Pointer addressing ROM.

An LROS is Z80 machine code and is in complete control of the TS 2068 hardware after transfer to the starting address has been made. It can directly implement an application, or it

can support multiple applications by implementing a language other than BASIC. An AROS dependent on such an LROS would have to be part of the same cartridge since there is only one cartridge connector.

Interruption Mode 1 has been set by the TS 2068 and interruptions are enabled prior to passing control to the LROS starting address, therefore the LROS must contain appropriate code at location 56 (38H) to cover the case where the interruption occurs after Chunk 0 in the Dock Bank has been enabled, but before any action by the software cartridge to disable the interruption has been taken. Once control is transferred, the LROS may then disable the standard TS 2068 interruption by setting bit 6 of Port FFH, mask the interruption by executing a DI instruction, or set a different Interruption Mode. It may change the location of the Machine Stack. It may also change the memory selection by writing to Port 0F4H with each bit set to 1 for the corresponding chunk to be enabled in the Dock Bank (high active format) or 0 to be enabled in the Home Bank. Thus, an LROS may contain code in Chunk 3, but it should be enabled after the OS RAM code has finished execution.

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

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

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

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

instruction, for example, would pass control to the address on the stack. Code to switch memory back to the Dock Bank could be mapped in a similar way.

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

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

5.1.2 AROS

An AROS is identified by the following overhead bytes:

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 be 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, i.e. making use of the multi-statement feature of the TS 2068. The reverse is true concerning execution of READ commands. An entire DATA statement is copied to the Home Bank RAM, but only the current item is accessed. It therefore will be more efficient to not make DATA statements excessively long. The BASIC program lines appear in the cartridge in exactly the same format used in the RAM, i.e. Line Number (2 bytes), Length (2 bytes), Command Token, etc. terminated by an Enter (0DH). Numerical constants appearing in a program line are followed by the CHR$ (0EH) byte and 5-byte floating point format described in the User Manual (see Appendix C of the TS 2068 User Manual). The Variables area is built in the RAM (address in VARS) exactly as though the program were in the RAM. All variables, including arrays, are built at the time of program execution - there is no provision for copying or accessing pre-defined variables from the cartridge, however, see Section 5.3.2. The last program line must be followed by a terminator byte having the Most Significant Bit set (e.g. 80H), otherwise the Interpreter cannot detect the end of the program.

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

executing your BASIC program, you must enable Chunks 0-3 in the Home Bank and have the stack and other Home Bank RAM in the proper state for return to the USR function code in the BASIC Interpreter, i.e. what it was when the USR function passed control to you.

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

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

The Autostart parameter should be set to 1 since if it is zero, control will be passed to the BASIC Interpreter as if the cartridge were not present. There is no BASIC command to directly start execution of a Machine Code AROS.

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

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

5.1.3 EPROM Cartridge Board Application

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

See Section 2.4.2 for mechanical details of the connector portion of the PCB.

FIGURE 5.1-1

PLUGGABLE EPROM CARTRIDGE BOARD

LOGIC DIAGRAM

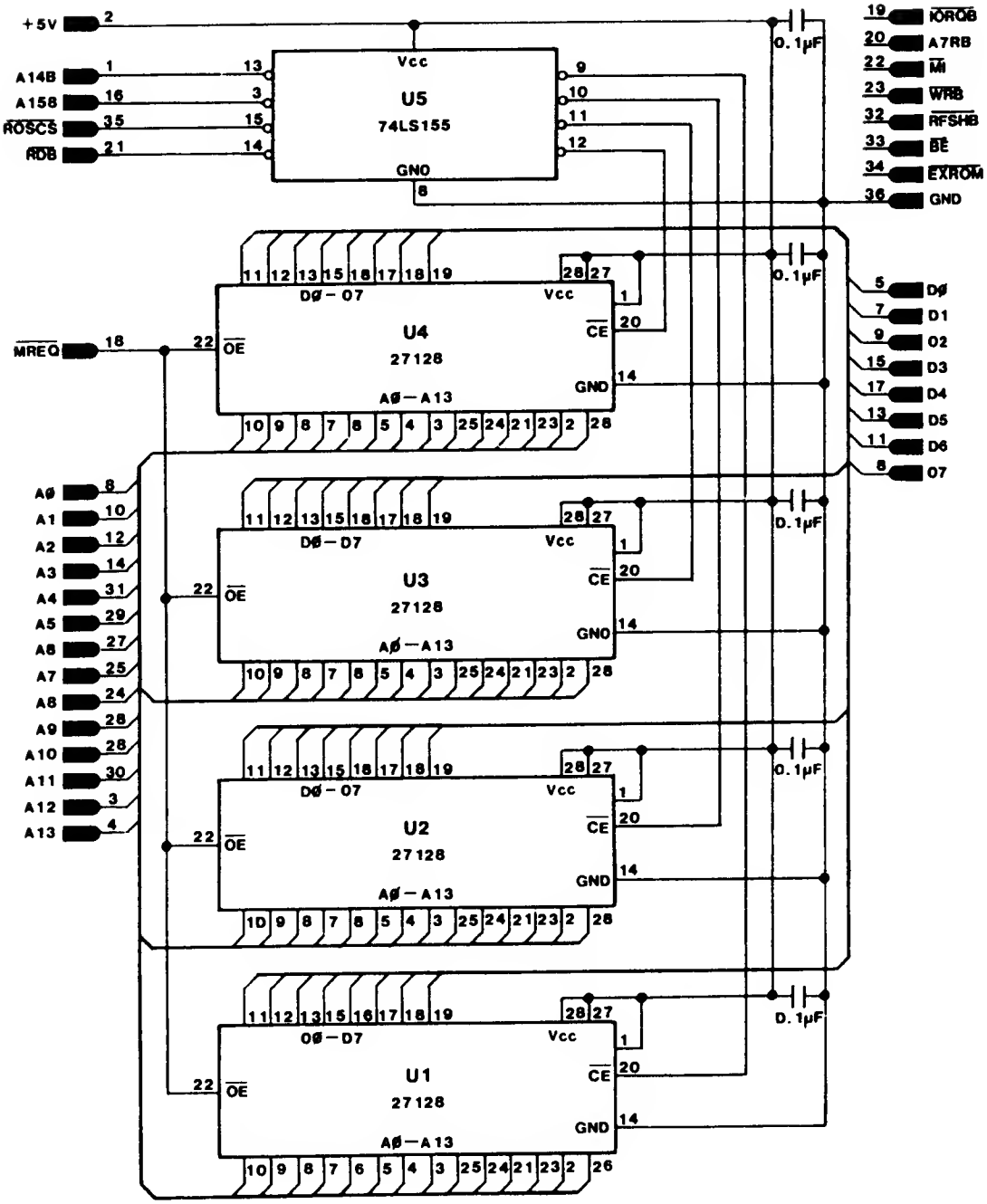
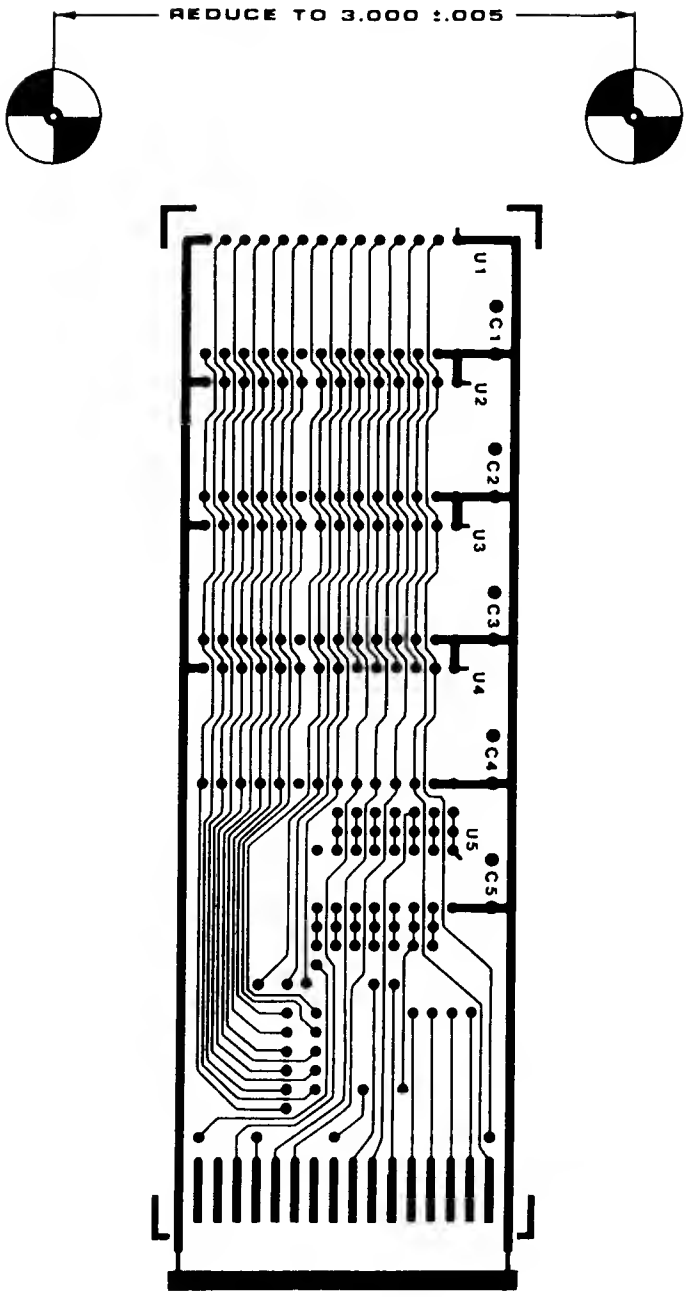


FIGURE 5.1-2

EPROM CARTRIDGE BOARD

COMPONENT SIDE ARTWORK



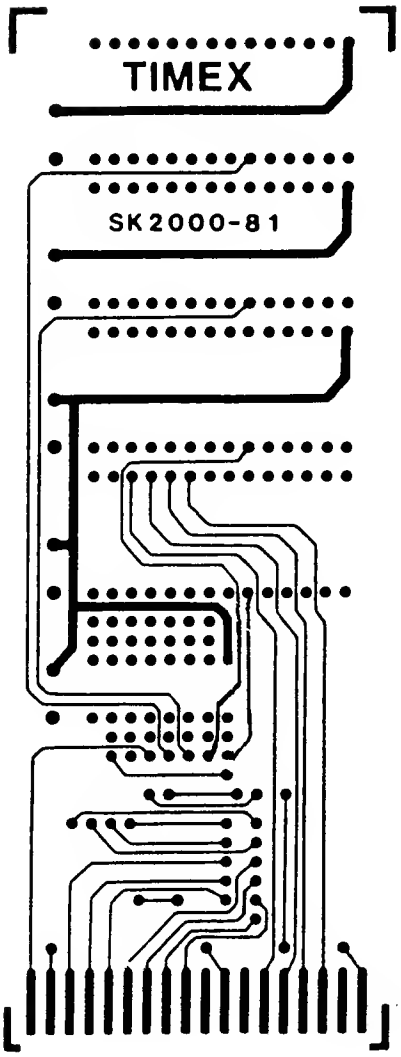
COMPONENT SIDE

SK2000-81

FIGURE 5.1-3

EPROM CARTRIDGE BOARD

SOLDER SIDE ARTWORK



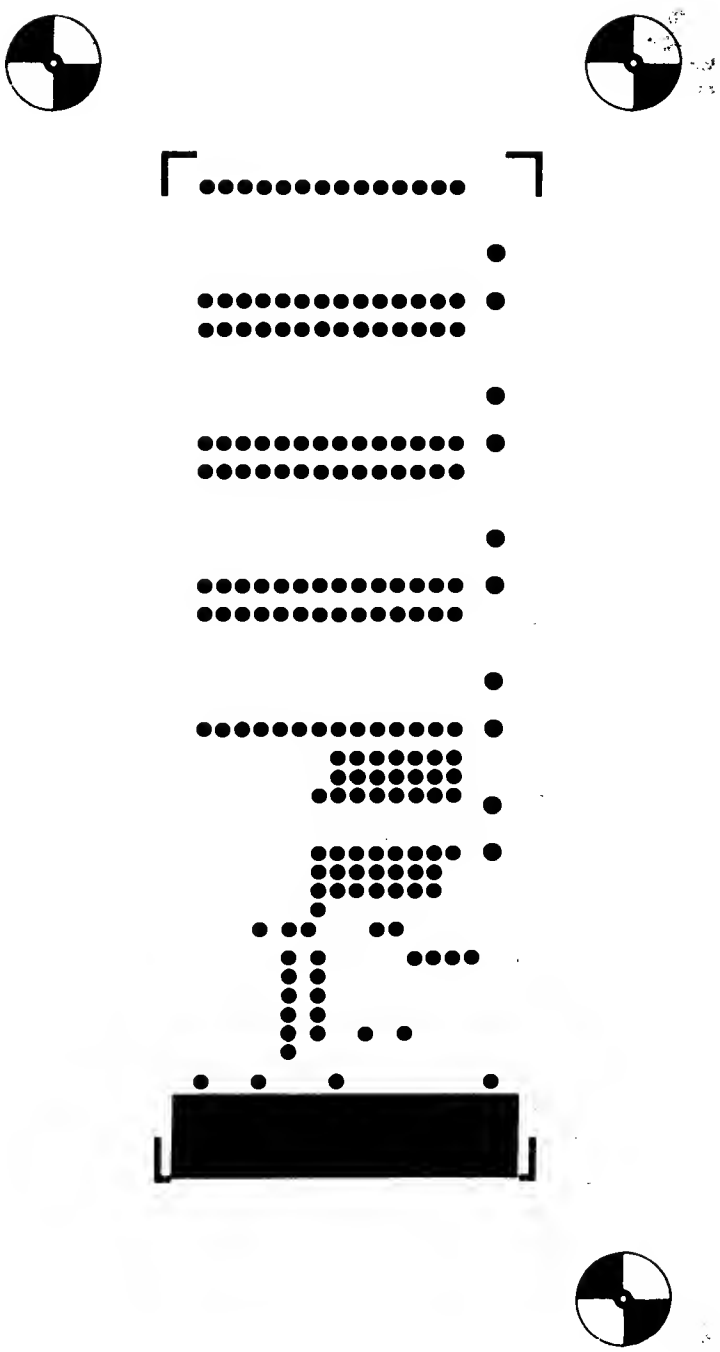
SOLDER SIDE

SK2000-81

FIGURE 5.1-4

EPROM CARTRIDGE BOARD

SOLDER MASK



SOLDER MASK

SK2000-81

5.2 Advanced Video Modes

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

The TS 2068 video mode hardware works out of two areas of RAM, the primary display file at 4000H and the second display file at 6000H. Each area consists of 6912 (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:

Address Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	DF1

4000 - 5AFFH (Bit 13 = 0)

Address Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	DF2

6000 - 7AFFH (Bit 13 = 1)

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

One method of determining the starting display file address for a particular line/column position is to build a table containing the starting address of each of the 24 lines (2 bytes per entry). Then construct an algorithm that takes the

line number and forms an index by multiplying it by 2 (shift left 1), add the index to the base address of the table, and read out the display file address. The column position is then simply an offset added to this address. By testing 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 40		Line 1
2	4	40 40		Line 2
.	.	(+20H)		
.	.	(+20H)		
7	14(0EH)	E0 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)	E0 48		Line 15(End of Middle Block)
16	32(20H)	00 50	5000H =	Line 16(Top of Bottom Block)
17	34(22H)	20 50		Line 17
.	.	(+20H)		
.	.	(+20H)		
23	46(2EH).....	E0 50		Line 23(End of Bottom Block)

Line 17, Column 23 (11H/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=1) we would set Bit 13 of the starting line address getting 7020H, then divide the column address by 2 (shift right 1) since there are only 32 columns in each display file. This would give us an offset of 11 (0BH) which added to the starting address results in a display file address of 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&1
          AND   3           Clear other bits
          OR    58H         OR in Attr.File Base Adrs.
          LD    H,A         Update MSB

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

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

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

In addition to data insertion, two major screen support functions are scrolling and clearing the screen. Scrolling is done in the System ROM by copying the entire display file data and attribute controls up one line position (Line 1 to Line 0, Line 2 to Line 1, etc.) and inserting a blank line at the bottom. Numerous more elaborate scrolling techniques can be implemented using various directions (up, down, left,

right) and smaller areas or "windows" of the screen. Similarly, clearing the screen, which consists of writing zeros to the data file and updating the attribute bytes to a uniform value, can be implemented on smaller sections of the screen. The software packages in Appendix C contain examples of such implementations.

5.2.1 Dual Screen Mode

In this mode the second display file is used to provide a second independent screen having the same data and attribute organization as the primary display file. By writing to Port FFH with Bits 0-5 = 1 (Bit 0 set), the second display file is activated at the video screen. Appendix C contains a software package supporting Dual Screen Mode. The software package uses the system variable 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 Mode

This mode is set by writing to Port OFFH with Bits 0-5=2 (Bit 1 set). In this mode, also called Extended Color Mode, the second display file is used to expand the number of Attribute bytes from one for each 8 X 8 pixel group to one for each 8 X 1 pixel group thus giving 32 X 192 positions within each of which two colors plus Bright and Flash can be defined. Each byte of pixel data entered into the primary display file has

its own Attribute byte in the corresponding location in the second display file, e.g. the byte written to Location 4000H has its Attribute byte at Location 6000H, the byte at 47FFH (last byte of last scan row in Line 7) has its Attribute byte at Location 67FFH, the byte at 57FFH (last byte of last scan row in Line 23) has its Attribute byte at Location 77FFH. The routine writing data to the screen would therefore enter the pixel data to the desired location and then set Address Bit 13 of the Primary Display File address and write the desired attribute control byte to the resultant location. If normal characters are being written to the screen in this mode, eight Attribute bytes must also be written, one for each of the bytes defining the character. The same technique would be used for writing to both display files, i.e. for each of the seven bytes entered after the first, the display file address would be incremented by 256 (100H).

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

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

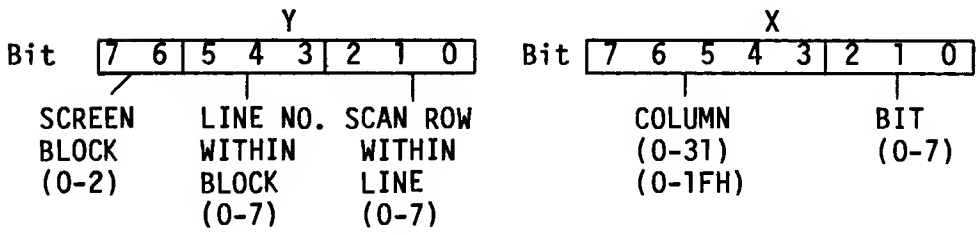
The two inputs are assumed to be as follows:

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               0=Leftmost (MSB);7=
;                           Rightmost (LSB)
      LD    B,A             Use as Control Count
      INC   B               B=1-8
      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 3.2.2.1 and 3.3.3.1. Because the Video Mode Change Service automatically updates internal branch addresses in the OS RAM code when it is relocated between Chunk 3 and Chunk 7, you probably do not want to directly overlay the OS RAM Interruption Handler with your own code if you will be using the Video Mode service. Your branch instruction at 62AEH, however, will be copied unmodified to location FA6EH in Chunk 7 and vice versa.

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

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

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

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

6.2 Machine Code AROS

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

6.3 BASIC AROS

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

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

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

6.4 Video Mode Change Service

6.4.1 Available Memory Test - When the size of memory needed is calculated by adding the size of the second display file (6912 bytes or 1B00H) to the memory now in use (address in System Variable STKEND), the code fails to check for overflow. Thus if the address in STKEND is greater than 58623 (E4FFH), the fact that there is not enough free memory to open the second display file will not be detected and the system will "crash". If your BASIC program and/or variables area are large, you may want to make this test yourself prior to invoking the Video Mode Change Service in order to avoid this problem. The size of memory needed is subsequently tested against the contents of RAMTOP and if there is not sufficient space (value in RAMTOP is less than size needed), you will get Error 4, Out of Memory.

6.4.2 RAMTOP - When the machine stack and OS RAM code is moved to Chunk 7, the User Defined Graphics area is moved down in RAM by 2112 bytes (840H) to make room for the stack and OS RAM routines at the top of memory. The pointer in UDG is updated, however, the value in RAMTOP is not modified to insure that the relocated UDG area as well as the OS code and stack are protected from expansion of the BASIC program. You can avoid problems by setting RAMTOP via a CLEAR command specifying an address no greater than 63255 (F717H) prior to invoking the Video Mode Change Service. This reserves space between RAMTOP and the end of memory of 2280 bytes (8E8H) utilized as:

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 be cleared to zeros by the NEW command.

6.4.4 VIDMOD - When Mode 128 (80H) is designated for activating the Primary Display File in Dual Screen Mode the System Variable VIDMOD at 23746 (5CC2H) is set to zero instead of to 128. This creates a potential problem if the 17 ms. interruption occurs before VIDMOD can be corrected since the interruption fielder will branch to Chunk 3 instead of to Chunk 7 and Chunk 3 is now in use for the second display file. This problem is corrected by disabling the interruption prior to calling the Video Mode Change Service and setting VIDMOD to the correct value prior to re-enabling it. These corrections are included in the Extension ROM Interface Routine in Figure 3.2.2-2.

NOTE: On an initial access changing video mode from normal to Mode 128, the interruption is re-enabled within the Video Mode Change Service itself after copying the stack and other Chunk 3 data to Chunk 7. This cannot be corrected, but has not proven to present a problem in actual use. At the point where the interruption is first enabled, the Chunk 3 code is still intact allowing for correct processing of one interruption, and the path length from there to the point of correcting VIDMOD is apparently less than 17 ms. The interruption is also re-enabled within the Video Mode Change Service if you have applied the patches for the BANK_ENABLE and RESTORE_STATUS routines (Section 6.5.4) which are executed in connection with inserting space into the RAM to open the second display file. Again, this has not proven to be a problem in actual use.

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

6.5 OS RAM Routines

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

6.5.1 Function Dispatcher -For a variety of reasons such as conflict with use of the IX Register, incorrect entries in the ROM Function Dispatcher Jump Table, etc. some Service Codes have been deleted from the Function Dispatcher table (Table 3.3.4-2). In addition, the following correction to the GET_STATUS routine is required in order to successfully utilize the Function Dispatcher from a cartridge.

6.5.2 GET_STATUS- Returns invalid memory selection status for the Home Bank, ROM Extension and Dock. This results in switching out of either the Home Bank or the Dock when status is "restored". This affects use of the Function Dispatcher and GET_WORD routines, and any other code using GET_STATUS. Figure 6.5-1 shows the patches and additions necessary to correct this routine.

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

6.5.4 BANK_ENABLE and RESTORE_STATUS-

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

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

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

FIGURE 6.5-1

GET_STATUS CORRECTIONS

```
   LOCATION   OBJ.CODE    SOURCE STATEMENT               COMMENTS
    (HEX)     (HEX)

                          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    0FEH        Test if Ext.(254)
*   640A      2824                    JR    Z,GS_EXT
    640C      FEFF                    CP    0FFH        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,0FFH      Assume none
*   6432      DBFF                    IN    A,(0FFH)    Test if selected
*   6434      E680                    AND   80H
*   6436      2812                    JR    Z,GS_XT1    Not active
*   6438      1808                    JR    GETHS       Get Hor.Select
*   643A      0EFF        GS_DOCK     LD    C,0FFH      Assume none
*   643C      DBFF                    IN    A,(0FFH)    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      D1          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 6.5-1

GET_STATUS CORRECTIONS

(continued)

From BASIC:

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

FIGURE 6.5-2

PUT_WORD CORRECTIONS

	LOCATION (HEX)	OBJ.CODE (HEX)	SOURCE STATEMENT			COMMENTS
			Input: Data in DE, Address in HL, Bank # in B			
	633B	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	Enbl.Target Mem.
*	6350	C1		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	CD9964		CALL	BANK_ENABLE	Restore Bank St.
	6359	C1		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
	651CH	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 26474,95).

6.6 GENERAL

6.6.1 Pressing ENTER multiple times with an invalid tape command on the edit line (syntax error) causes the system to reset. This is due to overflowing the Bank Status Stack in RAM Chunk 3/7 due to the multiple calls to and from the Extension ROM via the Call_Bank code without normal termination (the error causes a RESTART 8 to be executed out of Home ROM code called from the ROM Extension). It shouldn't take anybody that many tries to get a tape command right, so this is not a real problem, but you may want to keep it in mind. For any call made through the OS RAM services, you should have a corresponding return to keep the structures clean.

6.6.2 ON ERR GOTO - If a non-existent line number is specified, followed by an error, the system will hang. The ROM code is in an endless loop trying to report the absence of a valid error handler to the non-existent error handler!!! On some errors, you will get an unexpected 0 OK termination showing the line number of your Error Handler. This is because some ROM routines temporarily clear the INTPT Flag (Bit 7 of FLAGS). This flag is set to 0 when checking syntax and set to 1 when executing; if an error is detected while the Flag=0, the error handler code is branched to but is not executed.

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

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

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

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

6.6.5 When DELETE (Cap Shift/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   0227
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   FUNCTS
ADD           33D3   SUMS
ALNUM?        3046   IDENT
ALPHA?        304B   IDENT
ANGLE         3B9E   FUNCTS
AROS          18C6   AROS
ARRAY         37C5   CALC
AR_LN         17EA   AROS
AR_NXT        17FF   AROS
ASN           3C4E   FUNCTS
ATN           3BFD   FUNCTS
ATTBYT        0710   IO_1
BEEP          0436   KSCAN
BORDER        243E   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
CLPR          0A35   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
CP_BC         16E8   LIST
CTRO          371A   CALC
DATA          1E82   SYNTAX
DEF           201D   SYNTAX
DELREC        1750   LIST
DELSYM        0B7E   IO_2
DEL_DE        174D   LIST
DEL_K         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          0C83   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
ERR5          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        283E   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
INCH          11E1   EDIT
ININT         30F9   INOUT
INIT          0D31   EDIT
INPUT         222B   SYNTWO
INSI          12B8   EDIT
INSA          0AE7   IO_2
INSERT        12BB   EDIT
INT           3ACA   FUNCTS
INTDIV        3ABB   FUNCTS
```

```
INTPT?      2389  EXPRN
IN_K        0C0E  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  SYNTWO
K_SCAN      02B0  KSCAN
LCU2        132D  EDIT
LDDE        313D  INOUT
LDMES       3CA8  TAPEMSG
LDTVCU      061A  IO_1
LE3         0055  BASIC
LED18       0E2F  EDIT
LED4        0E8D  EDIT
LET         2EBD  IDENT
LINENO      1768  LIST
LIST        14E1  LIST
LN          3B2E  FUNCTS
LPO         15AC  LIST
LS4         1A44  SYNTAX
LT22        1BBC  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  EXPRN
OPCHAN      1465  CHANS
OPEN        142A  CHANS
OPTNO       1C49  SYNTAX
OUTPUT      31A1  INOUT
PAEDCB      2E74  IDENT
PARP        03F3  KSCAN
PASSEM      25B9  SYNTWO
PAUSE       1FEB  SYNTAX
PHLAF       004F  BASIC
PLOT        2635  GRAPHS
PLOTBC      263E  GRAPHS
PLUGIN      0000  BASIC
POPSTR      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        1ED4  SYNTAX
RDCH        11CF  EDIT
READ        1D97  SYNTAX
RECLEN      1720  LIST
REMGSZ      12CA  EDIT
RESET       1354  EDIT
RESTBC      1ECA  SYNTAX
RETURN      1FD4  SYNTAX
RND         29B6  EXPRN
ROOM?       3768  CALC
ROOT        3C65  FUNCTS
RSET        2454  SYNTWO
RSTSTR      13A8  CHANS
R_ATTS      0888  IO_1
SCRL        0939  IO_1
SCRMBL      2603  GRAPHS
SEARCH      136B  EDIT
SELECT      1230  EDIT
SEL_HL      1248  EDIT
SENDCH      11ED  EDIT
SENDTV      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        1D28  SYNTAX
SKIPIT      2569  SYNTWO
SLICER      2E10  IDENT
SMINIT      11C1  EDIT
SOUND       2128  SYNTAX
SRCHSC      1374  EDIT
STBOOL      3926  CALC
STDE_S      314C  INOUT
STDE_U      314A  INOUT
STKUSN      3059  INOUT
STK_0       1C51  SYNTAX
STK_A       30E6  INOUT
STK_BC      30E9  INOUT
STK_M       3773  CALC
STOP        1C59  SYNTAX
STRITO      220F  SYNTWO
STTVCU      05F3  IO_1
SUB         33CE  SUMS
SUBLIN      16F0  LIST
SUBLN1      16F3  LIST
SUMSLD      3379  SUMS
SYNERR      1BED  SYNTAX
SYNTAX      1A27  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
XEY         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                                          ORIGIN   LENGTH
CALL_B        65D0   DISPATCH
CREATE        66E8   DISPATCH                  TABLES:    FIXTBL      1D00     007C
DISPAT        6200   DISPATCH
GET_CH        644D   DISPATCH                             JMPTBL      1E0C     0124
GET_NU        645E   DISPATCH
GET_ST        6405   DISPATCH                  UNUSED:                1624     06DC
GET_WO        6316   DISPATCH
GOTO_B        6572   DISPATCH                                          1D7C     0160
GOTO_E        6815   DISPATCH
INT           62AE   DISPATCH
```

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

                  420  *LIST ON
                  421  *LIST ON
                  422  *INCLUDE NEW_SYSVARS.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     80H
                  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     0FFFFH
                  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  2A805C    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 LN13:   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
63A1   D3F6      731                OUT     (SDATPT), A
63A3   F1        732                POP     AF
63A4   77        733                LD      (HL), A          ;RESTORE (HL)
63A5   F1        734                POP     AF
63A6   3200C0    735                LD      (LOWNYB), A      ;RESTORE (0E000H)
63A9   E1        736                POP     HL               ;RESTORE REGISTERS
63AA   C1        737                POP     BC
63AB   F1        738                POP     AF
63AC   C9        739                RET
                 740  ;
                 741  ;
                 742  ;  READ_BS_REG (LIN_ADDR: D; MON_ADDR: E; BYTE_DATA: E)
                 743  ;
                 744  ;
63AD   F5        745  READ_BS_REG    PUSH    AF               ;SAVE REGISTERS
63AE   C5        746                PUSH    BC
63AF   E5        747                PUSH    HL
63B0   62        748                LD      H, D
63B1   2E00      749                LD      L, 0             ;HL = MEMORY MAPPED ADDR
63B3   3A00C0    750                LD      A, (LOWNYB)      ;SAVE (0E000H)
63B6   F5        751                PUSH    AF
63B7   7E        752                LD      A, (HL)          ;SAVE (HL)
63B8   F5        753                PUSH    AF
63B9   3E07      754                LD      A, 7
63BB   D3F5      755                OUT     (SADDPT), A      ;SAVE VALUES OF SOUND REGS 7 AND E
63BD   DBF6      756                IN      A, (SDATPT)
63BF   47        757                LD      B, A
63C0   3E0E      758                LD      A, 0EH
63C2   D3F5      759                OUT     (SADDPT), A
63C4   DBF6      760                IN      A, (SDATPT)
63C6   4F        761                LD      C, A
63C7   C5        762                PUSH    BC
63C8   3E07      763                LD      A, 7             ;SET IOA CHANNEL TO OUTPUT
63CA   D3F5      764                OUT     (SADDPT), A
63CC   3E40      765                LD      A, 40H
63CE   D3F6      766                OUT     (SDATPT), A
63D0   3E0E      767                LD      A, 0EH
63D2   D3F5      768                OUT     (SADDPT), A
63D4   AF        769                XOR     A
63D5   D3F6      770                OUT     (SDATPT), A
63D7   3E02      771                LD      A, 2
63D9   3200C0    772                LD      (LOWNYB), A      ;RESET NYBBLE STEERING LOGIC
63DC   7E        773                LD      A, (HL)          ;READ LSN OF DATA
63DD   E60F      774                AND     0FH
63DF   4F        775                LD      C, A
63E0   63        776                LD      H, E
63E1   7E        777                LD      A, (HL)          ;READ MSN OF DATA
63E2   CB27      778                SLA     A
63E4   CB27      779                SLA     A
63E6   CB27      780                SLA     A
63E8   CB27      781                SLA     A
63EA   B1        782                OR      C
63EB   5F        783                LD      E, A             ;RETURN BYTE DATA IN E
63EC   C1        784                POP     BC
63ED   3E07      785                LD      A, 7             ;RESTORE SOUND REGS
63EF   D3F5      786                OUT     (SADDPT), A
63F1   78        787                LD      A, B
63F2   D3F6      788                OUT     (SDATPT), A
63F4   3E0E      789                LD      A, 0EH
63F6   D3F5      790                OUT     (SADDPT), A
63F8   79        791                LD      A, C
63F9   D3F6      792                OUT     (SDATPT), A
63FB   F1        793                POP     AF
63FC   77        794                LD      (HL), A          ;RESTORE (HL)
63FD   F1        795                POP     AF
63FE   3200C0    796                LD      (LOWNYB), A      ;RESTORE (0E000H)
6401   E1        797                POP     HL               ;RESTORE REGISTERS
6402   C1        798                POP     BC
6403   F1        799                POP     AF
6404   C9        800                RET
                 801  ;
                 802  ;
                 803  ;  GET_BANK_STATUS (BANK: B; STATUS : B, HORIZONTAL_SELECT: C)
                 804  ;
                 805  ;
6405   F5        806  GET_STATUS      PUSH    AF               ;SAVE SAVE REGS
6406   D5        807                PUSH    DE
6407   78        808                LD      A, B
6408   FEFE      809                CP      0FEH
640A   282E      810                JR      Z, GS_EXT        ;IF BANK = 254
640C   FEFF      811                CP      0FFH
640E   281D      812                JR      Z, GS_HOME       ;IF BANK = 255
6410   A7        813                AND     A
6411   281F      814                JR      Z, GS_DOC1       ;IF BANK = 0
6413   1680      815                LD      D, BNA           ;HERE IF EXP. BANK
6415   58        816                LD      E, B
6416   CD5C63    817                CALL    WRITE_BS_REG
6419   1640      818                LD      D, HS_LSN
641B   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_G
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   DS_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, GN_RD_DOCK   ;IF NO EXP. BANKS
646A  47          880                   LD      B, A
646B  58          881   GN_CHECK        LD      E, B            ;SEARCH ALL EXP. BANKS
646C  CD0564      882                   CALL    GET_STATUS
646F  A1          883                   AND     C
6470  2823        884                   JR      Z, GN_EXP       ;FOUND THE CHUNK, SO EXIT LOOP
6472  10F7        885                   DJNZ    GN_CHECK
6474  DBF4        886   GN_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, GN_DOCK
647A  0D          890                   DEC     C               ;IF CHUNK > 1, THEN CAN'T BE IN EXT.
647B  2011        891                   JR      NZ, GN_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, GN_HOME      ;NOT IN EXT. BANK
648A  3EFE        900                   LD      A, 0FEH         ;IN EXT. BANK, SO RETURN 254
648C  1808        901                   JR      GN_EXIT
648E  3EFF        902   GN_HOME LD      A, 0FFH         ;IN HOME BANK, SO RETURN 255
6490  1804        903                   JR      GN_EXIT
6492  AF          904   GN_DOCK         XOR     A               ;FOUND CHUNK IN DOCK, SO RETURN 0
6493  1801        905                   JR      GN_EXIT
6495  78          906   GN_EXP          LD      A, B            ;RETURN EXP. BANK NUMBER
6496  D1          907   GN_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      0FFH
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      0FEH
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  200C      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      0FFH
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     (DKHSPT1), 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   DD460B     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  CD4D04    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  CD4D04    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    HL
6747  DDE1        1335                    POP     IX              ;IX NOW POINTS TO PARMS
6749  DD6E06      1336                    LD      L, (IX+SRC_ADDR)
674C  DD6607      1337                    LD      H, (IX+SRC_ADDR+1)
674F  CD8866      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  CD8866      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  DDE3      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, I
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  2166        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+90H
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.OFFH=0
                                COND .NOT.(PRBUFF.AND.OFFH=0)
                                ERROR IN ASSERTION PRBUFF.AND.OFFH=0
                                ENDC
```

```
35  KSTATE: EQU PRBUFF+L_LEN*8      ; SEE KB DOCUMENTATION
36   KS_A:  EQU 0                   ;    1ST BYTE IS A CHAR KEY PRESSED
37   KS_C:  EQU 1                   ;    2ND IS TIME TILL COUNTS AS RELEASED
38   KS_B:  EQU 2                   ;    3RD IS TIME (IN FRAMES) TILL REPEAT
39   KS_D:  EQU 3                   ;    4TH IS CODE WHEN REPEATS
40                                  ;    5TH - 8TH ARE A SECOND SET OF 1ST FOUR
41  LAST_K: EQU KSTATE+8            ; NEWLY PRESSED KEY
42  REPDEL: EQU LAST_K+1            ; DELAY BEFORE 1ST REPEAT (INITIALIZED TO 35)
43  REPPER: EQU REPDEL+1            ; DELAY BEFORE SUBSEQUENT REPEATS (INITIALIZED TO 5)
44  DEFADD: EQU REPPER+1            ; -> CHAR AFTER '(' IN FORMAL PARAMETER LIST: MUST BE
45                                  ;    0 WHEN NO USER-DEFINED FN BEING EVALUATED
46  K_DATA: EQU DEFADD+2            ; DATA BYTE IN COMPOSITE CHAR FROM KEYBOARD
47  TVDATA: EQU K_DATA+1            ; USED FOR STORING BYTES IN COMPOSITE CHARACTERS:
48                                  ;    (TVDATA) = KEY BYTE,
49                                  ;    (TVDATA+1) = 1ST DATA BYTE FOR AT OR TAB.
50  STRMS:  EQU TVDATA+2            ; STREAM DATA: POINTERS (OFFSETS FROM (CHANS)-1) TO
51                                  ;    CHANNELS.  0 = STREAM NOT OPEN.
52   HIDSTR:EQU 3                   ; NO. STREAMS HIDDEN FROM USER.  THESE ARE TIED
53                                  ;    UNALTERABLY TO SPECIFIC CHANNELS.
54   HID_K: EQU -3                  ; KEYBOARD
55   HID_S: EQU -2                  ; TV, UPPER HALF OF SCREEN
56   HID_R: EQU -1                  ; INSERTION IN RAM
57   COM_ST:EQU 0                   ; STREAM FOR COMMANDS
58   INP_ST:EQU 1                   ; STREAM FOR INPUT DATA
59   PR_ST: EQU 2                   ; STREAM FOR PRINT
60   LPR_ST:EQU 3                   ; STREAM FOR LPRINT
61  CHARS:  EQU STRMS+(HIDSTR+16)*2 ; -> 8*20H BYTES BEFORE CHARACTER SET
62  FART:   EQU CHARS+2             ; NO. CYCLES OF ERROR NOISE (2 8VES BELOW MIDDLE C)
63  PIP:    EQU FART+1              ; NO. CYCLES OF KEYBOARD NOISE (3 8VES ABOVE MIDDLE C)
64  Y:      EQU PIP+1               ; VALUE ALWAYS HELD IN IY
65  ERR_NR  EQU Y                   ; [RUN TIME ERROR #] - 1
66  FLAGS:  EQU ERR_NR+1            ; VARIOUS FLAGS
67   SPC:   EQU 0                   ;    SUPPRESS SPACE BEFORE TOKENS
68   PR:    EQU 1                   ;    PRINTING TO PRINTER, NOT TV
69   LMODE1:EQU 2                   ;    L MODE, NOT K, AT CURRENT CHARACTER
70   LMODE: EQU 3                   ;    L MODE, NOT K, AT CURSOR
71   KEYHIT:EQU 5                   ;    KEYHIT FOUND
72   NO:    EQU 6                   ;    EXPRESSION IS NUMERICAL, NOT STRING
73   INTPT: EQU 7                   ;    REQ INTERPRET RATHER THAN CHECK SYNTAX
74  TVFLAG: EQU FLAGS+1             ; FLAGS ASSOCIATED WITH THE TV
75   LHS:   EQU 0                   ;    PRINTING TO LOWER HALF OF SCREEN
76   EDIT:  EQU 1                   ;    OUTPUTTING LINE FOR EDIT OR NO. FOR STRING
77   ECHREQ:EQU 3                   ;    ECHO REQUESTED IF INPUTTING FROM KEYBOARD
78   LIST:  EQU 4                   ;    OUTPUTTING AN AUTOMATIC LISTING
79   CLHS:  EQU 5                   ;    CLEAR LOWER HALF WHEN KEY PRESSED
80  ERR_SP: EQU TVFLAG+1            ;-> BOTTOM ITEM ON MACHINE STACK.
81  LISTSP: EQU ERR_SP+2            ; -> RETURN ADDRESS FROM AUTOMATIC LISTING
82  MODE:   EQU LISTSP+2            ; 0 = K OR L, 1 = F, 2 = G.
83  NEWPPC: EQU MODE+1              ; LINE TO BE JUMPED TO
84  NSPPC:  EQU NEWPPC+2            ; SUBLINE TO BE JUMPED TO (BIT 7 OFF FORCES JUMP)
85  PPC     EQU NSPPC+1             ; LINE # OF INSTR BEING INTERPRETED
86  SUBPPC: EQU PPC+2               ; NO. WITHIN LINE OF INSTR BEING INTERPRETED
87  BORDCR: EQU SUBPPC+1            ; BORDER COLOUR (SHIFTED LEFT BACKG BITS WITH 0S IN
88                                  ;    BITS 0-2 & 6-7)
89  E_PPC   EQU BORDCR+1            ; LINE # OF "CURRENT" LINE IN LISTING
90                                  ;THE VARIABLES FROM (VARS) UP TO & INCLUDING (STKEND)
91                                  ;    ARE 'MOVABLE' IN THE SENSE THAT THEY ARE ADJUSTED
92                                  ;    (BY REMOSZ IN MODULE EDIT) WHENEVER STUFF IS
93                                  ;    INSERTED IN OR DELETED FROM RAM.
94  VARS    EQU E_PPC+2             ; -> 1ST RECORD FOR A VARIABLE (LAST IS 1 BYTE 80H)
95  DEST    EQU VARS+2              ; -> VAR MATCHED BY TEMPL CODE 1 OR 4 (TEXT OR RECORD)
96  CHANS_: EQU DEST+2              ; -> CHANNEL DATA (INCLUDING FLOPPY BUFFERS).
97                                  ;    EACH ITEM COMPRISES:
98                                  ;    THE ADDRESS OF AN OUTPUT ROUTINE FOR WRCH,
99                                  ;    THE ADDRESS OF AN INPUT ROUTINE FOR INCH,
100                                 ;    A 1-BYTE CODE FOR THE DEVICE TYPE,
101                                 ;    &, WHERE APPROPRIATE, A FILE NAME, ADDITIONAL
102                                 ;    DATA & A BUFFER.
103 CURCHL: EQU CHANS_+2            ; -> DATA FOR CURRENT CHANNEL
104 PROG:   EQU CURCHL+2            ; -> BASIC PROGRAM
105 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-1

64 COLUMN MODE

TIMEX COMPUTER CORPORATION

APPLICATION DEVELOPMENT LIBRARY

Application Software Component 001

64-COLUMN MODE

Date: 12/15/83

ASC Number: 001

Version: 002

Author: Carol Corcoran

Name: 64 Column Mode Support

Description:

This component provides support to the application programmer for using the 64-column mode feature of the TS 2068. The services include opening/closing the second display file (saving the machine stack, OS RAM routines and BASIC structures), PRINT position control, attribute control, clear screen and scroll screen services and display of characters. For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and designated non-zero values for other conditions. The interface from BASIC is by means of POKE'ing the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chunk 7 (E000h/57344).

Version Control

Version	Description	Date
001	Original	11/28/83
002	ADD: 1) Std. and User-Defined Graphics to WRCHAR and GTCHAR. 2) WRSTRG (Write String)	12/15/83

- -

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

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

Free machine code: Register A
From BASIC: In VIMODE

Description:

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

Output: BC = 0 Successful
BC = 1 Invalid parameters (not equal to 0 or 64)
BC = 2 Not enough memory

- -

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

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

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

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

Description:

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

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

- -

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

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

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

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

Description:

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

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

- -

- -

Name: SETATT (SETATB from BASIC - parameter to ATTCTL)

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

From Machine Code: Register A
From BASIC: In ATTCTL

Description:

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

Output: BC = 0 Successful

- -

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

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

From Machine Code: Register A

From BASIC: MSKCTL

Description:

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

Output: BC = 0 for successful completion

- -

Name: WRCHR (WRCHB from BASIC - parameter to DATAB)

Input: Character code for character to be displayed

20H TO 7FH - Std. TS2068 Character Set
80H TO 8FH - Std. Graphics Set
90H TO A4H - User-Defined Graphics Set

From Machine Code: Register A

From BASIC: in DATAB

Description:

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

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

- -

- -

Name: WRSTRG (WRSTR3 from BASIC - String Identifier in PARAMS)

Input: Character Code String

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

From BASIC: String Variable Identifier in
System Variable PARAMS - 23747 (5CC3H)

Description:

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

NOTE: Characters within the string which are outside of the supported range (32 through 164 (20H-A4H)) will be ignored. E.g., BASIC Token codes and control codes embedded in an INPUT string will not be displayed or decoded.

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

```
000B  LET a$="------string-------"
0010  POKE 23747,CODE "a"
0015  IF USR (WRSTRG)<>0 THEN -----
         (continue)
```

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

- -

Name: SCRCLL (SCRLB from BASIC - parameters to SCRCTL)

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

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

Description:

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

Note: See Usage Section on "automatic" scrolling.

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

- -

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

Input: As for GTCHAR

Description:

Returns in register C of the BC register pair the attribute byte for the character at the designated screen position. Note that in 64 column mode the entire screen has common attributes. The value returned will describe the current selection:

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

- -

- -

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

Input: Line/Column position on for SETCUR

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

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

Description:

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

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

Output: BC = 0 for no find
BC = 1 invalid parameters
BC = character code (20H-A4H)

- -

Name: GETCUR (GETCUB from BASIC)

Input: None

Description:

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

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

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

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

- -

Usage:

Memory Usage:

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

Name	Size	Description
SCRCTL	2	Scroll Control LSB = Starting line number MSB = Number of lines to be scrolled
BOTLN	1	Bottom Line - Line number (0-23) after which test for scroll will be made.
SCRLCT	1	Scroll Count- Number of times + 1 that automatic scroll will be done. When decremented to zero, data will not be displayed and a condition code will be returned to user.
CHTBL	2	Character Table (Base Address-100H)
GRTBL	2	Std.Graphics Character Table (Base-10DH)
LINLEN	1	Line Length - (64 when in 64-Col.Mode)
CURPOS	2	Current Position (Internal Format) LSB = Column Position MSB = Line Position
DFADDR	2	Current Display File Address
MASKB	1	Mask Byte (bit 0 = OVER) (bit 2 = INVERSE)

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

GTINDX          1       'Get' Index - Used by GTCHAR
STRGCT          2       String Count -  Contains remaining byte count
;                                       when BC=0 (Screen Full) is
;                                       returned from the Write String
;                                       service WRSTRG (&RSTRB).
```

- -

Initial values set via SETMODE (SETMDE) are as follows:

Variable Name	Value
SCRCTL	1701H
BOTLN	17H
SCRLCT	1H
CMTBL	3C00H
GRTBL	(Internal to Module)
LINLEN	40H
CURPOS	1841H
OFADDR	4D00H
MASKE	0H
ATTBYT	38H
GTINDX	80H
STRGCT	0H

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

Variable Name	Size	Value
* DATAB	1	0H
* LINCOL	2	0H
* CLRCTL	2	1800H
* ATTCTL	1	38H
* MSKCTL	1	0H
* GETCTL	2	0H
VIMODE	1	

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

Location:

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

The routine SETMODE (SETMDB) cannot be executed from a cartridge because of the necessity to enable the ROM Extension which disables the DOCK Bank.

Registers:

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

Automatic Scrolling:

As initialized, test for scrolling will be made when the print position goes to the next line following the bottom line on screen (BOTLN=23=24th line). Condition Code 3 (Screen Full) will be returned since SCRLCT will decrement to zero. If SCRLCT is set to some larger value, then the parameters in SCRCTL will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a POKE or setting the variables BOTLN and SCRCTL to the desired values, automatic scrolling can be done using smaller sections of the screen. When working from BASIC it is recommended that BOTLN be set to Line 21 (15H) and SCRCTL+1 be set to 21 (15H) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Note that once SCRLCT expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the 'Screen Full' condition.

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

NOTES:

1. All screen operations done by the system ROM (PRINT, LIST, Edit line I/O, CLS, scrolling, etc.) relate only to the main Display File. This means that only the even columns on the 64-column mode screen will be affected. You will want to execute the Clear Screen function in this module to guarantee that no data in the second display file interferes with the use of a system screen service, e.g. prior to doing a LIST.

2. The COPY command will print only the even columns of the screen to the 2040 Printer.

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

4. The SAVE filename SCREEN$ will save only the main Display File data. The second can be saved by a SAVE filename CODE 24576, 6144. Be careful that you have the computer in 64-column mode when you load this data or you will overwrite the OS RAM routines and "crash" the system!! (The count for saving the display file is for the data portion only since the Attribute File area from 7800H-7AFFH (30720-31487) is not used by the video mode H/W in 64-Column mode.

- -

```
ADL - ASC 001 64-COL.MODE SUPPORT       CROSS/Z8 version 10.36.1A      18-Mar-84  15:8:15
           IO64.SRC

                         NAME ADL - ASC 001 64-COL.MODE SUPPORT
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;          ************************************
                  ;          *                                  *
                  ;          *              TIMEX               *
                  ;          * APPLICATION DEVELOPMENT LIBRARY  *
                  ;          *                                  *
                  ;          * NAME: 64-COL.MODE SUPPORT        *
                  ;          *                                  *
                  ;          * ASC NO: 001                      *
                  ;          * VERSION: 002                     *
                  ;          * AUTHOR: C. CORCORAN              *
                  ;          * DATE:    12/15/83                *
                  ;          ************************************
                         SUBTTL VERSION LEVEL CONTROL
                  ;
                  ;
                  ;      VERSION        DATE          COMMENTS
                  ;      -------        ----          --------
                  ;
                  ;        001          11/25/83      ORIGINAL
                  ;
                  ;        002          12/15/83      ADD STANDARD AND USER-DEFINED
                  ;                                   GRAPHICS TO WRCHAR AND GTCHAR
                  ;
                  ;                                   ADD WRSTRG (WRITE STRING)
                  ;                                   CAPABILITY
                         SUBTTL DEFINITIONS
                  ;
                  ;
                  ;
                  ;                     ****DEFINITIONS****
                  ;
                         INTERN  WRCHAR,WRITAG,SETCUR,SETATT,SETMOD,CLRSCN,SCROLL
                         INTERN  GTCHAR,GETATT,GETCUR
                         INTERN  WRCHAB,WRITSA,SETCUB,SETATB,SETMOB,CLRSCB,SCRLB
                         INTERN  GTCHAB,GETATB,GETCUB
                         INTERN  GETMODE,GETMOB
                  ;
=001E             SCRSZ        EQU     24       ; 24 LINES
=1701             SCINIT       EQU     1701H    ; SCROLL CONTROL INITIALIZATION
=1800             CLINIT       EQU     1800H    ; CLEAR SCREEN CTL. INIT.
=3C00             CHRSET       EQU     3C00H    ; TOP CHAR.TABLE-100H
=0195'            GRPHST       EQU     ((GRPST)-150H) ;STD.GRAPHICS CHARS.
                  ;
=0SSE             CHRCVID      EQU     0SSEH    ; ROUTINE IN ROM EXTENSION
=1720             RECLEN       EQU     1720H    ; ROUTINE IN HOME ROM
                  ;
=00FF             HRSLPT       EQU     0FFH     ; HOME ROM EXT.SELECT PORT (BIT 7)
=00F4             DOHSPT       EQU     0F4H     ; DOCK HORIZONTAL SELECT PORT
                  ;
=5C48             VARS         EQU     5C48H    ; SYSTEM VARIABLES
=5C65             STKEND       EQU     5C65H    ;
=5C7B             UDG          EQU     5C7BH    ;
=5C7D             COORDS       EQU     5C7DH    ;
=5C82             RAMTOP       EQU     5C82H    ;
=5C5A             PRONT        EQU     5C5AH    ;
=5CC1             VIDMOD       EQU     5CC1H    ;
=5CC3             PASOMS       EQU     5CC3H    ; (LOCATION 23747)
```

```
=6840                 DRIVES        EQU    A840H    ;
=12C0                 INSDATSZ      EQU    7800H-DRIVES
=0840                 MOVESZ        EQU    DRIVES-A000H
=F7C0                 DESTT         EQU    0FFFFH-MOVESZ+1
=97C0                 PIA           EQU    DESTT-6000H

=1008                 PIXTBL        EQU    1008H


                                SUBTTL INPUTS AND ENTRIES
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;                   **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                      ;
                      ;                                                              WITH
                      ;                                                       MODULE LOADED
                      ;                                                       AT 57344(E000H)
                      ;                                                       DECIMAL ADDRESS
                      ;                                                       ---------------
0000'  20             DATAB         DEFB   20H      ; DATA BYTE FROM BASIC          57344
0001'  C3 0030'       WRCHR01       JP     WRCH0    ; ENTRY TO WRITE CHARACTER      57345
000A'  C3 000C'       WRSTR01       JP     WRSTR0   ; ENTRY TO WRITE STRING         57348
                      ;                             ; (see PARAMS above)
0007'  0000           LINCOL        DEFW   0        ; SET CURSOR PARAMETERS
                                                    ;   LINCOL=COL. (0-63)          57351
                                                    ;   LINCOL+1=LINE (0-23)        57352
0009'  C3 0135'       SETCUR01      JP     SETCR0   ; ENTRY TO SET CURSOR POSN.     57353
000C'  1800           CLRCTL        DEFW   1800H    ; CLEAR SCREEN CONTROLS
                                                    ;   CLRCTL=STARTING LINE NO.    5735A
                                                    ;   CLRCTL+1=LINE COUNT         57357
0008'  C3 01F3'       CLRSC01       JP     CLS100   ; ENTRY TO CLEAR SCREEN         57358
0031'  58             ATTCTL        DEFB   58H      ; ATTRIBUTE CONTROL             57561
                                                    ;   BIT 7=FLASH (0 IN 64-COL.)
                                                    ;       6=BRIGHT(0 IN 64-COL.)
                                                    ;       5
                                                    ;       4=PAPER COLOR
                                                    ;       3
                                                    ;       2
                                                    ;       1= INK COLOR
                                                    ;       0
0012'  C3 0260'       SETAT01       JP     SETAB0   ; ENTRY TO SET ATTRIBUTES       573A2
0015'  00             MSKCTL        DEFB   0        ; MASK CONTROL FROM BASIC:      573A5
                                                    ;   BIT 2=INVERSE
                                                    ;       0=OVER
0016'  C3 0290'       SETMSR1       JP     SETMS0   ; ENTRY TO SET MASK CONTROLS    575A6
0019'  0000           GETCTL        DEFW   0        ; "GET" CONTROL FROM BASIC      573A9
                                                    ; FORMAT IS AS FOR LINCOL
0018'  C3 02A1'       GTCHR01       JP     GTCH00   ; ENTRY TO GET CHARACTER        57371
0030'  C3 0321'       GETAT01       JP     GETAB0   ; ENTRY TO GET ATTRIBUTE        57574
0021'  C3 0320'       GETCUR1       JP     GETCR0   ; ENTRY TO GET CURSOR POSN.     57377
002A'  00             VIMODE        DEFB   0        ; VIDEO MODE CONTROL            57380
                                                    ;       0=NORMAL
                                                    ;       6=64-COL.MODE
0025'  C3 0345'       SETMOD1       JP     STMOD0   ; ENTRY TO SET VIDEO MODE       57581
0028'  1701           SCRCTL        DEFW   1701H    ; SCROLL CONTROL
                                                    ;   SCRCTL=STARTING LINE NO.    5730A
                                                    ;   SCRCTL+1=LINE COUNT         57565
0028'  C3 0145'       SCRL01        JP     SCRL00   ; ENTRY TO SCROLL               5730A
                                SUBTTL  OTHER VARIABLES
                      ;
                      ;
                      ;
                      ;
0020'  17             BOTLN         DEFB   17H      ; BOTTOM LINE (LINE AFTER       57389
                                                    ; WHICH SCREEN IS CONSIDERED
                                                    ; FULL.  AUTOMATIC SCROLL
                                                    ; WILL BE DONE IF SCRLCT
                                                    ; DOES NOT DECREMENT TO 0.
002E'  01             SCRLCS        DEFB   1        ; SCROLL COUNT - 1 PLUS NO.     57390
                                                    ; OF TIMES AUTOMATIC SCROLL
                                                    ; WILL BE DONE.
                      ;
                      ;
                      ;
                      ;
002F'  5C08           CHTBL         DEFW   CHRSET   ; ADDRESS OF CHARACTER TABLE    57391
                                                    ; (DEFAULTS TO TS2068 SYSTEM
                                                    ;  ROM)
0031'  0595'          GRTBL         DEFW   GRPHST   ; ADDRESS OF STD.GRAPHICS TBL.57393
0033'  40             LINLEN        DEFB   40H      ; LINE LENGTH (64)              57339
003A'  10A3           CURPOS        DEFW   1041H    ; CURRENT POSITION (INTERNAL    57396
                                                    ;  FORMAT)
0036'  4000           DFADRS        DEFW   4000H    ; CURRENT DISPLAY FILE ADDRS.   57398
0038'  00             MASKB         DEFB   0        ; MASK TO BE APPLIED TO         57400
                                                    ; DISPLAY CHARACTERS
0039'  38             ATTBYT        DEFB   38H      ; CURRENT ATTRIBUTES            57401
                                                    ;  (BLACK ON WHITE)
0036'  80             GTINDX        DEFB   80H      ; INDEX FOR ADJUSTING           57402
                                                    ; CH.CODE IN GTCHAR
0038'  0000           STRGCT        DEFW   0        ; REMAINING COUNT FROM WRSTRG
                      ;                             ; WHEN "SCREEN FULL"            57403
                                SUBTTL  WRITE CHARACTER
                      ;
                      ;
0030'                 WRCH01                        ; HERE FROM BASIC ENTRY TO DISPLAY THE
                                                    ; CHARACTER WHOSE CODE IS IN "DATAB"
0030'  3A 0000'                     LD     A,(DATAB)
                      ;
0040'                 WRCHAR1                       ; ENTRY WITH CODE IN A
0040'  CD 0AE0'                     CALL   LDPOSN   ; LOAD REGISTERS FOR CURRENT POSITION
                                                    ;   BC=CURSOR POSITION FROM "CURPOS"
                                                    ;   HL=DISPLAY FILE ADDRESS FROM "DFADRS"
                                                    ;   A=CODE FOR CHAR. TO BE DISPLAYED
0045'  FE 20                        CP     20H      ; TEST VALID RANGE
0045'  DA 00C2'                     JP     C,INVPAR ; < 20H
0048'  FE A5                        CP     0A5H     ;
004A'  D2 00C2'                     JP     NC,INVPAR ; > A4H
004D'  C5                           PUSH   BC       ; SAVE CURSOR POSITION
004E'  FE 80                        CP     80H      ; TEST IF ASCII PRINTABLE (20H-7FH)
0050'  38 13                        JR     C,WRCH12 ; YES
```

```
0052'  FE 90                 CP      90H              ; TEST IF STD.GRAPHICS(80H-8FH)
0054'  38 09                 JR      C,WRCH11
0056'  ED 4B 5C7B            LD      BC,(UDG)         ; USER-DEFINED GRAPHICS(90H-A4H)
005A'  05                    DEC     B                ; ADJUST ADDRESS-100H
005B'  D6 70                 SUB     70H              ; ADJUST FOR INDEX INTO TABLE
005D'  18 0C                 JR      WRCH13
005F'  ED 4B 0031'   WRCH11  LD      BC,(GRTBL)       ; ADDRESS OF GRAPHICS TABLE
0063'  D6 80                 SUB     80H              ; ADJUST FOR INDEX INTO TABLE
0065'  18 04                 JR      WRCH13
0067'  ED 4B 002F'   WRCH12  LD      BC,(CHTBL)       ; CHARACTER TABLE
006B'  EB            WRCH13  EX      DE,HL            ; DE=POSN.IN DISPLAY FILE
006C'  26 00                 LD      H,0
006E'  6F                    LD      L,A              ; CODE IS INDEX INTO TABLE
006F'  29                    ADD     HL,HL
0070'  29                    ADD     HL,HL
0071'  29                    ADD     HL,HL
0072'  09                    ADD     HL,BC            ; HL=PIXEL PATTERN IN TABLE
0073'  C1                    POP     BC
0074'  EB                    EX      DE,HL
0075'  79                    LD      A,C              ; TEST IF END OF LINE
0076'  3D                    DEC     A
0077'  3A 0033'              LD      A,(LINLEN)
007A'  20 05                 JR      NZ,WRCH14
007C'  3C                    INC     A                ; START OF NEW LINE
007D'  05                    DEC     B                ; BUMP LINE NO.
007E'  4F                    LD      C,A              ; LINLEN+1=START OF LINE
007F'  18 01                 JR      WRCH15
0081'  3C            WRCH14  INC     A
0082'  B9            WRCH15  CP      C                ; TEST IF START OF LINE
0083'  D5                    PUSH    DE
0084'  CC 00C6'              CALL    Z,TVFULQ         ; TEST IF OFF SCREEN
0087'  D1                    POP     DE
                                                      ; HERE TO PUT CHARACTER IN DISPLAY FILE
0088'  C5            WRCH2   PUSH    BC               ; CURSOR POSITION
0089'  E5                    PUSH    HL               ; DISPLAY FILE ADRS.
008A'  3A 0030'              LD      A,(MASKB)
008D'  06 FF                 LD      B,-1
008F'  1F                    RRA
0090'  38 01                 JR      C,WRCH3          ; IF ORING NEW INTO OLD
0092'  04                    INC     B                ; B=-1 IF ORING =0 IF NOT
0093'  1F            WRCH3   RRA
0094'  1F                    RRA
0095'  9F                    SBC     A,A
0096'  4F                    LD      C,A              ; C=-1 FOR INVERSE =0 IF NOT
0097'  3E 08                 LD      A,8              ; SCAN COUNT
0099'  A7                    AND     A
009A'  EB                    EX      DE,HL
009B'  08            WRCH5   EX      AF,AF'
009C'  1A                    LD      A,(DE)           ; OLD BIT ROW
009D'  A0                    AND     B
009E'  AE                    XOR     (HL)
009F'  A9                    XOR     C                ; INVERT IF C=-1
00A0'  12                    LD      (DE),A           ; UPDATE DISPLAY FILE
00A1'  08                    EX      AF,AF'
00A2'  14                    INC     D                ; NEXT SCAN
00A3'  23                    INC     HL               ; NEXT BYTE IN CHAR.FILE
00A4'  3D                    DEC     A                ; TEST IF MORE SCANS
00A5'  20 F4                 JR      NZ,WRCH5
00A7'  E1            WRCH7   POP     HL
00A8'  C1                    POP     BC
00A9'  0D                    DEC     C                ; ADJUST CURSOR POSITION
00AA'  79                    LD      A,C              ; TEST IF EVEN/ODD
00AB'  CB 47                 BIT     0,A
00AD'  20 06                 JR      NZ,CCLODD
00AF'  7C            CCLEVN  LD      A,H
00B0'  F6 20                 OR      20H
00B2'  67                    LD      H,A
00B3'  18 05                 JR      WRCHRT
00B5'  7C            CCLODD  LD      A,H
00B6'  E6 DF                 AND     0DFH
00B8'  67                    LD      H,A
00B9'  23                    INC     HL               ; GPS NEXT BYTE
00BA'  CD 0465'      WRCHRT  CALL    STPOSN           ; STORE NEW POSITION
00BD'  0E 00         GOODRET LD      C,0              ; RETURN BC=0
00BF'  06 00         ERRRET  LD      B,0              ; ENTER HERE WITH C NON-ZERO
00C1'  C9                    RET
                     ;
00C2'  0E 01         INVPAR  LD      C,1              ; RETURN BC=1 FOR INVALID PARAMETERS
00C4'  18 F9                 JR      ERRRET
                     ;
00C6'  3E 18         TVFULQ  LD      A,18H            ; TEST IF NEW LINE 23 OFF SCREEN
00C8'  90                    SUB     B                ; A=LINE NO.
00C9'  57                    LD      D,A              ; SAVE IN D
00CA'  3A 002D'              LD      A,(BOTLN)        ; GET BOTTOM LINE
00CD'  BA                    CP      D
00CE'  D2 0A14'              JP      NC,UPDPOSN       ; ON SCREEN - RETURN TO WRITE CHAR.
                                                      ;  VIA UPDATE AND STORE POSITION
00D1'  3A 002E'              LD      A,(SCROLCT)      ; TEST SCROLL COUNT
00D4'  3D                    DEC     A
00D5'  20 0B                 JR      NZ,TVFUL1        ; DO AUTOMATIC SCROLL USING SCRCTL
00D7'  3E 01                 LD      A,1              ; REINITIALIZE TO 1
00D9'  32 002E'              LD      (SCRLCT),A
00DC'  C1                    POP     BC
00DD'  C1                    POP     BC               ; DISCARD RETURN TO WRITE CHAR.
00DE'  0E 03                 LD      C,3              ; RETURN BC=3 FOR SCREEN FULL
00E0'  18 DD                 JR      ERRRET
00E2'  32 002E'      TVFUL1  LD      (SCRLCT),A       ; UPDATE VARIABLE
00E5'  ED 4B 002D'           LD      BC,(SCRCTL)      ; GET SCROLL CONTROLS (STARTING LINE
                                                      ;   AND LINE COUNT)
00E9'  C3 053F'              JP      SCROLL           ; RETURN TO WRITE CHAR. VIA SCROLL
                     ;
                             SUBTTL WRITE STRING
                     ;                                ; HERE FROM BASIC ENTRY TO
                     ;                                ; WRITE CHARACTER STRING TO SCREEN.
                     ;                                ; STRING IDENTIFIER (CH.CODE) IN
                     ;                                ; SYSTEM VARIABLE PARAMA AT 5CC3H
00EC'  3A 5CC3       WRSTRG  LD      A,(PARAMA)       ; GET STRING ID
00EF'  E6 1F                 AND     1FH              ; MASK OFF UPPER BITS
00F1'  F6 40                 OR      40H              ; STRING VARIABLE IDENTIFIER
00F3'  57                    LD      D,A              ; SAVE IN D
00F4'  2A 5C4B               LD      HL,(VARS)        ; FIND STRING
00F7'  7E            NXSTRS  LD      A,(HL)           ;
```

```
00FE'  EA 7F          299           AND   07FH              ; TEST IF END OF VARS AREA
00FA'  2A 35          300           JR    Z,NOSTRG          ; NO FIND - RETURN BC=2
00FC'  BA             301           CP    D                 ; TEST IF MATCH
00FD'  2B 0E          302           JR    1,WRSTR2          ; FOUND STRING
00FF'  D5             303           PUSH  DE                ; SAVE STRING ID
0100'  CC 1720        304           CALL  RECLEN            ; RETURNS ADRS. OF NEXT VAR, IN DE
0103'  EE             305           XR    DE,HL             ; ADRS. TO HL
0104'  01             306           POP   DE                ; RESTORE STRING ID
0105'  18 F0          307           JR    WRSTR1            ; LOOK AGAIN
0107'  23             308  WRSTR2   INC   HL                ; GET LENGTH
0108'  4E             309           LD    C,(HL)
0109'  23             310           INC   HL
010A'  46             311           LD    B,(HL)
010B'  23             312           INC   HL                ; FIRST TEXT CODE
010C'  7E             313           LD    A,B               ; TEST IF NULL STRING
010D'  B1             314           OR    C
010E'  C8             315           RET   Z                 ; RETURN IF SO
                      316  ;
                      317  ;
                      318  ;                               ; ENTRY FROM MACHINE CODE WITH
                      319  ;                               ; ADDRESS OF CHAR.CODE "STRING"
                      320  ;                               ; IN HL AND LENGTH IN BC
                      321  ;
010F'  7E             322  WRSTRG  LD    A,(HL)            ; GET CODE
0110'  E5             323           PUSH  HL                ; SAVE ADDRESS
0111'  C5             324           PUSH  BC                ; AND COUNT
0112'  CD 0040'       325           CALL  WRCHAR            ; WRITE "A"
0115'  79             326           LD    A,C               ; TEST IF GOOD
0116'  B0             327           OR    B
0117'  20 09          328           JR    NZ,WRITX1         ; EXIT IF INVALID CODE OR
                      329  ;                               ; IF SCREEN FULL
0119'  C1             330  WRSTRS   POP   BC                ; COUNT
011A'  E1             331           POP   HL                ; ADDRESS
011B'  23             332           INC   HL                ; NEXT CHAR.
011C'  0B             333           DEC   BC                ; ADJUST COUNT
011D'  78             334           LD    A,B               ; TEST IF DONE
011E'  B1             335           OR    C
011F'  20 EE          336           JR    NZ,WRSTRG         ; WRITE NEXT CHAR.
0121'  C9             337           RET                     ; RETURN BC=0
                      338  ;
0122'  79             339  WRSTR1  LD    A,C               ; SAVE ERROR
0123'  FE 05          340           CP    5                 ; TEST IF UNRECOGNIZED CODE
0125'  2E F2          341           JR    Z,WRSTR3          ; CONTINUE
0127'  E1             342           POP   HL                ; REMAINING COUNT TO HL
0128'  22 0038'       343           LD    (STRGC7),HL       ; STORE REMAINING COUNT
012B'  E1             344           POP   HL                ; ADDRESS TO HL
012C'  0E 03          345           LD    C,3               ; RETURN EC=3
012E'  0A 00          346  WRSTR2  LD    B,0
0130'  C9             347           RET
                      348  ;
0131'  0E 02          349  NOSTRG  LD    C,2               ; RETURN BC=2 IF STRING NOT FOUND
0133'  18 F9          350           JR    WRSTR2
                      351  ;
                      352           SUBTTL POSITION CONTROL
                      353  ;
                      354  ;
0135'                 355  SSTCR0:                         ; ENTRY FROM BASIC ENTRY.  PARAMETERS
                      356                                  ; IN LINCOL
0135'  ED 4B 0007'    357           LD    BC,(LINCOL)
                      358  ;
0139'                 359  SSTCUR:                         ; ENTRY WITH LINE/COL. IN BC REG.
0139'  CD 043A'       360           CALL  TSTPAR            ; TEST PARAMETERS FOR VALIDITY
013C'  CD 0448'       361           CALL  CONVPP            ; CONVERT TO INTERNAL FORMAT
013F'  CD 045A'       362           CALL  UPDPOSN           ; CALCULATE AND STORE POSITION
0142'  C3 00A0'       363           JP    GOODRET           ; RETURN BC=0
                      364  ;
                      365           SUBTTL SCROLL
                      366  ;
                      367  ;
                      368  ;
0143'                 369  SCRL80:                         ; HERE FROM BASIC ENTRY TO SCROLL
                      370  ;                               ; SCREEN
0143'  ED 4B 0030'    371           LD    BC,(SCRCTLS       ; GET CONTROL INFO.
                      372  ;
0147'                 373  SCROLL:                         ; HERE WITH CONTROLS IN BC
                      374  ;                               ;  B=NO. OF LINES
                      375  ;                               ;  C=STARTING LINE NO.
0147'  78             376           LD    A,B               ; TEST VALIDITY
0148'  67             377           AND   A
0149'  CA 00C3'       378           JP    Z,INVPAR          ; ERROR IF COUNT=0
014C'  81             379           ADD   A,C
014D'  FE 01          380           CP    1
014F'  DA 00C3'       381           JP    C,INVPAR          ; ERROR IF B+C<1
0152'  FE 19          382           CP    25
0154'  D2 00C2'       383           JP    NC,INVPAR         ; ERROR IF B+C>24
0157'  CD 015F'       384           CALL  SCRL0             ; DO SCROLL
015A'  C3 00A0'       385           JP    GOODRET           ; RETURN BC=0
                      386  ;
015F'  C5             387  SCRL0   PUSH  BC
0160'  3A 0031'       388           LD    A,(SCRSI          ; GET STARTING LINE IN INTERNAL FORMAT
0163'  81             389           SUB   C
0164'  47             390           LD    B,A
0165'  3A 003E'       391           LD    A,(CLSNLEN)
0168'  3C             392           INC   A                 ; COL. 0
0169'  4F             393           LD    C,A
016A'  CD 0474'       394           CALL  LNBU
016D'  C1             395           POP   BC
016E'  C5             396           PUSH  BC                ; SAVE ORIG.BC FOR SRST
016F'  7C             397  SSTAC   LD    A,H               ; TEST IF AT START OF BLOCK
0170'  A7             398           AND   A
0171'  28 35          399           JR    Z,STBLK           ; YES
0173'  07             400           RLCA
0174'  07             401           RLCA
0175'  07             402           RLCA
0176'  4F             403           LD    C,A               ; GET NO. OF LINES THIS BLOCK
0177'  3E 00          404           LD    A,0
0179'  91             405           SUB   C
017A'  0E             406           CP    B                 ; TEST IF LINES REMAIN TOTAL LINES
017B'  3E 3F          407           JR    C,GRBLK           ; TOTAL GREATER THAN THIS BLOCK
017D'  9E             408  INSDBLK LD    A,B
017E'  0A 0E          409           LD    B,0               ; REMAINING COUNT
```

```
017F'  C9                  411          PUSH  BC
0180'  47                  412  LOOP    LD    B,A              ; B=LINES THIS BLOCK
0181'  81 0E               413          LD    C,A              ; C=SCAN COUNT
0183'  7B                  414  LOOP0   LD    A,E
0184'  C5                  415          PUSH  AC               ; SAVE SCAN COUNT
0185'  0F                  416          RRCA
0186'  0F                  417          RRCA
0187'  0F                  418          RRCA                   ; CALCULATE NO. OF BYTES
0188'  6F                  419          LD    C,A
0189'  3A 00               420          LD    B,0              ; BC=32OR OF LINES
018A'  EE                  421  LOOP1   EX    DE,HL
018C'  21 FF30             422          LD    HL,-20H          ; GET ADRS. OF PREV. LINE
018F'  19                  423          ADD   HL,DE
0190'  00                  424          EX    DE,HL            ; TO DE
0191'  C5                  425          PUSH  BC       ; SAVE COUNT
0192'  E5                  426          PUSH  HL       ; SAVE ADDRESS
0193'  ED B0               427          LDIR           ; DO MOVE
0195'  E1                  428          POP   HL
0196'  C1                  429          POP   BC
0197'  7C                  430          LD    A,H
0198'  CA 0F               431          BIT   3,A              ; TEST IF OP3
019A'  20 05               432          JR    NZ,NOTSC         ; NO
019C'  F6 20               433          OR    20H
019E'  67                  434          LD    H,A              ; DO OP2
019F'  18 0A               435          JR    LOOP1
01A1'  E6 0F               436  NOTSC   AND   00FH            ; BACK TO OP1
01A3'  47                  437          LD    H,A
01A4'  2A                  438          INC   H                ; NEXT SCAN LINE
01A5'  C1                  439          POP   BC               ; SCAN COUNT
01A6'  0D                  440          DEC   C
01A7'  20 D4               441          JR    NZ,LOOP0         ; MORE SCANS
01A9'  C1                  442          POP   BC               ; RESIDUAL LINE COUNT
01AA'  78                  443          LD    A,B
01AB'  A7                  444          AND   A
01AC'  28 04               445          JR    Z,SCRLRT         ; DONE
01AE'  26 00               446          LD    L,0              ; START OF NEXT BLOCK
01B0'  18 BC               447          JR    TESTAD           ; DO REMAINING LINE(S)
01B2'  C1                  448  SCRLRT  POP   BC               ; GR16.BC
01B3'  79                  449          LD    A,C              ; STARTING LINE
01B4'  ED                  450          NEG   A,B              ; ADD NO. OF LINES MOVED
01B5'  3D                  451          DEC   A                ; LAST LINE SCROLLED
01B6'  4F                  452          LD    C,A
01B7'  06 01               453          LD    B,1              ; CLEAR 1 LINE
01B9'  18 92               454          JR    CLR0C0           ; CLEAR VACATED LINE AND RETURN
                           455  ;
01BB'  4F                  456  BRBLK   LD    C,A
01BC'  78                  457          LD    A,B
01BD'  41                  458          SUB   C                ; ADJUST TOTAL LINE COUNT BY NO.
01BE'  47                  459          LD    B,A              ; OF LINES THIS BLOCK
01BF'  C5                  460          PUSH  BC
01C0'  78                  461          LD    A,C              ; LOAD A NO. OF LINES THIS BLOCK
01C1'  18 80               462          JR    LOOP
                           463  ;
01C3'  05                  464  STBLK   DEC   B                ; ADJUST TOTAL LINE COUNT-1
01C4'  C5                  465          PUSH  BC
01C5'  0E 00               466          LD    C,0              ; SCAN COUNT
01C7'  C0                  467  STBLR0  PUSH  BC               ; SAVE SCAN COUNT
01C8'  58                  468  STALR1  EX    DE,HL
01C9'  21 F8D0             469          LD    HL,-730H
01CC'  19                  470          ADD   HL,DE
01CD'  58                  471          EX    DE,HL            ; ADRS. OF PREV. LINE
01CE'  01 3020             472          LD    BC,32            ; BYTE COUNT
01D1'  E5                  473          PUSH  HL
01D2'  ED B0               474          LDIR                   ; DO MOVE
01D4'  E1                  475          POP   HL
01D5'  7C                  476          LD    A,H
01D6'  CB 6F               477          BIT   5,A
01D8'  20 05               478          JR    NZ,STBLR2
01DA'  F6 20               479          OR    20H              ; DO OP2
01DC'  67                  480          LD    H,A
01DD'  18 E8               481          JR    STBLR1
01DF'  E6 DF               482  STBLR2  AND   0DFH            ; BACK TO OP1
01E1'  67                  483          LD    H,A
01E2'  2A                  484          INC   H                ; TO NEXT SCAN LINE
01E3'  C1                  485          POP   BC
01E4'  0D                  486          DEC   C                ; TEST IF MORE SCANS
01E5'  20 00               487          JR    NZ,STBLR0
01E7'  C1                  488          POP   BC               ; REMAINING LINE COUNT
01E8'  78                  489          LD    A,B
01E9'  A7                  490          AND   A
01EA'  28 CB               491          JR    Z,SCRLAT
01EC'  11 F820             492          LD    DE,-7E0H         ; ADJUST TO LINE 1
01EF'  19                  493          ADD   HL,DE
01F0'  C3 0146'            494          JP    TESTAD           ; BC=LINE COUNT
                           495                                 ; HL=OP ADDRESS
                           496  ;
                           497          SUBTTL  CLEAR SCREEN
                           498  ;
                           499  ;
01F3'                      500  CLRSBR  ;                      ; MORE FROM BASIC ENTRY TO CLEAR SCREEN
01F3'  0C 48 000C'         501          LD    BC,(CCLRCTL)     ; GET CONTROL INFO.
                           502  ;
01F7'                      503  CLRSCN  ;                      ; ENTRY WITH BC=LINE COUNT AND
                           504                                 ; STARTING LINE NO. FOR CLEAR
01F7'  78                  505          LD    A,B
01F8'  A7                  506          AND   A                ; TEST VALIDITY
01F9'  CA 00C2'            507          JP    Z,INVPAR         ; ERROR IF COUNT=0
01FC'  81                  508          ADD   A,C
01FD'  F8 01               509          CP    1
01FF'  DA 00C2'            510          JP    C,INVPAR         ; ERROR IF B=C<1
0202'  F5 18               511          CP    25
0204'  D2 00C2'            512          JP    NC,INVPAR        ; ERROR IF B=C>24
0207'  CC 0200'            513          CALL  CLRSC0           ; DO SERVICE
020A'  C3 00E0'            514          JP    GOODRSY          ; RETURN AC=0
                           515  ;
                           516  ;
0200'  C5                  517  CLRSC0  PUSH  BC
0201'  55 10               518          LD    A,SCR32          ; CONVERT TO INTERNAL FORMAT
```

```
0210'  91                520          IUB   C
0211'  47                521          LD    B,A
0212'  56 0033'          522          LD    A,(LINLEN)
0215'  5C                523          INC   A                    ; SET TO COL.0
0216'  4F                524          LD    C,A
0217'  CD 047A'          525          CALL  LNADD                ; GET ADDRESS
021A'  50                526          LD    D,B
021B'  59                527          LD    B,C                  ; SAVE POSITION
021C'  C1                528          POP   BC                   ; ORIG. BC
021D'  05                529          PUSH  DE
021E'  78                530  CLRSC1  LD    A,L                  ; GET # OF LINES THIS BLOCK
021F'  07                531          RLCA
0220'  07                532          RLCA
0221'  07                533          RLCA
0222'  4F                534          LD    C,A
0223'  38 08              535          LD    A,8
0225'  91                536          SUB   C
0226'  BB                537          CP    B                    ; COMPARE TO TOTAL LINE COUNT
0227'  58 5C              538          JR    C,CLRSC2
0229'  78                539  CLRSC2  LD    A,B                  ; NO. OF LINES
022A'  06 00              540          LD    B,0                  ; REMAINING LINES
022C'  C5                541          PUSH  BC
022D'  A7                542  CLRSC3  LD    B,A
022E'  0E 08              543          LD    C,8                  ; SCAN COUNT
0230'  78                544  CLRSCA  LD    A,B
0231'  C5                545          PUSH  BC
0232'  E6 07              546          AND   7
0234'  0F                547          RRCA
0235'  0F                548          RRCA
0236'  0F                549          RRCA
0237'  4F                550          LD    C,A                  ; BC=32*NO. OF LINES
0238'  06 00              551          LD    B,0                  ; (=0 IF 256 FOR 8 LINES)
023A'  09                552          ADD   C                    ; ADJUST
023B'  C5                553  CLRSC5  PUSH  BC
023C'  E5                554          PUSH  HL
023D'  66                555          LD    H,M
023E'  6D                556          LD    L,L
023F'  36 00              557          LD    (HL),00              ; CLEAR OP
0241'  IS                558          INC   DE
0242'  ED B8              559          LDIR
0244'  E1                560          POP   HL
0245'  C1                561          POP   BC
0246'  7C                562          LD    A,H
0247'  C6 6F              563          ADI   H,A
0249'  26 08              564          JR    NZ,CLRSC6
024B'  F6 20              565          OR    20H
024D'  67                566          LD    H,A                  ; DO OP2
024E'  18 88              567          JR    CLRSC8
0250'  E6 DF              568  CLRSC6  AND   0DFH                 ; RESTORE OP1
0252'  67                569          LD    H,A
0253'  24                570          INC   H                    ; NEXT SCAN ROW
0254'  C1                571          POP   BC                   ; POP SCAN COUNT
0255'  0D                572          DEC   C
0256'  20 08              573          JR    NZ,CLRSC4
0258'  C1                574          POP   BC                   ; TOTAL LINE COUNT
0259'  78                575          LD    A,B
025A'  A7                576          AND   A
025B'  28 05              577          JR    I,CLRSX7
025D'  28 00              578          LD    L,0                  ; HL=START OF NEXT BLOCK
025F'  C3 0218'           579          JP    CLASC1               ; B=REMAINING LINE COUNT
0262'  CI                580  CLRSX7  POP   BC                   ; POSITION
0263'  C3 0456'           581          JP    UPDPOSN              ; RETURN VIA UPDATE AND STORE POSITION
0266'  AF                582  CLRSC7  LD    C,A                  ; SAVE NO. LINES THIS BLOCK
0267'  78                583          LD    A,B
0268'  91                584          SUB   C                    ; ADJUST TOTAL LINE COUNT
0269'  47                585          LD    B,A
026A'  C5                586          PUSH  BC                   ; REMAINING LINE COUNT
026B'  79                587          LD    A,C                  ; LINES THIS BLOCK
026C'  18 BF              588          JR    CLRSC3               ;
                         590          SUBTTL ATTRIBUTE CONTROL
                         591  ;
                         592  ;
                         595  ;
026E'                    596  IETA80:                            ; HERE FROM BASIC ENTRY TO SET ATTRIBUTES
0268'  3A 0011'          597          LD    A,(ATTCTL)           ; GET CONTROL INFO.
                         598  ;
0271'                    599  SETATT:                            ; ENTRY WITH ATTRIBUTE BYTE IN A
0271'  E6 07              600          AND   7                    ; GET INK SELECTION
0273'  AF                601          LD    C,A                  ; SAVE
0274'  3A 5CC2            602          LD    A,(VIDMOD)           ; TEST IF OP2 OPEN
0277'  A7                603          AND   A
0278'  CA 00C2'           604          JP    Z,INVPAR             ; INVALID IF OP2 NOT OPEN
027B'  79                605          LD    A,C
027C'  17                606          RLA
027D'  17                607          RLA
027E'  17                608          RLA                        ; SHIFT TO H/W POSITION
027F'  F5                609          PUSH  AF                   ; SAVE ROTATED VALUE
0280'  F6 06              610          OR    6                    ; SET H/W MODE
0282'  47                611          LD    B,A                  ; VALUE FOR H/W
0283'  DB FF              612          IN    A,(HREXP7)           ;
0285'  E6 C0              613          AND   0C0H                 ; PRESERVE BITS 7 AND 6
0287'  90                614          OR    B                    ; SET VIDEO MODE VALUE
0288'  D3 FF              615          OUT   (HREXP7),A           ; SET H/W
028A'  F6 40              616          OR    40H                  ; UPDATE VIDMOD
028C'  32 5CC2            617          LD    (VIDMOD),A           ;
028F'  F1                618          POP   AF                   ; INK SELECTION IN BITS 3-5
0290'  2F                619          CPL                        ; GET COMPLEMENTARY COLOR
0291'  B1                620          OR    C                    ; COMBINE WITH INK
0292'  32 0039'           621          LD    (ATTBYT),A           ; SAVE IN VARIABLE
0295'  C3 0080'           622          JP    GOODRET              ; RETURN BC=0
                         623  ;
                         624  ;
                         625  ;
0298'                    626  SETMSK:                            ; HERE FROM BASIC ENTRY TO SET MASK
0298'  3A 0015'           627          LD    A,(MSKCTL)           ; GET CONTROL INFO.
                         628  ;
0298'                    629  SETMSK1:                           ; ENTRY WITH MASK VALUE IN A
0298'  32 0055'           630          LD    (MASK0),A            ; STORE MASK FOR APPLICATION TO FUTURE
0298'  C3 0080'           631          JP    GOODRET              ; DISPLAYS
                         650  ;
```

```
                                632           SUBTTL  'GET' ROUTINES
                                633   ;
                                634   ;
02A1'                           635   GTCH001:                        ; HERE FROM BASIC ENTRY TO GET CHARACTER
                                636                                   ; (RETURNS CODE FOR CHARACTER AT
                                637                                   ;  POSITION SPECIFIED IN GETCTL)
02A1' 3C 48 0018'               638           LD    BC,(GETCTL)       ; GET CONTROL INFO.
                                639   ;
02A3'                           640   GTCHAR:                         ; ENTNT WITH POSITION IN BC
02A5' CD D45A'                  641           CALL  TSTPAR            ; TEST PARAMETERS
02A8' CD D465'                  642           CALL  CONVBM            ; CONVERT TO INTERNAL FORMAT
02AB' CD D455'                  643           CALL  CALCPOS           ; GET DISPLAY FILE ADRS.
02AE' ED 58 0020'               644           LD    DE,(CHTBL)        ; CHAR.TABLE
02B2' DA 6D                     645           LD    B,96              ; NO. OF PRINTABLE CHARACTERS
02B4' 3E 80                     646           LD    A,80H             ; SET ADJUSTMENT INDEX
02B6' 32 003A'                  647   GTCHI   LD    (GTINDX),A
02B9' 14                        648           INC   D                 ; ADJUST TO START OF TABLE
02BA' ER                        649           ER    DE,HL             ; DF ADRS.IN DEICHAR.SET IN HL
02BB' C5                        650   GTCH2   PUSH  BC                ; SAVE COUNT
02BC' D5                        651           PUSH  DE                ; SAVE ADRS. IN CF
02BD' E5                        652           PUSH  HL                ; SAVE ADRS. IN CHAR.TABLE
02BE' 1A                        653           LD    A,(DE)            ; SCAN ROW PACK DF
02BF' AE                        654           XOR   (HL)              ; TEST AGAINST CHAR. SET
02C0' 28 10                     655           JR    Z,GTCH3           ; MATCH
02C2' F5                        656           PUSH  AF
02C3' 3A 003A'                  657           LD    A,(GTINDX)
02C6' FE 80                     658           CP    80H               ; TEST IF STD.GRAPHICS
02C8' 10 03                     659           JR    NS,GTCH23
02CA' F1                        660           POP   AF
02CB' 1B 2T                     661           JR    GTCH4             ; DO NOT TEST FOR INVERSE
02CD' F1                        662   GTCH23  POP   AF
02CE' 1C                        663           INC   A                 ; TEST INVERSE
02CF' 10 23                     664           JR  N2,GTCH4
02D1' 1D                        665           DEC   A                 ; A=-1 FOR INVERSE
02D2' 4F                        666   GTCH3   LD    C,A               ; C=0 FOR MATCH  -1 FOR INVERSE
02D3' 06 0T                     667           LD    B,T
02D5' L4                        668   GTCH31  INC   D                 ; NEXT SCAN IN DF
02D6' 23                        669           INC   HL                ; NEXT BYTE IN CHAR.SET
02D7' 1A                        670           LD    A,(DE)
02D8' AE                        671           XOR   (HL)
02D9' A8                        672           XOR   C
02DA' 20 10                     673           JR    N2,GTCH4          ; NO MATCH
02DC' 10 P7                     674           DJN2  GTCH31    ; FOR NEXT SCAN
                                675                           ;
                                676                           ; HERE IF MATCH WITH (SP)=CHAR IN CHAR SET
                                677                           ; (SP+2)=CHAR.IN DISPLAY FILE (SP+5)=CHAR.
                                678                           ; COUNT
02D9' 19                        679           LD    A,C       ; A=-1 IF MATCH ON INVERSE
02DA' CI                        680           POP   BC
02E0' C1                        681           POP   BC
02E1' C1                        682           POP   BC
0232' 4F                        683           LD    C,A       ; B=COUNT C=-1 IF MATCH ON INVERSE
02E5' 3A 003A'                  684           LD    A,(GTINDX)        ; ADJUST CHAR.CODE
02E8' 8D                        685           SUB   B         ; A=CHAR.CODE
02E7' FE 20                     686           CP    20H       ; TEST IF SPACE
02E9' 20 09                     687           JR    NZ,GTCHS2
02EB' 9C                        688           INC   C         ; TEST IF MATCH ON INVERSE SPACE
02EC' 20 02                     689           JR    N1,GTCHS2
02EE' 3E 8F                     690           LD    A,9FH     ; SET CODE FOR GRAPHICS BLOCK
02F0' 4F                        691   GTCH32  LD    C,A
02F1' 0R 00                     692           LD    B,0
02F3' C9                        693           RET             ; CHAR.CODE IN A AND IN BC
                                694   ;
02F4' E1                        695   GTCH4   POP   HL        ; HERE WHEN NO MATCH
02F5' 11 0000                   696           LD    DE,0      ; LOOK AT NEXT CHAR. IN SET
02F8' 19                        697           ADD   HL,DE             ; CHAR. SET
02F9' 01                        698           POP   DE                ; CHAR. IN DF
02FA' C1                        699           POP   BC                ; CHAR.COUNT
02FB' 10 0E                     700           DJNS  GTCH2     ; LOOK AGAIN
                                701                           ;
                                702                           ; HERE TO EXIT IF DONE
02FD' E8                        703           EX    DE,HL     ; DF ADRS.TO HL
02FE' 3A 003A'                  704           LD    A,(GTINDX)
0301' FE 80                     705           CP    80H       ; TEST IF STD.CHR.SET
0303' 20 0A                     706           JR    NS,GTCHS  ; TRY GRAPHICS
0303' ED 58 0031'               707           LD    DE,(GRTBL); STD.GRAPHICS TABLE
0309' 06 I0                     708           LD    B,16      ; NO. OF ENTRIES
030B' 3E 8D                     709           LD    A,80H     ; INDEX
030D' 18 A7                     710           JR    GTCHS     ;
030F' FE 80                     711   GTCH3   CP    80H       ; TEST IF STD.GRAPHICS
0311' 20 08                     712           JR    NZ,GTCHR  ; DONE IF NOT
0315' E4 58 SCT0                713           LD    DE,(UDG)  ; TRY USER-DEFINED GRAPHICS AREA
0317' 19                        714           DEC   D         ; ADJUST ADDRESS-100H
0318' 06 19                     715           LD    B,21      ; NO OF ENTRIES
031A' 3E A5                     716           LD    A,0A5H    ; INDEX
031C' 58 98                     717           JR    GTCH1
031E' 48                        718   GTCH6   LD    C,B       ; HERE WHEN NO MATCH ANYWHERE
031F' AF                        719           RDB   A         ; SET BC AND A=ZERO
0320' C9                        720           RET
                                721
                                722                           ; GET ATTRIBUTE BYTE
                                723                           ; NOTE THAT IN 64-COL.MODE ALL SCREEN POSITIONS
                                724                           ; HAVE THE SAME ATTRIBUTES, THEREFORE IT IS
                                725                           ; NOT NECESSARY TO USE THE 'GETCTL' POSITION
                                726                           ; PARAMETERS
                                727   ;
0321'                           728   GET600:                 ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                                729   ;
0321' 3A 0058'                  730   GETATT: LD    A,(ATTRTT) ; GET ATTRIBUTE BYTE
0324' 4F                        731           LD    C,A       ; PUT IN BC
0325' 0R 00                     732           LD    B,0
0327' C9                        733           RET
                                734   ;                       ; GET CURSOR POSITION
0328'                           735   GETC00:                 ; HERE FROM BASIC ENTRY
```

```
0328'                   GETCUE:
032E'  ED 4B 0034'              LD    BC,(CURPOS)        ; GET INTERNAL POSITION
032C'  CD 044E'                 CALL  CONVPM             ; CONVERT TO USER FORMAT
032F'  3A 0033'                 LD    A,(LINLEN)
0332'  B9                       CP    C                  ; TEST IF END OF LINE(=64)
0333'  2E 0E                    JR    NZ,GETC3           ; NO
0335'  0E 00                    LD    C,0                ; NEXT POSN.START OF NEXT LINE
0337'  78                       LD    A,B
0338'  3C                       INC   B                  ; BUMP TO NEXT LINE
0339'  47                       LD    B,A
033A'  FE 1E                    CP    24                 ; TEST IF OFF SCREEN
033C'  38 02                    JR    C,GETC3            ; NO
033E'  0A 37                    LD    B,23               ; POSN. IS AT BOTTOM LINE
0340'  ED 43 000T'      GETC3   LD    (LINCOLS,BC        ; STORE FOR BASIC PROGRAM
0344'  C9                       RET                      ; VALUES ALSO IN BC
                        ;
                                SUBTTL  VIDEO MODE CONTROL
                        ;
                        ;
0345'                   STMODO:                   ; HERE FROM BASIC ENTRY WITH MODE IN
                                                  ; "VIMODE"
0345'  3A 0024'                 LD  A,(VIMODE)
                        ;
0348'                   SETMODE:                  ; ENTRY WITH MODE IN A
                        ;
0348'  A7                       AND   A
0349'  28 4A                    JR    Z,SETM01    ; TEST FOR VALIDITY
034B'  FE 0A                    CP    A
034D'  C2 00C2'                 JP    NZ,INVPAR   ; INVALID PARAMETERS
0350'  21 1701                  LD    HL SCINIT   ; INITIALIZE VARIABLES
0353'  22 0028'                 LD    (SCRCTL),HL ; SCROLL CONTROL
035A'  3E 17                    LD    A,23        ;
035E'  32 0020'                 LD    (BOTLN),A   ; BOTTOM LINE
035E'  3E 01                    LD    A,1         ;
0350'  32 002E'                 LD    (SCRLCT),A  ; SCROLL COUNT
03A0'  3E 40                    LD    A,64        ;
0382'  32 0033'                 LD    (LINLEN),A  ; LINE LENGTH
0383'  21 1800                  LD    HL,CLINIT   ;
036E'  22 000C'                 LD    (CLRCTL),HL ; CLEAR SCREEN CONTROL
036B'  AF                       XOR   A
03AC'  32 0038'                 LD    (MASKB),A   ; MASK BYTE
03AF'  32 0033'                 LD    (STRGCT),A  ; WRSTRG REMAINING COUNT
0372'  32 003C'                 LD    (STRGCT+1),A ;
0375'  21 3C00                  LD    HL,CHRSET   ; STD.CHAR.SET
0378'  22 002F'                 LD    (CHTBL),HL  ;
037E'  21 0395'                 LD    HL,GRPHST   ; STD. GRAPHICS SET
037E'  22 0031'                 LD    (GRTBL),HL  ;
                        ;                         ;
                        ;                         ; INITIALIZE BASIC INPUTS
03E1'  32 0000'                 LD    (DATAB),A   ; DATA BYTE
03E4'  32 0007'                 LD    (LINCOL),A  ; COLUMN
03E7'  32 0008'                 LD    (LINCOL+1),A ; LINE
038A'  32 0013'                 LD    (MSKCTL),A  ; MASK CONTROL
038D'  32 0019'                 LD    (GETCTL),A  ; "GET" COLUMN
0390'  32 001A'                 LD    (GETCTL+1),A ; "GET" LINE
0393'  3E 06                    LD    B,6         ; SET H/W TO 64-COL.MODE
                        ;
0399'  47               SETM01  LD    B,A         ; SAVE MODE
039A'  3A 3CC2                  LD    A,(VIDMOD)
0399'  A7                       AND   A           ; TEST CURRENT MODE
039A'  20 10                    JR    NZ,SETM02
039C'  B0                       OR    B           ; CURRENT MODE=0  TEST IF NEW MODE=0
039D'  CA 0C50'                 JP    Z,GOODRET   ; RETURN BC=0
03A0'  21 12C0                  LD    HL,INSERTS
03A3'  11 0840                  LD    DE,MOVES2
03A4'  19                       ADD   HL,DE       ; HL=TOTAL ROOM NEEDED
03A7'  ED 5B 5CA5               LD    DE,(STKEND)
03A8'  19                       ADD   HL,DE       ; ADD MEMORY IN USE
03AC'  DA 03F6'                 JP    C,EXTNRM    ; NOT ENOUGH MEMORY TO OPEN OP2
03AF'  ED 5B 5CB2               LD    DE,(RAMTOP)
03B3'  A7                       AND   A
03B4'  ED 52                    SBC   HL,DE
03BA'  D2 03FA'                 JP    NC,EXTNRM   ; NOT ENOUGH MEMORY
03B9'  CC 0421'         SETM02  CALL  ENBLEXT     ; ENABLE ROM EXT.
03BC'  78                       LD    A,B         ; MODE TO A
03BD'  CD 03FE'                 CALL  GETVAL      ; H/W VALUE TO B; VIDMOD VALUE TO C
03C0'  C5                       PUSH  BC          ; SAVE
03C1'  78                       LD    A,B
03C2'  CD 0EAE                  CALL  CHNGVID     ; CALL RTN. TO OPEN/CLOSE 2ND OP
03C5'  CD 042E'                 CALL  ENBLHOME    ; REENABLE HOME BANK
03C8'  C1                       POP   BC
03C9'  79                       LD    A,C
03CA'  32 3CC2                  LD    (VIDMOD),A; UPDATE VIDMOD
03CD'  CB 77            UPDATE  BIT   A,A               ; UPDATE SYSTEM VARS. BASED ON MODE
03CF'  CA 00E0'                 JP    Z,GOODRET         ; DONE IF MODE 0
03D2'  3E C1            SXT4    LD    A,CC3H            ; 64/80-COL. MODE
03D4'  E0                       OR    B                 ; STORE ATTRIBUTE BYTE BASED
03D5'  2F                       CPL                     ; ON INK SELECTION
03DA'  CB 3E                    SRL   E
03DB'  CB 3E                    SRL   E
03DA'  CB 3E                    SRL   B
03DC'  B0                       OR    E
03DD'  32 0039'                 LD    (ATTBYT),A        ; ATTRIBUTE BYTE
03E0'  32 0011'                 LD    (ATTCTL),A        ; BASIC PARAMETER
03E3'  3A 0033'         UPDT1   LD    A,(LINLEN)        ; GET LINE LENGTH
03EA'  3C                       INC   A
03E7'  4F                       LD    C,A               ; COLUMN
03EE'  0A 1E                    LD    B,SC833           ; BC=HOME POSITION (LN.0/COL.0)
03EA'  21 0000                  LD    HL,0
03ED'  22 3C70                  LD    (COORDS),HL       ; INITIALIZE PLOT POSITION
03F0'  CD 045A'                 CALL  UPCPOSN           ; UPDATE AND STORE HOME POSITION
03F3'  C3 00E0'                 JP    GOODRET           ; RETURN BC=0
03F6'  0E 02            EXTNRM  LD    C,2               ; RETURN BC=2 FOR NO ROOM
03F8'  C3 008F'                 JP    ERRRET
                        ;
```

```
                        SUBTTL  INTSRNAL SUB-RTNS.
                ;
                ;
                ;
                                        ; INPUT:  V1080 MCDE IN A
                                        ; OUTPUT: M/W SSTTING IN B
                                        ;         VIOMCD SETTING IN C
                                        ; BSTURNS N1 STATUS IF MCDE INVALIO
                                        ;
                                        ;  VBLID VALUSS:  INPUT     M/W      VIOMOD
                                        ;                 -----     ---      ------
                                        ;   G=DISPLAY PILS ACTIVS AT SCREEN
                                        ;
                                        ;    DP1 ONLY       0         0          0
                                        ;   DP1&L DP2      80         0         R0
                                        ;   DP1 S DP2=      1         1         81
                                        ;   HIGH RS3.S0.    I         2          2
                                        ;  64/80-COLUMN  B6 - 55   B6 - 5E   46 - T8
03PB'   4P      GETVAL  LD      C,A
03PC'   CB 4T           BIT     0,A
03PE'   28 18           JR      NZ,TSTS1
0400'   PS 05           CP      00H
0402'   28 14           JR      1,SETO
0404'   CB 7P           BIT     7,A
0406'   C0              RET     N1
0407'   47              LD      B,A
0408'   PE 02           CP      2
040A'   CR              RST     1
040B'   E6 C0           AND     0C0H
040D'   EE 06           XOR     0
040F'   CR              RET     NI
0410'   3E 4R           LD      A,4BH
0412'   B1              CR      C
0415'   4P              LD      C,A
0414'   AP              ROR     A
0415'   C9              RST
0416'   PS 01   TSTB3   CP      1
0418'   C0              RET     N2
0419'   06 01           LD      B,3
041B'   05 01           LD      C,01H
041D'   C9              RST
041E'   AP      SSTB    ROR     A
041F'   47              LD      B,A
0420'   C9              RET
                ;
                ;
                                        ; ROUTINSS TO SNABLE CHUNK R IN RCM B4T.
                                        ; FOR ACCESS TO CHMGVIO ROUTINE
0421'   P8      SNBLB4T PUSH    AP
0422'   DR PP           IN      A,(HRBRPT)
0424'   P6 80           CR      80H         ; SELSCT ROM SRT.
042A'   D5 PP           OUT     (HREXPT),A
0425'   35 03           LD      A,3
0424'   D5 P6           OUT     (OKMSPT),A
042C'   P1              POP     AP
042D'   C9              RET
                ;
041E'   PS      ENBLMONS  PUSH    AP
042F'   AP                ROR     A
0430'   D3 P6             OUT     (OKM3PT3,A
0432'   08 PP             IN    A,(HRB4PT)
0434'   58 3P             AND   03PH
0438'   D5 PP             OUT   (H4EXPT3,A
043R'   P1                POP   AP
0439'   C9                RET
                ;
                                                ; TEST PARAMETER VALIDITY
                                                ; IF NOT VALID, DIICARDS RETURN
                                                ; AND EXITS VIA INVPAR
043A'   3E 1T   TSTPAR  LD      A,23            ; TEST LINE >23
043C'   58              CP      5
0430'   30 04           JR      NC,TSTPR1
043F'   P1      PARERR  POP     AP
0440'   C5 00C2'        JP      INVPAR
0443'   3A 0035' TSTPR1 LD      A,(LINLEN)
0446'   3C              DEC     A
0447'   B9              CP      C               ; TEST COL. >83
0445'   38 PS           JR      C,PARSER
044A'   C0              RET
                ;                               ; CONVSRT LINE/COL. FROM USER TO
                                                ; INTERNAL FORMAT AND VICE VERSA
                                                ; USER FORMAT:  LINES 0-23
                                                ;               COL.0-A3
                                                ; INTERNAL FORMAT: LINES 24-1
                                                ;                  COL. 65-2
                                                ;                  (COL.1 END OF LINE)
044B'   3A 0053' CONVPM LD      A,(LINLEN)
0445'   3C              INC     A
044F'   91              SUB     C
0450'   4P              LD      C,A
0451'   36 18           LD      A,3CRSI
0433'   95              SUB     B
0454'   47              LD      B,A
0455'   C9              RST
                ;
                ;
                ;
0456'           UPDPOSN                         ; ENTER HERE WITH BC=LINE/COL.IN
                                                ; INTERNAL FORMAT
0458'   CD 0658'        CALL    CALCPOS         ; CALCULATE POSITION
0455'   15 26           JR      STPOSN          ; STORE UPDATED POSITION AND RETURN
                ;
                ;
                ;                               ; ROUTINE TO CALCULATE POSITION IN DP
```

```
                                          1 R0TURN3 OP ADR3. IN HL.  PRESERVES
                                          1 LINE/COLUMN PO3I7ION IN BC
                  1
049E'  CD 0474'   CALCPOS C0LL   LN9U     1 GET DI3PLAY PILE 0DRS.
049E'  30 0033'           LD     A,(LINLEN)  1 7EST WHICH OP
0441'  3C                 INC    A
0442'  91                 SUE    C
04A3'  CE 47              EIT    0,0
0449'  2R 0A              JR     2,C0LCP1
0447'  F9                 PUSH   AP
04A9'  3E 20              LD     A,20H
04A0'  A6                 OR     H        1 IN OP2
04AE'  A7                 LD     H,A
I46C'  F1                 PGP    AF
04AD'  A7         CALCP1  AND    A
04AE'  1P                 RA4             1 COL/2 IS POSITION IN DISPLAT PILE
046P'  9P                 LD     0,A
0470'  1A 00              LD     D,E
0472'  19                 ADD    HL,DE
0475'  C9                 RET
                  1
                  1
0474'             LNEU:                   1 G0T DISPLAT PILE ADDA09S
                                          1 POR STAR7 OP LINE IN E
0474'  3E 1E              LD  A,SC0S2
047A'  90                 SUE  E
0477'  37                 LD   D,0
047R'  0F                 ARCA
0479'  0F                 0RCA
047A'  0F                 0ACA
047E'  E6 E0              AND  0E0H
047D'  AP                 LD   L,A
D47E'  7A                 LD   A,D
047P'  EA 1R              AND  1EH
04R1'  *A 40              OR   40H
04E3'  67                 LD   H,A
04E4'  C9                 RE7

0489'             9TPOSN:                 1 STOR0 CURSOR POSITION
04E5'  EC 43 0034'        LD  (CURPCS),EC
04E9'  22 003A'           LD  (DPACA9),HL
040C'  C9                 R0T
                  1
04ED'             LDPOSN:                 1 LOAD CURSO0 POSITION
04ED'  0C 4E 0034'        LD  EC,(CLAPD3)
0491'  20 0035'           LD  HL,(SP0DAS)
0494'  C9                 00T
                  1
                  1
                  1
                  1
                          SU0TTL GAAPHICS CNAR.10T
                  !
                  1                       1PIREL PATTERNS *OR STD.GRAPHICS 107
                  1                       1 0=0ACAGROUND(PAP0R) 1=PDAEG0JUND(INK)
                  1
0499'  00         GRPS7   6efb 00000000b  ! RA6A 00        GRAPHIRE A0EA0
0496'  00                 Aefb 00000000b
0497'  00                 Aefb 0000000A
0498'  00                 ReFA 0000000e
0499'  00                 Refb 00000000b
049A'  00                 Aefb 0000000A
049E'  00                 Refb 00000000b
049C'  00                 Refb 00000000b
                  1
049D'  0F                 Refb 000C1111b  1 eeRe 01        GRePNIAe
0498'  0F                 Aefb 00011111b
049P'  0F                 Aefb 0001111A
040E'  0P                 Aefb 000C1111b
0441'  00                 AefA 00000000b
0442'  00                 Aefb 000000E0b
0445'  00                 Refb 00000000b
0446'  00                 6efb 00000000b
                  1

0445'  F0                 6efb 11110000b  1 AAde 02        GRRPNIRA
0446'  F0                 6efb 11110000b
0447'  F0                 RefA 11110000b
0448'  F0                 6efb 11110000b
0449'  00                 Refb 000C0000A
044A'  00                 6efb 000C0000b
044E'  00                 6efb 00000000b
044C'  0E                 6efb 00000000b

04AD'  PP                 Aefb 11111111b  1 Ae6e 03        GRePHIAA
04A0'  FP                 6efb 11111111b
04A*'  PP                 6efb 1111111e
0490'  PP                 6efb 11111111b
0401'  00                 Refb 00000000b
04E2'  00                 Aefb 0000000A
04A5'  00                 Refb 00000000b
0404'  00                 6efb 000C0000b

                  !
04A5'  00                 Refb 00000000b  1 eARA 04        EREPHIAR
04EA'  00                 defb 0000000A
04E7'  00                 Refb 00000000b
04AR'  00                 eefb 0000000A
04E8'  0P                 RefA 0001111A
04AA'  0P                 Refb 0001111A
04EE'  0P                 Refb 00011111b
046C'  0P                 6efb 00001111b
                  1
```

INSDBL	4090						
INSBXT	700	400					
INV04R	180	182	2430	379	382	384	588
		511	513	402	745	914	
LOPQSN	175	9900					
L2NCOL	870	357	7498	767R	7888		
L2NLBN	1840	267	392	822	739	7738	882
		917	829	951			
LNBU	395	525	950	R490			
LOJ0	4120	442					
LOJP0	4140	441					
LOJP1	4210	435					
M4BA6	1500	223	626R	777R			
MOVBS2	710	72	801				
MSKCTL	1150	625	7BR8				
NOSTRG	300	3490					
NXTSC	432	4340					
P484MR	480	283					
P888R8	9150	920					
PR4NT	440						
R4MTOP	650	806					
RECLBN	540	304					
SC1WIT	500	766					
SCRCTL	1280	283	372	767B			
SCRLO	285	385	3880				
SCRL8	65	1310					
SCRL86	131	3700					
SCRLCT	1430	273	277R	2828	771R		
SCRLXT	445	4480	481				
SCRCLL	43	3740					
SCRS2	490	246	389	51R	834	933	871
S8T0	565	8840					
S8T480	114	5R40					
S8T4T8	45	1140					
S8T4TT	43	59T0					
S8TC80	100	3550					
S8TCU8	45	1000					
S8TCU8	43	3590					
S8TM80	118	4240					
S8TM01	743	T940					
S8TM02	T9T	810B					
S8TM08	47	12T0					
S8TM0D	47	T600					
S8TM58	45	1180					
S8TM5R	43	627B					
S78LR	400	4440					
S78LR0	4670	487					
S78LR1	4680	481					
S78LR2	478	4820					
STKEND	62	803					
S7M880	127	T560					
S7P05N	258	843	T870				
STRGCT	164R	343R	7788	T798			
SXT4	8220						
T85T40	3880	447	494				
TSTR1	863	87R0					
TSTP4R	366	441	9120				
TSTPR1	914	9170					
TVPUL1	275	2820					
TV0ULQ	214	2460					
U0G	630	186	T13				
UPD4T8	8200						
UPDPQ3	271	362	561	837	9480		
UPDT1	8310						
V4R8	610	287					
V2D4D8	670	600	615R	785	6598		
V2MQ08	1240	T58					
W8CM8	94	1T0B					
W8CM11	187	1920					
W8CM12	188	1850					
W8CM13	491	194	1860				
W8CM1C	286	2130					
W8CM18	212	2140					
W8CM2	2190						
W8CM3	224	2260					
W8CM5	2330	243					
W8CM7	2440						
W8CM48	43	1T40	328				
W8CMX6	48	940					
W8CMXT	253	2580					
W8STR6	98	2530					
W8STR1	2980	307					
W8STR2	302	3080					
W8STX3	3300	341					
W8ST86	45	850					
W8ST0B	43	3220	336				
W8STR1	328	3380					
W8STX2	3460	356					

APPENDIX C-2

80 COLUMN MODE

Date: 12/16/83

ASC Number: 002

Version: 001

Author: C. Corcoran/O. Boyle

Name: 80 Column Mode Support

Description:

This component provides support to the application programmer for using the 80-column mode feature of the TS 2068. 80-Column Mode is implemented by using the 64-Column Mode feature of the 2068 and modifying the character width from 8 to 6 pixels. The services include opening/closing the second display file (moving the machine stack, OS RAM routines and BASIC structures), PRINT position control, attribute control, clear screen and scroll screen services and display of characters. For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and designates non-zero values for other conditions. The interface from BASIC is by means of POKE'ing the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chunk T (E000H/57344).

- -

Applications Services

- -

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

Input: MODE (0=normal , 8=80 column mode)

From machine code: Register A
From BASIC: In VIMODE

Description:

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

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

- -

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

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

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

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

Description:

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

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

- -

- -

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

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

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

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

Description:

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

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

- -

Name: SETATT (SETATB from BASIC - parameter to ATTCTL)

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

From Machine Code: Register A
From BASIC: In ATTCTL

Description:

The specified INK color (0-7) is used to set the video mode hardware and to update VIDMOD. The complementary PAPER color is fixed by the INK selection. FLASH and BRIGHT are fixed at zero by the hardware. Note that in 80 column mode the entire screen has the same attributes.

Output: BC = 0 Successful

- -

Name: WRCHR (WRCHB from BASIC - parameter to DATAB)

Input: Character code for character to be displayed

20H TO 7FH - Std. TS2068 Character Set
80H TO 8FH - Std. Graphics Set
90H TO A4H - User-Defined Graphics Set

From Machine Code: Register A

From BASIC: in DATAB

Description:

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

Note that only the first 8 bits of each byte in the User Defined Graphics area will be transferred to the display file.

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

- -

- -

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

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

From Machine Code: Register A

From BASIC: MSKCTL

Description:

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

Output: BC = 0 for successful completion

- -

Name: WRSTRG (WRSTRB from BASIC - String Identifier to PARAMS)

Input: Character Code String

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

From BASIC: String Variable Identifier in
System Variable PARAMS - 23747 (5CC3H)

Description:

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

NOTE: Characters within the string which are outside of the supported range (32 through 170 (20H-AAH)) will be ignored. E.g., BASIC Token codes and control codes embedded in an INPUT string will not be displayed or decoded.

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

```
0005  LET a$="-----string------"
0010  POKE 23747,CODE "a"
0015  IF USR (WRSTRB)<>0 THEN -----
         (continue)
```

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

- -

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

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

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

Description:

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

Note: See Usage Section on 'automatic' scrolling.

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

- -

- -

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

Input: Line/Column position as for SETCUR

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

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

Description:

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

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

Output: BC = 0 for no find
BC = 1 invalid parameters
BC = character code (20h-A4H)

- -

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

Input: As for GTCHAR

Description:

Returns in register C of the BC register pair the attribute byte for the character at the designated screen position. Note that in 80 column mode the entire screen has common attributes. The value returned will describe the current selection:

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

- -

Name: GETCUR (GETCUB from BASIC)

Input: None

Description:

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

Output: B = Line number (0-23)
C = Column number (0-79) or (0-84)

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

NOTE: If the last character was printed at Col.79 (84) of Line 23 (last position on the screen), then Col.0/Line 23 will be returned.

- -

Usage:

Memory Usage:

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

Name	Size	Description
SCRCTL	2	Scroll Control LSB = Starting line number MSB = Number of lines to be scrolled
BOTLN	1	Bottom Line - Line number (0-23) after which test for scroll will be made.
SCRLCT	1	Scroll Count- Number of times + 1 that automatic scroll will be done. When decremented to zero, data will not be displayed and a condition code will be returned to user.

CHTBL	2	Character Table (Base Address-100H)
GRTBL	2	Std.Graphics Character Table (Base-100H)
LINLEN	1	Line Length - (80 or 85 when in 80-Col.Mode)
CURPOS	2	Current Position (Internal Format) LSB = Column Position MSB = Line Position
DFADDR	2	Current Display File Address
MASKB	1	Mask Byte (bit 0 = OVER) (bit 2 = INVERSE)
ATTBYT	1	Attribute Byte (bits 0 - 2 - INK) (bits 3 - 5 - PAPER) (bit 6 - BRIGHT (Set to zero (bit 7 - FLASH by M/c in 80-col.mode)
GTINDX	1	'Get' Index - Used by GTCHAR
STRGCT	2	String Count - Conteins remaining byte count when EC=2 (Screen Full) is returned from the Write String service WRSTRG (WRSTRB).
MARGIN	1	Margin - Margin Adjust (0-2)
DFBIT	1	Display File Bit - Current Bit Position

Initial values set via SETMODE (SETMOB) are as follows:

Variable Name	Value
SCRCTL	1701H
BOTLN	17H
SCRLCT	1H
CHTBL	(Internal to Module)
GRTBL	(Internal to Module)
LINLEN	50H
CURPOS	1851H
DFADDR	4000H
MASKB	0H
ATTBYT	38H
GTINDX	80H
STRGCT	0H
MARGIN	1H
DFBIT	7H

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

Variable Name	Size	Value
* DATAB	1	0H
* LINCOL	2	0H
* CLRCTL	2	1800H
* ATTCTL	1	38H
* MSKCTL	1	0H
* GETCTL	2	0H
VIMODE	1	

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

Location:

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

The routine SETMODE (SETMOB) cannot be executed from a cartridge because of the necessity to enable the ROM Extension which disables the DOCK Bank.

Registers:

Other than as documented for output values, no claims are made as to preservation of any register contents except for the IY Register which must always contain the value 5C3Ah for access to the standard system variables.

Automatic Scrolling:

As initialized, test for scrolling will be made when the print position goes to the next line following the bottom line on screen (BOTLN=23=24th line). Condition Code 3 (Screen Full) will be returned since SCRLCT will decrement to zero. If SCRLCT is set to some larger value, then the parameters in SCRCTL will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a POKE or setting the variables BOTLN and SCRCTL to the desired values, automatic scrolling can be done using smaller sections of the screen. When working from BASIC it is recommended that BOTLN be set to Line 21 (15H) and SCRCTL+1 be set to 21 (15H) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Note that once SCRLCT expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the 'Screen Full' condition.

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

Margin Control:

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

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

ADDITIONAL NOTES:

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

2. The COPY command will print only every other grouping of 8 pixels across the screen to the 2040 Printer.

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

4. The SAVE filename SCREEN$ will save only the main Display File data. The second can be saved by a SAVE filename CODE 24576,6144. Be careful that you have the computer in 80-column mode when you load this data or you will overwrite the OS RAM routines and "crash" the system!! (The count for saving the display file is for the data portion only since the Attribute File area from 7880H-7AFFH (30720-31487) is not used by the video mode H/W in 80-Column mode.

```
                         NAME ADL - ASC 002 80-COL.MODE SUPPORT
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;                   **********************************
                 ;                   *                                *
                 ;                   *              TIMEX             *
                 ;                   * APPLICATION DEVELOPMENT LIBRARY *
                 ;                   *                                *
                 ;                   * NAME: 80-COL.MODE SUPPORT      *
                 ;                   *                                *
                 ;                   * ASC NO:  002                   *
                 ;                   * VERSION: 001                   *
                 ;                   * AUTHOR:  C. CORCORAN/D. BOTLE  *
                 ;                   * DATE:    12/16/83              *
                 ;                   **********************************

                         SUBTTL DEFINITIONS
                 ;
                 ;
                 ;                         *****DEFINITIONS*****
                 ;
                         INTERN  WRCHAR, WRSTRG, SETCUR, SETATT, SETMSK, CLRSCN, SCROLL
                         INTERN  GTCHAR, GETATT, GETCUR
                 ;
                         INTERN  WRCHR8, WRSTR8, SETCU8, SETAT8, SETMS8, CLRSC8, SCRL8
                         INTERN  GTCHR8, GETAT8, GETCU8
                 ;
                         INTERN  SETMOD8, SETMD8
                 ;
                 ;
=0018            SCRSZ           EQU     24      ; 24 LINES
=1701            SCINIT          EQU     1701H   ; SCROLL CONTROL INITIALIZATION
=1800            CLINIT          EQU     1800H   ; CLEAR SCREEN CTL. INIT.
=0470'           CHASET          EQU     (ICHRST)-100H) ;CHAR.TABLE-100H
=077E'           GRPHST          EQU     ((GRPST)-100H) ; GRAPHICS TABLE
                 ;
=0E80            CHNGVID         EQU     0E80H   ; ROUTINE IN ROM EXTENSION
=1720            RECLEN          EQU     1720H   ; ROUTINE IN HOME ROM
                 ;
=00FF            HREXPT          EQU     0FFH    ; HOME ROM EXT.SELECT PORT (BIT 7)
=00F4            DRHSPT          EQU     0F4H    ; DOCK HORIZONTAL SELECT PORT
                 ;
=5C40            VARS            EQU     5C4BH   ; SYSTEM VARIABLES
=5C65            STKEND          EQU     5C65H   ;
=5C7B            UDG             EQU     5C7BH   ;
=5C7D            COORDS          EQU     5C7DH   ;
=5C82            PROTOP          EQU     5C82H   ;
=5C84            PARMT           EQU     5C84H   ;
=5CC2            VIDMOD          EQU     5CC2H   ;
=5CC3            PARMMA          EQU     5CC3H   ; (Location 23747)
=8840            DRIVES          EQU     4840H   ;
=12C0            INSERTS2        EQU     7E00H-DRIVES
=0040            MOVES1          EQU     DRIVES-6000H
=F7C0            DEST7           EQU     EFFFFH-MOVES2+.
=07C0            FIX             EQU     DEST7-6E00H
                 ;
=1000            FIXTBL          EQU     1000H
                 ;
                 ;
                 ;
                 ;
                 ;
                 ;                  **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                 ;
                 ;                                                               WITH
                 ;                                                         MODULE LOADED
                 ;                                                        AT 57344(E000H)
                 ;                                                        DECIMAL ADDRESS
                 ;                                                       ---------------
0000'  20        DATR8           DEFB    20H     ; CHAR.CODE FROM BASIC          57344
0001'  C3 003F'  WRCHR8:         JP      WRCH8   ; ENTRY TO WRITE CHARACTER      57345
0004'  C3 0171'  WRSTR8:         JP      WRSTR8  ; ENTRY TO WRITE STRING         57348
                 ;                               ; (see PARAMS above)
0007'  0008      LINCOL          DEFW    0       ; SET CURSOR PARAMETERS
                                                 ;   LINCOL=COL. (0-79)          57351
                                                 ;   LINCOL+1=LINE (0-23)        57352
0009'  C3 010A'  SETCU8:         JP      SETC80  ; ENTRY TO SET CURSOR POSN.     57353
000C'  1800      CLRCTL          DEFW    1800H   ; CLEAR SCREEN CONTROLS
                                                 ;   CLRCTL=STARTING LINE NO.    57356
                                                 ;   CLRCTL+1=LINE COUNT         57357
000E'  C3 017E'  CLRSC8:         JP      CLRS80  ; ENTRY TO CLEAR SCREEN         57358
0011'  38        ATTCTL          DEFB    38H     ; ATTRIBUTE CONTROL             57361
                                                 ;   BIT 7=FLASH (0 IN 80-COL.)
                                                 ;       6=BRIGHT(0 IN 80-COL.)
                                                 ;       5
                                                 ;       4=PAPER COLOR
                                                 ;       3
                                                 ;       2
                                                 ;       1= INK COLOR
                                                 ;       0
0012'  C3 02F3'  SETAT8:         JP      SETA80  ; ENTRY TO SET ATTRIBUTES       57362
0015'  00        MSKCTL          DEFB    0       ; MASK CONTROL FROM BASIC:      57365
                                                 ;   BIT 2=INVERSE
                                                 ;       0=OVER
0016'  C3 0310'  SETMS8:         JP      SETM80  ; ENTRY TO SET MASK CONTROLS    57366
0019'  0000      GETCTL          DEFW    0       ; "GET" CONTROL FROM BASIC      57369
                                                 ; FORMAT IS AS FOR LINCOL
001C'  C3 0326'  GTCHR8:         JP      GTCH80  ; ENTRY TO GET CHARACTER        57371
001F'  C3 03C0'  GETAT8:         JP      GETA80  ; ENTRY TO GET ATTRIBUTE        57374
0021'  C3 0304'  GETCU8:         JP      GETC80  ; ENTRY TO GET CURSOR POSN.     57377
0024'  00        VDMODE          DEFB    0       ; VIDEO MODE CONTROL            57380
                                                 ;       0=NORMAL
                                                 ;       8=80-COL.MODE
0025'  C3 03F1'  SETMD8:         JP      STMD80  ; ENTRY TO SET VIDEO MODE       57381
0028'  1701      SCRCTL          DEFW    1701H   ; SCROLL CONTROL
                                                 ;   SCRCTL=STARTING LINE NO.    57384
                                                 ;   SCRCTL+1=LINE COUNT         57385
002A'  C3 01CA'  SCRL8:          JP      SCRL80  ; ENTRY TO SCROLL               57386
```

```
                                SUBTTL  OTHER VARIABLES
                        ;
002D'   17              BOTLN           DEFB    17H     ; BOTTOM LINE (LINE AFTER        ET389
                                                        ; WHICH SCREEN IS CONSIDERED
                                                        ; FULL.  AUTOMATIC SCROLL
                                                        ; WILL BE DONE IF SCRLCT
                                                        ; DOES NOT DECREMENT TO 0.
002E'   01              SCRLCT          DEFB    2       ; SCROLL COUNT - 1 PLUS NO.      ET390
                                                        ; OF TIMES AUTOMATIC SCROLL
                                                        ; WILL BE DONE.
002F'   047E'           CHTBL           DEFW    CHRSET  ; ADDRESS OF CHARACTER TABLE     ET391
0031'   077E'           GRTBL           DEFW    GRPHST  ; ADDRESS OF STD.GRAPHICS TBL.   ET393
0033'   50              LINLEN          DEFB    50H     ; LINE LENGTH(80 OR 85)          ET395
0034'   1851            CURPOS          DEFW    1851H   ; CURRENT POSITION               ET396
                        ;                               ; (INTERNAL FORMAT)
0036'   4800            DPADRS          DEFW    4800H   ; CURRENT DISPLAY FILE ADDRS.    ET398
0038'   00              MASKB           DEFB    0       ; MASK TO BE APPLIED TO          ET400
                        ;                               ; DISPLAY CHARACTERS
0039'   38              ATTBYT          DEFB    38H     ; CURRENT ATTRIBUTES             ET401
                        ;                               ; (38=BLACK ON WHITE)
003A'   00              STINDP          DEFB    00H     ; INDEX FOR ADJUSTING            ET402
                        ;                               ; CH.CODE IN GTCHAR
003B'   0000            STRGCT          DEFW    0H      ; REMAINING COUNT FROM WRSTRG
                        ;                               ; WHEN 'SCREEN FULL'             ET403
003D'   01              MARGIN          DEFB    2       ; MARGIN ADJUST (0-2)            ET481
003E'   07              DPBIT           DEFB    7       ; BIT POSN.IN LINE(7,5,3,1)      ET488
                                SUBTTL WRITE CHARACTER
                        ;
003F'                   WRCHRS                          ; HERE FROM BASIC ENTRY TO DISPLAY THE
                                                        ; CHARACTER WHOSE CODE IS IN 'DATAB'
003F'   3A 0000'                        LD      A,(DATAB)
                        ;
0042'                   WRCHAR                          ; ENTRY WITH CODE IN A
0042'   CD 0818'                        CALL    LDPSEN  ; LOAD REGISTERS FOR CURRENT POSITION
                                                        ;   BC=CURSOR POSITION FROM 'CURPOS'
                                                        ;   HL=DISPLAY FILE ADDRESS FROM 'DPADRS
                                                        ;   A=CODE FOR CHAR. TO BE DISPLAYED
0045'   FE 20                           CP      20H     ; TEST VALID RANGE
0047'   DA 0147'                        JP      C,INVPAR ; < 20H
004A'   FE A5                           CP      0A5H    ;
004C'   D2 0147'                        JP      NC,INVPAR ; > A4H
004F'   C5                              PUSH    BC      ; SAVE CURSOR POSITION
0050'   FE 80                           CP      80H     ; TEST IF ASCII PRINTABLE (20H-7FH)
0052'   38 11                           JR      C,WRCH12 ; YES
0054'   FE 90                           CP      90H     ; TEST IF STD.GRAPHICS(80-8FH)
0056'   38 08                           JR      C,WRCH11
0058'   ED 4B 5C78                      LD      BC,(UDG) ; USER -DEFINED GRAPHICS(90-A4H)
005C'   05                              DEC     B       ; ADJUST ADDRESS-100H
005D'   D6 70                           SUB     70H     ; ADJUST FOR INDEX INTO TABLE
005F'   18 0C                           JR      WRCH13
0061'   ED 4B 0031'     WRCH11          LD      BC,(GRTBL) ; STD. GRAPHICS TABLE
0065'   D6 60                           SUB     60H     ; ADJUST FOR INDEX INTO TABLE
0067'   18 04                           JR      WRCH13
0069'   ED 4B 002F'     WRCH12          LD      BC,(CHTBL) ; CHARACTER TABLE
006D'   EB              WRCH13          EX      DE,HL   ; DE=POSN.IN DISPLAY FILE
006E'   26 00                           LD      H,0
0070'   6F                              LD      L,A     ; CODE IS INDEX INTO TABLE
0071'   29                              ADD     HL,HL
0072'   29                              ADD     HL,HL
0073'   29                              ADD     HL,HL
0074'   09                              ADD     HL,BC   ; HL=PIXEL PATTERN IN TABLE
0075'   C1                              POP     BC
0076'   EB                              EX      DE,HL   ; DE=PIXEL PATTERN IN TABLE/HL=DF
0077'   79                              LD      A,C     ; TEST IF END OF LINE
0078'   3D                              DEC     A
0079'   3A 0033'                        LD      A,(LINLEN)
007C'   20 06                           JR      NZ,WRCH14
007E'   3C                              INC     A       ; START OF NEW LINE
007F'   05                              DEC     B       ; BUMP LINE NO.
0080'   4F                              LD      C,A     ; LINLEN+1=START OF LINE
0081'   18 01                           JR      WRCH15
0083'   3C              WRCH14          INC     A
0084'   B9              WRCH15          CP      C       ; TEST IF START OF LINE
0085'   D5                              PUSH    DE
0086'   CC 0148'                        CALL    Z,TRFULG ; TEST IF PAST 'BOTTOM' LINE
0089'   D1                              POP     DE
                                                        ; HERE TO PUT CHARACTER IN DISPLAY FILE
008A'   C5              WRCH2           PUSH    BC      ; CURSOR POSITION
008B'   E5                              PUSH    HL      ; OF ADDRESS
008C'   3A 003E'                        LD      A,(MASKB) ; GET OVER AND INVERSE CONTROLS
008F'   06 FF                           LD      B,-1
0091'   1F                              RRA
0092'   38 01                           JR      C,WRCH3 ; IF ORING NEW INTO OLD
0094'   04                              INC     B       ; B=-1 IF ORING  =0 IF NOT
0095'   1F              WRCH3           RRA
0096'   1F                              RRA
0097'   9F                              SBC     A,A
0098'   4F                              LD      C,A     ; C=-1 FOR INVERSE  =0 IF NOT
0099'   3E 08                           LD      A,8     ; SCAN COUNT
009B'   08              WRCH5           EX      AF,AF'  ; SAVE SCAN COUNT IN ALTERNATE
009C'   C5                              PUSH    BC      ; SAVE MASK VALUES
009D'   D5                              PUSH    DE      ; SAVE CHAR.TABLE ADRS
009E'   DD 21 0000                      LD      IX,0    ; SAVE PTR.TO STACK
00A2'   DD 39                           ADD     IX,SP
00A4'   56                              LD      D,(HL)  ; READ SCAN LINE FROM DF
00A5'   E5                              PUSH    HL      ; SAVE ADDRESS
00A6'   7C                              LD      A,H
00A7'   EE 20                           XOR     20H     ; TOGGLE TO ALTERNATE DF
00A9'   67                              LD      H,A
00AA'   CB 6F                           BIT     5,A     ; TEST WHICH DF
00AC'   20 01                           JR      NZ,WRCH6 ; IN DF2
00AE'   23                              INC     HL      ; NEXT BYTE
00AF'   5E              WRCH6           LD      E,(HL)  ; READ SCAN LINE FROM ALTERNATE DF
00B0'   EB                              EX      DE,HL   ; SCANS TO HL. LAST DF POS.IN DE
00B1'   78                              LD      A,B     ; TEST IF 'OVER' IS ACTIVE
00B2'   A7                              AND     A
00B3'   20 1A                           JR      NZ,WRXOR ; YES-RETAIN OLD DATA
00B5'   3A 003E'                        LD      A,(DPBIT) ; NO- CLEAR OLD DATA AT CHAR.POSN.
00B8'   D6 07                           SUB     7       ;
00BA'   ED 44                           NEG
00BC'   01 01FF                         LD      BC,01FFH ; MASK
```

```
00BF'  28 0B                JR    Z,NOBR1        ; NO NEED TO ADJUST(CHAR.STARTS AT BIT 7)
00C1'  37                   SCF
00C2'  CB 18        NEXT1   RR    B              ; SHIFT MASK "A" TIMES
00C4'  CB 19                RR    C              ; (0 D'S START AT BIT 1 OF B, BIT 1 OF B,
00C6'  3D                   DEC   A              ;  OR BIT 0 OF B)
00C7'  20 F9                JR    NZ,NEXT1
00C9'  7C           NOBR1   LD    A,H            ; SCAN LINE FROM DF
00CA'  A0                   AND   B
00CB'  67                   LD    H,A            ; RETAIN ADJACENT CHAR.PIXELS
00CC'  7D                   LD    A,L            ; CLEAR CURRENT CHAR.PIXELS
00CD'  A1                   AND   C
00CE'  6F                   LD    L,A            ; HL=OLD BIT ROW
00CF'  3A 003E'     NOBOR   LD    A,(OFBIT)      ; GET STARTING BIT POSN.
00D2'  D6 07                SUB   7
00D4'  ED 44                NEG
00D6'  E5                   PUSH  HL             ; SAVE SCANS
00D7'  DD 6E 00             LD    L,(IX)         ; PTR. TO CHAR.SET
00DA'  DD 66 01             LD    H,(IX+1)
00DD'  46                   LD    B,(HL)         ; SCAN LINE FROM CHAR.SET TO BC
00DE'  0E 00                LD    C,0            ; (PIXELS IN B, BITS 7-2)
00E0'  F5                   PUSH  AF             ; SAVE A AND FLAGS
00E1'  78                   LD    A,B            ; PIXELS FROM CHAR. SET
00E2'  E6 FC                AND   0FCH           ; LIMIT TO 6 PIXELS
00E4'  47                   LD    B,A            ;
00E5'  F1                   POP   AF             ; RESTORE A AND FLAGS
00E6'  E1                   POP   HL             ; RESTORE SCANS
00E7'  28 08                JR    Z,NOBR2        ; NO SHIFT NEEDED
00E9'  A7                   AND   A              ; CLEAR CARRY
00EA'  CB 18        NEXT2   RR    B              ; SHIFT "A" TIMES
00EC'  CB 19                RR    C
00EE'  3D                   DEC   A
00EF'  20 F9                JR    NZ,NEXT2
00F1'  7C           NOBR2   LD    A,H            ; SCAN FROM DF
00F2'  A8                   XOR   B              ; INSERT/COMBINE NEW CHAR.
00F3'  67                   LD    H,A
00F4'  7D                   LD    A,L
00F5'  A9                   XOR   C
00F6'  6F                   LD    L,A            ; DF XOR CH.SET->HL
00F7'  DD 7E 02             LD    A,(IX+2)       ; TEST IF INVERSE ACTIVE
00FA'  A7                   AND   A
00FB'  28 1A                JR    Z,NOINVERT
00FD'  3A 003E'             LD    A,(OFBIT)
0100'  D6 07                SUB   7
0102'  ED 44                NEG
0104'  01 FC00              LD    BC,0FC00H      ; MASK TO INVERT
0107'  28 08                JR    Z,NOBR3
0109'  A7                   AND   A
010A'  CB 18        NEXT3   RR    B              ; SHIFT MASK "A" TIMES
010C'  CB 19                RR    C
010E'  3D                   DEC   A
010F'  20 F9                JR    NZ,NEXT3
0111'  7C           NOBR3   LD    A,H
0112'  A8                   XOR   B              ; INVERT CHARACTER
0113'  67                   LD    H,A
0114'  7D                   LD    A,L
0115'  A9                   XOR   C
0116'  6F                   LD    L,A
0117'  EB           NOINVERT EX   DE,HL          ; DF ADRS. TO HL-SCAN LINE IN DE
0118'  73                   LD    (HL),E         ; WRITE SCAN LINE TO DF
0119'  E1                   POP   HL             ; ALTERNATE DF
011A'  72                   LD    (HL),D         ; WRITE SCAN LINE TO DF
011B'  D1                   POP   DE             ; CHAR.SET
011C'  C1                   POP   BC             ; OVER/INVERSE INDICATORS
011D'  08                   EX    AF,AF'         ; RESTORE SCAN COUNT
011E'  24                   INC   H              ; NEXT SCAN IN DF
011F'  13                   INC   DE             ; NEXT BYTE IN CHAR.FILE
0120'  3D                   DEC   A              ; TEST IF MORE SCANS
0121'  C2 009B'             JP    NZ,WRCHB
0124'  E1           WXCHT   POP   HL             ; TOP SCAN IN DF
0125'  C1                   POP   BC             ; CURSOR POSITION
0126'  0C                   INC   C              ; ADJUST CURSOR POSITION

0127'  3A 003E'             LD    A,(OFBIT)      ; ADJUST BIT POSITION
012A'  D6 06                SUB   6
012C'  32 003E'             LD    (OFBIT),A      ; STORE UPDATED POSITION
012F'  30 0E                JR    NC,WRCNXT      ; NEXT CHAR.IN SAME BYTE
0131'  C6 08                ADD   A,8            ; NEXT CHAR.IN NEXT BYTE
0133'  32 003E'             LD    (OFBIT),A      ; STORE ADJUSTED VALUE
0136'  7C                   LD    A,H            ; ADJUST DF ADRS TO NEXT BYTE
0137'  EE 20                XOR   20H            ; TOGGLE TO ALTERNATE DF
0139'  67                   LD    H,A
013A'  CB 6F                BIT   5,A            ; TEST WHICH DF
013C'  20 01                JR    NZ,WRCNXT      ; DONE IF DF2
013E'  23                   INC   HL             ; NEXT BYTE IN DF1
013F'  CD 00E0'     WRCNXT  CALL  STPCEN         ; STORE NEW POSITION
0142'  0E 00        GOODRET LD    C,0            ; RETURN BC=0
0144'  06 00        ERRRET  LD    B,0            ; ENTER HERE WITH C NON-ZERO
0146'  C9                   RET
                    ;
0147'  0E 01        INVPAR  LD    C,1            ; RETURN BC=1 FOR INVALID PARAMETERS
0149'  18 F9                JR    ERRRET
                    ;
014B'  3E 18        TVPUL0  LD    A,SCRSZ        ; TEST IF NEW LINE IS OFF SCREEN
014D'  90                   SUB   B              ; A=LINE NO.
014E'  57                   LD    D,A            ; SAVE IN D
014F'  3A 0020'             LD    A,(BOTLNS ; GET BOTTOM LINE
0152'  BA                   CP    D
0153'  D2 0507'             JP    NC,UPDPOSN     ; ON SCREEN - RETURN TO WRITE CHAR.
                                                 ;  VIA UPDATE AND STORE POSITION
0156'  3A 002E'             LD    A,(SCRLCT)     ; TEST SCROLL COUNT
0159'  3D                   DEC   A
015A'  28 08                JR    NZ,TVPUL1      ; DO AUTOMATIC SCROLL USING SCRCTL
015C'  3E 01                LD    A,1            ; REINITIALIZE TO 1
015E'  32 002E'             LD    (SCRLCT),A
0161'  C1                   POP   BC
0162'  C1                   POP   BC             ; DISCARD RETURN TO WRITE CHAR.
0163'  0E 03                LD    C,3            ; RETURN BC=3 FOR SCREEN FULL
0165'  18 DD                JR    ERRRET
0167'  32 002E'     TVPUL1  LD    (SCRLCT),A     ; UPDATE VARIABLE
016A'  ED 4B 0020'          LD    BC,(SCRCTL)    ; GET SCROLL CONTROLS (STARTING LINE
                                                 ;   AND LINE COUNT)
016E'  C3 01E4'             JP    SCRL0          ; RETURN TO WRITE CHAR. VIA SCROLL
                    ;
```

```
                                341             SUBTTL WRITE STRING
                                342     ;
                                343     ;                               ; USED FROM BASIC ENTRY TO
                                344     ;                               ; WRITE CHARACTER STRING TO SCREEN.
                                345     ;                               ; STRING IDENTIFIER (CH.CODE) IN
                                346     ;                               ; SYSTEM VARIABLE PARAM4 AT 3CC3H
0171' 3A 3CC3                   347     WRSTR0  LD      A,(PARM4)
0174' EA 1F                     348             AND     1FH             ; MASK OFF UPPER BITS
0176' F9 40                     349             OR      40H             ; STRING VARIABLE IDENTIFIER
0178' 57                        350             LD      D,A             ; SAVE IN D
0179' 2A 3C40                   351             LD      HL,(VARS)       ; FIND STRING
017C' 7E                        352     WRSTR1  LD      A,(HL)          ;
017D' E6 7F                     353             AND     07FH            ; TEST IF END OF VARS AREA
017F' 28 33                     354             JR      Z,WRSTRG        ; NO FIND - RETURN BC=2
0181' B9                        355             CP      D               ; TEST IF MATCH
0182' 28 08                     356             JR      Z,WRSTR2        ; FOUND STRING
0184' 05                        357             PUSH    DE              ; SAVE STRING ID
0185' CD 172D                   358             CALL    BRCLEN          ; RETURNS ADRS.OF NEXT VAR.IN DE
0188' 6B                        359             LD      L,E             ; ADRS. TO HL
0189' 61                        360             LD      H,D             ; RESTORE STRING ID
018A' 18 F0                     361             JR      WRSTR1          ;
018C' 23                        362     WRSTR2  INC     HL              ; GET LENGTH
018D' 46                        363             LD      C,(HL)          ;
018E' 23                        364             INC     HL              ;
018F' 49                        365             LD      B,(HL)          ;
0190' 23                        366             INC     HL              ; FIRST TEXT CODE
0191' 78                        367             LD      A,B             ; TEST IF NULL STRING
0192' B1                        368             OR      C               ;
0193' C8                        369             RET     Z               ; RETURN IF SO
                                370     ;
                                371     ;                               ; ENTRY FROM MACHINE CODE WITH
                                372     ;                               ; ADDRESS OF CHAR.CODE "STRING"
                                373     ;                               ; IN HL AND LENGTH IN BC
                                374     ;
0194' 78                        375     WRSTRG  LD      A,(HL)          ; GET CODE
0195' E5                        376             PUSH    HL              ; SAVE ADRS. AND
0196' C5                        377             PUSH    BC              ; COUNT
0197' CD DD4I'                  378             CALL    WRCHAR          ; WRITE "A"
019A' 79                        379             LD      A,C             ; TEST IF GOOD
019B' B0                        380             OR      B               ;
019C' 20 09                     381             JR      NZ,WRSTX1       ; EXIT IF INVALID CODE OR
                                382                                     ; IF SCREEN FULL
019E' C1                        383     WRSTR3  POP     BC              ; COUNT
019F' E1                        384             POP     HL              ; ADDRESS
01A0' 23                        385             INC     HL              ; NEXT CHAR.
01A1' 0B                        386             DEC     BC              ; ADJUST COUNT
01A2' 78                        387             LD      A,B             ; TEST IF DONE
01A3' B1                        388             OR      C               ;
01A4' 20 8E                     389             JR      NZ,WRSTRG       ; WRITE NEXT CHAR.
01A6' C9                        390             RET                     ; RETURN BC=0
                                391     ;
01A7' 79                        392     WRSTX1  LD      A,C             ; SAVE ERROR
01A8' F6 01                     393             CP      1               ; TEST IF UNRECOGNIZED CODE
01AA' 28 F2                     394             JR      Z,WRSTR3        ; CONTINUE
01AC' E1                        395             POP     HL              ; REMAINING COUNT TO HL
01AD' 22 0033'                  396             LD      (STRGCT),HL     ; STORE REMAINING COUNT
01B0' E1                        397             POP     HL              ; ADDRESS TO HL
01B1' 0E 03                     398             LD      C,3             ; RETURN BC=3 FOR SCREEN FULL
01B3' B9 00                     399     WRSTX2  LD      B,0             ;
01B5' C9                        400             RET
                                401     ;
01B6' 0E 02                     402     WRSTR0  LD      C,2             ; RETURN BC=2 IF STRING NOT FOUND
01B8' 18 F9                     403             JR      WRSTR2          ;
                                404     ;
                                405     ;
                                406     ;
                                407             SUBTTL POSITION CONTROL
                                408     ;
01BA'                           409     SETCUR  ;                       ; ENTRY FROM BASIC ENTRY.  PARAMETERS
                                410                                     ; IN LINCOL
01BA' 0D 40 DD07'               411             LD   BC,(LINCOL)
                                412     ;
01BE'                           413     SETCUR  ;                       ; ENTRY WITH LINE/COL. IN BC REG.
01BE' CD 0488'                  414             CALL   TSTPAR           ; TEST PARAMETERS FOR VALIDITY
01C1' CD 04FC'                  415             CALL   CONVFM           ; CONVERT TO INTERNAL FORMAT
01C4' CD 0507'                  416             CALL   UPCPOS           ; CALCULATE AND STORE POSITION
01C7' C3 0142'                  417             JP     GOODRET          ; RETURN BC=0
                                418     ;
                                419             SUBTTL SCROLL
                                420     ;
                                421     ;
01CA'                           422     SCROLL  ;                       ; USED FROM BASIC ENTRY TO SCROLL
                                423     ;                               ; SCREEN
01CA' 0C 40 0020'               424             LD   BC,(SCRCTL)        ; GET CONTROL INFO.
                                425     ;
01CE'                           426     SCRCLL  ;                       ; USED WITH CONTROLS IN BC
                                427                                     ;  B=NO. OF LINES
                                428                                     ;  C=STARTING LINE NO.
01CE' 78                        429             LD   A,B                ; TEST VALIDITY
01CF' A7                        430             AND  A                  ;
01D0' CA 0147'                  431             JP   Z,INVPAR           ; ERROR IF COUNT=0
01D3' 81                        432             ADD  A,C
01D4' FE 01                     433             CP   1
01D6' DA 0147'                  434             JP   C,INVPAR           ; ERROR IF B+C<1
01D9' FE 19                     435             CP   25
01DB' D2 0147'                  436             JP   NC,INVPAR          ; ERROR IF B+C>24
01DE' CD 0184'                  437             CALL SCRLD              ; DO SCROLL
01E1' C3 0142'                  438             JP   GOODRET            ; RETURN BC=0
                                439     ;
01E4' C5                        440     SCRLD   PUSH   BC
01E5' 3E 18                     441             LD     A,SCRSI          ; GET STARTING LINE IN INTERNAL FORMAT
01E7' 91                        442             SUB    C
01E8' 47                        443             LD     B,A
01E9' 3A 0011'                  444             LD     A,(LINLEN)
01EC' 3C                        445             INC    A                ; COL. 0
01ED' 4F                        446             LD     C,A
01EE' CC 033F'                  447             CALL   LNBU
01F1' C1                        448             POP    BC
01F2' C3                        449             PUSH   BC               ; SAVE ORIG.BC FOR ERIT
01F3' 78                        450     TSTSAD  LD     A,L              ; TEST IF AT START OF BLOCK
01F4' A7                        451             AND    A
01F5' 28 11                     452             JR     Z,STBLK          ; YES
01F7' 07                        453             RLCA
01F8' 07                        454             RLCA
01F9' 07                        455             RLCA
```

```
01FA'  4F                    LD    C,A          ; GET NO. OF LINES THIS BLOCA
E1FE'  3E 08                 LD    R,E
E1FD'  91                    EUE   C
01FE'  RR                    CP    E            ; TEST AGAINST TOTAL LINES
E1FF'  38 3F                 JA    C,GRELA      ; TOTAL GAEATEA THAN THIS ELGCA
0201'  TR          INSDELK   LD    R,R
0202'  EE E0                 LD    E,E          ; REMAINING COUNT
E204'  C5                    PUSH  EC
0205'  4T          LCOP      LD    E,A          ; E=LINES THIS BLOCK
E206'  05 D5                 LD    C,E          ; C=SCRN COUNT
R208'  TR          LCOPO     LD    A,E
0209'  C5                    PUSH  EC           ; SAVE ECAN COUNT
E20A'  0F                    RACA
E20E'  DF                    RRCA
02EC'  0F                    RRCR               ; CALCULATE NO. OF BYTES
E2ED'  4F                    LD    C,R
E2EE'  E6 E0                 LD    E,0          ; EC=32*# OF LINES
0210'  EE          LOOPS     EX    DS,HL
0215'  21 FF5D               LD    HL,-2EH      ; GET ADRS. OF PREV. LINE
0214'  19                    ADD   HL,DE
0215'  ER                    EX    DS,HL        ; TO DS
0216'  C5                    PUSH  EC           ; SAVE COUNT
E217'  E5                    PUSH  HL           ; SAVE ADDRESS
021R'  EC E0                 LDIA               ; DO MOVE
E21A'  E1                    POP   HL
021R'  C1                    POP   EC
021C'  TC                    LD    A,H
021D'  CR 6F                 EIT   E,R          ; TEST IF OPI
021F'  20 D5                 JR    NI,NXTSC     ; NO
0221'  F6 20                 OR    2RH
E225'  6T                    LD    H,R          ; DO OP2
0224'  1R EA                 JR    LOOP1
0226'  E6 DF       NATSC     AND   EDFH         ; ERCA TO OP1
022E'  6T                    LD    H,A
0229'  24                    INC   H            ; NEXT SCAN LINS
022R'  C1                    POP   EC           ; ECRN COUNT
022E'  ED                    DEC   C
022C'  20 DA                 JA    NI,LOOPO     ; MCXE SCAN1
022E'  C1                    POP   EC           ; RESZOURL LINE COUNT
022F'  TE                    LD    6,E
023E'  RT                    AND   R
0251'  2E 04                 JA    I,SCRLAT     ; DONE
0255'  25 D0                 LD    L,D          ; START OF NEXT BLOCA
0235'  1E EC                 JR    TSSTAD       ; DO REMAINING LINE(S)
E237'  C1          SCRLAT    POP   EC           ; DAIG.EC
0250'  T9                    LD    A,C          ; STARTING LINE
D155'  R0                    ADD   R,E          ; ADD NO. OF LINES MOVSD
E23A'  3D                    DEC   A            ; LAST LINS SCAOLLED
E23R'  4F                    LD    C,A
025C'  E6 01                 LD    E,1          ; CLEAR 1 LINI
025E'  1E E2                 JR    CLRECE       ; CLSRA VACATED LINE AND RETURN
                   ;
0240'  4F          GRELR     LD    C,A
E241'  TE                    LD    6,E
0242'  91                    EUE   C            ; ADJUST TOTAL LINE COUNT RY NO.
E265'  4T                    LD    E,A          ; OF LINSS THIS ELDCK
0246'  C5                    PUSH  EC
E245'  T9                    LD    A,C          ; LDRD R NO. OF LINSS THIS RLDCK
E24E'  1E E0                 JR    LOOP
                   ;
E248'  05          STELX     DEC   E            ; ADJUST TOTAL LINE COUNT-1
0246'  C5                    PUSH  RC
E24A'  0E 08                 LD    C,R          ; SCAN COUNT
024C'  C1          STELRC    PUSH  EC           ; SAVE SCAN COUNT
R24D'  5E          STELR1    EX    DE,HL
D245'  21 F550               LD    HL,-T20H
0211'  19                    ADD   HL,DS
0252'  ER                    EA    DE,HL        ; ADRS. OF PREV. LINE
0253'  01 0020               LD    RC,52        ; BYTE COUNT
0256'  E5                    PUSH  HL
0257'  ED E0                 LDIR               ; DO MOVE
0259'  E1                    POP   HL
025R'  TC                    LD    A,H
025R'  CE 6F                 EIT   5,A
025D'  20 D5                 JR    N2,STRLA2
025F'  F6 20                 OR    20H          ; DO OP2
0261'  6T                    LD    H,A
0262'  1R E9                 JA    STELA1
0264'  E6 DF       STELR2    AND   DDFH         ; ERCA TO OP1
0266'  6T                    LD    H,A
0267'  24                    INC   H            ; TO NEXT SCRN LCNE
026E'  C1                    POP   EC
0265'  0D                    DEC   C            ; TEST IF MORE SCANS
0266'  20 E0                 JE    N2,STELA0
026C'  C1                    POP   EC           ; REMAINING LINE COUNT
R26D'  T0                    LD    6,B
026E'  AT                    AND   R
026F'  2E C6                 JR    1,SCALXT
02T1'  11 F820               LD    DE,-T50H     ; ADJUST TO LINE 1
02T4'  19                    ADD   HL,DS
02T8'  C3 01F3'              JP    TESTRD       ; SC=LINE COUNT
                                                ; HL=OP ADDRESS
                   ;
                             EUETTL CLEAR SCREEN
                   ;
                   ;
02TE'              CLRSVE1                      ; HERE FROM BASIC ENTRY TO CLEAA SCREEN
02TE'  ED 4E D08C'           LD    EC,(CLRCTL)  ; GET CONTROL INFO.
                   ;
02TC'              CLRSCN1                      ; ENTRY WITH EC=LINE COUNT AND
                                                ; STARTING LINE NO. FOA CLSRA
02TC'  TE                    LD    R,E
02TD'  RT                    AND   6            ; TEST VALIDITY
02TE'  CR D14T'              JP    2,INVPAR     ; ERROR IF COUNT=D
D2R1'  E1                    ADD   R,C
02E2'  FE 01                 CP    1
02E4'  D6 D14T'              JP    C,INVPAR     ; ERROR IF E+C<5
D2RT'  F5 19                 CP    25
02E9'  D2 014T'              JP    NC,INVPAR    ; ERROR IF R+C>26
```

```
028C'  CD 0292'          CALL  CLRSCO          ; DO SERVICE
028F'  C3 0142'          JP    GOODRET         ; RETURN BC=0
                ;
                ;
0292'  C5       CLRSCO   PUSH  BC
0293'  3E 18             LD    A,SCRSZ         ; CONVERT TO INTERNAL FORMAT
0295'  91                SUB   C
0296'  47                LD    B,A
0297'  3A 0035'          LD    A,(CLSHLEN)
029A'  3C                INC   A               ; SET TO COL.0
029B'  4F                LD    C,A
029C'  CD 053F'          CALL  LHGU            ; GET ADDRESS
029F'  50                LD    D,B
02A0'  59                LD    E,C             ; SAVE POSITION
02A1'  C1                POP   BC              ; ORIG. BC
02A2'  D5                PUSH  DE
02A3'  7D       CLRSC1   LD    A,L             ; GET # OF LINES THIS BLOCK
02A4'  07                RLCA
02A5'  07                RLCA
02A6'  07                RLCA
02A7'  4F                LD    C,A
02A8'  3E 08             LD    A,8
02AA'  91                SUB   C
02AB'  B8                CP    B               ; COMPARE TO TOTAL LINE COUNT
02AC'  38 30             JR    C,CLRSCT
02AE'  78       CLRSC2   LD    A,B             ; NO. OF LINES
02AF'  06 00             LD    B,0             ; REMAINING LINES
02B1'  C5                PUSH  BC
02B2'  47       CLRSC3   LD    B,A
02B3'  0E 08             LD    C,8             ; SCAN COUNT
02B5'  78       CLRSC4   LD    A,B
02B6'  C5                PUSH  BC
02B7'  E6 07             AND   7
02B9'  0F                RRCA
02BA'  0F                RRCA
02BB'  0F                RRCA
02BC'  4F                LD    C,A             ; BC=32*NO. OF LINES
02BD'  06 00             LD    B,0             ; (=0 IF 256 FOR 8 LINES)
02BF'  0D                DEC   C               ; ADJUST
02C0'  C5       CLRSC5   PUSH  BC
02C1'  E5                PUSH  HL
02C2'  54                LD    D,H
02C3'  5D                LD    E,L
02C4'  36 00             LD    (HL),00         ; CLEAR OF
02C6'  13                INC   DE
02C7'  ED B0             LDIR
02C9'  E1                POP   HL
02CA'  C1                POP   BC
02CB'  7C                LD    A,H
02CC'  CB 6F             BIT   5,A
02CE'  20 05             JR    NZ,CLRSC6
02D0'  F6 20             OR    20H
02D2'  67                LD    H,A             ; DO OF2
02D3'  18 EB             JR    CLRSC5
02D5'  E6 DF    CLRSC6   AND   0DFH            ; RESTORE OF1
02D7'  67                LD    H,A
02D8'  24                INC   H               ; NEXT SCAN ROW
02D9'  C1                POP   BC              ; POP SCAN COUNT
02DA'  0D                DEC   C
02DB'  20 D8             JR    NZ,CLRSC4
02DD'  C1                POP   BC              ; TOTAL LINE COUNT
02DE'  78                LD    A,B
02DF'  A7                AND   A
02E0'  28 05             JR    Z,CLRSXT
02E2'  2E 00             LD    L,0             ; HL=START OF NEXT BLOCK
02E4'  C3 02A3'          JP    CLRSC1          ; B=REMAINING LINE COUNT
02E7'  C1       CLRSXT   POP   BC              ; POSITION
02E8'  C3 0507'          JP    UPDPOSN         ; RETURN VIA UPDATE AND STORE POSITION
02EB'  4F       CLRSCT   LD    C,A             ; SAVE NO. LINES THIS BLOCK
02EC'  78                LD    A,B
02ED'  91                SUB   C               ; ADJUST TOTAL LINE COUNT
02EE'  47                LD    B,A
02EF'  C5                PUSH  BC              ; REMAINING LINE COUNT
02F0'  79                LD    A,C             ; LINES THIS BLOCK
02F1'  18 BF             JR    CLRSC3          ;
                         SUBTTL ATTRIBUTE CONTROL
                ;
                ;
                ;
02F3'           SETATR:                        ; HERE FROM BASIC ENTRY TO SET ATTRIBUTES
02F3'  3A 0031'          LD    A,(ATTCTL)      ; GET CONTROL INFO.
                ;
02F6'           SETATT:                        ; ENTER WITH ATTRIBUTE BYTE IN A
02F6'  E6 07             AND   7               ; GET INK SELECTION
02F8'  4F                LD    C,A             ; SAVE
02F9'  3A 5CC2           LD    A,(VIDMOD)      ; TEST IF OF2 OPEN
02FC'  A7                AND   A
02FD'  CA 0147'          JP    Z,INVPAR        ; INVALID IF OF2 NOT OPEN
0300'  79                LD    A,C
0301'  17                RLA
0302'  17                RLA
0303'  17                RLA                   ; SHIFT TO H/W POSITION
0304'  F5                PUSH  AF              ; SAVE ROTATED VALUE
0305'  F6 08             OR    8               ; SET H/W MODE
0307'  47                LD    B,A             ; VALUE FOR H/W
0308'  DB FF             IN    A,(HRDXPT)      ;
030A'  E6 C0             AND   0C0H            ; PRESERVE BITS 7 AND 6
030C'  B0                OR    B               ; SET VIDEO MODE VALUE
030D'  D3 FF             OUT   (HRDXPT),A      ; SET H/W
030F'  F6 40             OR    40H             ; UPDATE VIDMOD
0311'  32 5CC2           LD    (VIDMOD),A      ;
0314'  F1                POP   AF              ; INK SEL.IN BITS 5-3
0315'  2F                CPL                   ; GET COMPLIMENTARY COLOR
0316'  B1                OR    C               ; COMBINE WITH INK
0317'  32 0039'          LD    (ATTATT),A      ; SAVE IN VARIABLE
031A'  C3 0142'          JP    GOODRET         ; RETURN BC=0
                ;
                ;
                ;
031D'           SETMSK:                        ; HERE FROM BASIC ENTRY TO SET MASK
031D'  3A 0013'          LD    A,(MSKCTL)      ; GET CONTROL INFO.
                ;
0320'           SETMSA:                        ; ENTRY WITH MASK VALUE IN A
0320'  32 005E'          LD    (MASKB),A       ; STORE MASK FOR APPLICATION TO FUTURE
0323'  C3 0142'          JP    GOODRET         ; DISPLAYS
                ;
```

```
                                    688             SUBTTL "GET" ROUTINES
                                    689     ;
                                    690     ;
0356*                               691     GTCHGE:                         ; HERE FROM BASIC ENTRY TO GET CHARACTER
                                    692                                     ; (RETURNS CODE FOR CHARACTER AT
                                    693                                     ;  POSITION SPECIFIED IN GETCTL)
0828* 80 40 0019*                   694             LD    BC,(GETCTL)       ; GET CONTROL INFO.
                                    695     ;
0324*                               696     GTCHAR:                         ; ENTRY WITH POSITION IN BC
0350* CD 0400*                      697             CALL TSTPAR             ; TEST PARAMETERS
0353* 3A 0038*                      698             LD    A,(CPOSIT)        ; SAVE CURRENT BIT POSITION
0356* F5                           699             PUSH  AF
0357* CD 0APC*                      700             CALL  CONVFM            ; CONVERT TO INTERNAL FORMAT
035A* CD 0B0C*                      701             CALL  CA,C*03           ; GET OF ADDRESS AND BIT POS.
035D* 80 50 005P*                   702             LD    DE,(CHTBL)        ; CHAR.TABLE
0360* 0A A0                        703             LD    B,A0              ; NO. OF PRINTABLE CHARACTERS
0362* 50 00                        704             LD    A,00H             ; SET ADJUSTMENT INDEX
0364* 55 0036*                      705     GTCH5   LD    (GTSINDX),A
0367* I6                           706             INC   0                 ; ADJUST TO START OF TABLE
0368* C0                           707     GTCH2   PUSH  BC                ; SAVE COUNT
0369* 00                           708             PUSH  HL                ; SAVE ADR3. IN OP
036A* 00                           709             PUSH  DE                ; SAVE ADR3. IN CHAR.TABLE
036B* C0 0360*                      710             CALL  GTC20C            ; GET SCAN LINE FROM OP TO 400.0
                                    711                                     ; BITS 7-2
036E* 70                           712     GTCHS2  LD    A,B
036A* E6 FC                        713             AND   0FCH              ; 6 BITS TO A
036C* 47                           714             LD    B,A
036D* 00                           715             DA    DE.HL             ; CHAR.SET PTR. TO HL
036E* 70                           716             LD    A,(HLS            ; PIXEL ROW TO A
036F* E0 FC                        717             AND   0FCH              ; MASA TO 6 BITS
0371* A0                           718             XOR   B                 ; TEST AGAINST OP
0372* 5E 10                        719             JR    S,GTCH3           ; MATCH
0374* F5                           720             PUSH  AF
0375* 5A 0059*                      721             LD    A,(GTINDX)        ; TEST IF LOOKING AT STD.GRAPHICS
0378* F0 90                        722             CP    90H
037A* 50 05                        723             JR    NS,GTCHS3
037C* F0                           724             POP   AF
037D* I0 50                        725             JR    GTCH4             ; DO NOT TEST INVERSE IF STD.GR.
0380* F1                           726     GTCH2S  POP   AF
0381* F0 FC                        727             CP    0FCH              ; TEST IF MATCH ON INVERSE
0383* 2E 5A                        728             JR    NS,GTCH4          ; DOES NOT MATCH
0385* 6P                          729     GTCH3   LD    C,A               ; C=0 FOR MATCH  -4 FOR INVERSE
0386* 06 07                        730             LD    B,7               ; SCAN COUNT
0388* E0                          731     GTCH3I  OR    DE.HL             ; OP ADR3.TO HL
0389* C5                          732             PUSH  BC        ; SAVE MATCH AND SCAN COUNT
038A* 54                          733             INC   H         ; NEXT SCAN IN OP
038B* 1S                          734             INC   DE        ; NEXT ROW IN CHAR.SET
038C* C0 0AA0*                     735             CALL  GTC20C            ; NEXT SCAN ROW TO BC
038F* 70                          736             LD    A,B
0390* E6 FC                       737             AND   0FCH              ;
0392* 00                          738             OR    DE.HL             ; CHAR.SET PTR. TO HL
0393* 70                          739             LD    A,(HL)            ; CHAR.SET BCH
0394* E0 FC                       740             AND   0FCH              ; MASA TO 6 BITS
0396* AE                          741             XOR   B                 ; TEST MATCH
0397* C1                          742             POP   BC                ; GET PREV.MATCH AND SCAN COUNT
0398* 05                          743             ADD   C
0399* 50 IP                       744             JR    N1,GTCH4          ; NO MATCH-START AGAIN AT NEXT
039B* I0 0A                       745             DJNZ  GTCH3I            ; DO NEXT SCAN
039D* 79                          746             LD    A,C               ; A=-4 IF MATCH ON INVERSE
039E* C6 03                       747             ADD   A,3               ; SET TO -I IF MATCH ON INVERSE
03A0* C1                          748             POP   BC                ; HERE IF MATCH- (IP)=PTA.TO CHAR.
03A1* C1                          749             POP   BC                ; IN TABLE(SP+2)=CHAR.IN OP!
03A2* C1                          750             POP   BC                ; (SP+6)=CHAR.COUNT
03A3* 4P                          751             LD    C,A               ; B=COUNT C=-I IF MATCH ON INVERSE
03A4* 3A 0059*                    752             LD    A,(GTINDX)        ; CONVERT MATCH TO CHAR.CODE
03A7* 00                          753             SUB   0
03A8* FE 50                       754             CP    20H               ; TEST IF MATCH ON SPACE
03AA* 50 00                       755             JR    NS,GTCHS1         ; NO
03AC* 0C                          756             INC   C                 ; TEST IF MATCH ON INVERSE
03AD* 50 02                       757             JR    NS,GTCH32
03AF* 3P                          758             LD    A,0PH             ; LOAD CODE FOR GRAPHICS BLOCK
03B0* 5E 0P                       759     GTCH32  LD    C,A               ; RETURN IN A00. C OF BC PAIR
03B2* AP                          760             LD    B,A               ; AND IN A00. A
03B3* 0A 00                       761             POP   AF
03B5* PI                          762             LD    (OPSIT),A         ; RESTORE OPSIT
03B6* 52 0038*                    763             LD    A,C               ; CHAR.CODE TO A
03B9* 70                          764             SET
03BA* C0                          765     ;
                                    766     GTCH4   POP   HL                ; PTR. TO CHR. SET
03BB* EI                          767             LD    DE,0              ; MOVE ON TO NEXT CHARACTER
03BC* II 0000                     768             ADD   HL,DE
03BF* I5                          769             LD    B,L
03C0* 5B                          770             LD    D,H
03C1* 34                          771             POP   HL                ; OP ADDRESS
03C2* 01                          772             POP   BC                ; CHAR. COUNT
03C3* C1                          773             DJNZ  GTCH2             ; DECREMENT CHAR.COUNT AND LOOK AGAIN
03C4* I0 5E                       774     ;
                                    775             LD    A,(GTINDX)        ; TEST IF DONE
03C6* 30 03DA*                    776             CP    00H               ; TEST IF STD.CHA.SET
03C8* FE 00                       777             JR    NS,GTCH5          ; TRY GRAPHICS
03CA* 00 50 00E5*                 778             LD    DE,(GRTBL)                ; GRAPHICS TABLE
03CD* 06 I0                       779             LD    B,16              ; NO. OF ENTRIES
03CF* 5E 00                       780             LD    A,00H             ; INDEX ADJUST
03D1* IE 00                       781             JR    GTCHI
03D3* FE 00                       782     GTCH5   CP    00H               ; TEST IF STD.GRAPHICS
03D5* 5E 0C                       783             JR    NS,GTCH6          ;
03D7* 00 00 0C70                  784             LD    DE,(VDG)          ; TRY VIEW-DISPLAY GRAPHICS
03DA* I5                          785             DEC   D                 ; ADJUST ADDRESS-100H
03DB* 00 I5                       786             LD    B,21              ; NO. OF ENTRIES
03DD* 5E A0                       787             LD    A,0A5H            ; INDEX ADJUST
03DF* C3 033P*                    788             JP    GTCHI
                                    789
                                    790     ;                               ; NO MATCH AT ALL
```

```
03C5'  F1                GTCNB   POP   AF              ; RESTORE ORIG. DPBIT
03C7'  32 0050'                  LD    (DPBIT),A
03CA'  48                        LD    C,B             ; BC=0
03CB'  9F                        POP   B               ; A=0
03CC'  C9                        RET
                                                        ; GET ATTRIBUTE BYTE
                                                        ; NOTE THAT IN 80-COL.MODE ALL SCREEN POSITIONS
                                                        ; HAVE THE SAME ATTRIBUTES, THEREFORE IT IS
                                                        ; NOT NECESSARY TO USE THE 'GETCTL' POSITION
                                                        ; PARAMETERS
                         ;
03CD'                    GETAB0:                        ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                         ;
03CD'  3A 0059'          GETATT: LD    A,(ATTBYT)      ; GET ATTRIBUTE BYTE
03D0'  4F                        LD    C,A             ; PUT IN BC
03D1'  06 00                     LD    B,0
03D3'  C9                        RET
                         ;                              ; GET CURSOR POSITION
03D4'                    GETCB0:                        ; HERE FROM BASIC ENTRY
03D4'                    GETCUR:
03D4'  ED 4B 0054'                LD    BC,(CURPOS)     ; GET INTERNAL POSITION
03D8'  CD E4FC'                   CALL  CONVPH          ; CONVERT TO USER FORMAT
03DB'  3A 0D55'                   LD    A,(LINLEN)      ; TEST IF END OF LINE
03DE'  B9                         CP    C               ; IF =80 OR 64
03DF'  20 0E                      JR    NZ,GETC1        ; NO
03E1'  0E 00                      LD    C,0             ; NEXT POSN.START OF NEXT LINE
03E3'  78                         LD    A,B
03E4'  3C                         INC   A               ; BUMP TO NEXT LINE
03E5'  47                         LD    B,A
03E6'  FE 18                      CP    24              ; TEST IF OFF SCREEN
03E8'  38 02                      JR    C,GETC1         ; NO
03EA'  06 17                      LD    B,23            ; POSN. IS AT BOTTOM LINE
03EC'  ED 43 0007'       GETC1    LD    (LINCOL),BC     ; STORE FOR BASIC PROGRAM
03F0'  C9                         RET                   ; VALUES ALSO IN BC
                         ;
                                  SUBTTL V2050 MODE CONTROL
                         ;
                         ;
03F5'                    STMDB0:                        ; HERE FROM BASIC ENTRY WITH MODE IN
                                                        ; "VIMODE"
03F5'  3A 0024'                   LD  A,(V2MODE)
                         ;
03F4'                    SETMODE:                       ; ENTRY WITH MODE IN A
                         ;
03F4'  A7                         AND   A
03F8'  28 4F                      JR    Z,SETM01        ; TEST FOR VALIDITY
03F7'  F0 00                      CP    B               ; 0-80 COL.MODE
03F9'  C2 0147'                   JP    NZ,INVPAR       ; INVALID PARAMETERS
                         ;                              ;
                         ;                              ;
03FC'  21 1701                    LD    HL,SCINIT       ; INITIALIZE VARIABLES
03FF'  22 002E'                   LD    (SCRCTL),HL     ; SCROLL CONTROL
0402'  3E 17                      LD    A,23            ;
0404'  32 0020'                   LD    (BOTLN),A       ; BOTTOM LINE
0407'  3E 01                      LD    A,1             ;
0409'  32 0028'                   LD    (SCRLCT),A      ; SCROLL COUNT
040C'  3E 50                      LD    A,80            ;
040E'  32 3055'                   LD    (LINLEN),A      ; LINE LENGTH
0411'  3E 01                      LD    A,1             ;
0413'  32 0050'                   LD    (MARGIN),A      ; MARGIN
0419'  21 1E00                    LD    HL,CLINIT       ;
0419'  22 000C'                   LD    (CLRCTL),HL     ; CLEAR SCREEN CONTROL
041C'  AF                         XOR   A
041D'  32 E059'                   LD    (MASKB),A       ; MASK BYTE
0420'  32 0050'                   LD    (STRGCT),A      ; RESTRG REMAINING COUNT
0423'  32 005C'                   LD    (STRGCT+1),A    ;
0426'  21 0475'                   LD    HL,CHRSET       ; STD. CHAR. SET
0429'  22 002F'                   LD    (CHTBL),HL      ;
042C'  21 0770'                   LD    HL,GPHST        ; STD. GRAPHICS SET
042F'  22 3051'                   LD    (GPTBL),HL      ;
                         ;                              ;
                         ;                              ; INITIALIZE BASIC INPUTS
                         ;                              ;
0432'  32 0000'                   LD    (DATAB),A       ; DATA BYTE
0435'  32 0007'                   LD    (LINCOL),A      ; COLUMN
0438'  32 0030'                   LD    (LINCOL+1),A    ; LINE
043B'  32 0019'                   LD    (MSKCTL),A      ; MASK CONTROL
043E'  32 0019'                   LD    (GETCTL),A      ; 'GET' COL.
0441'  32 0019'                   LD    (GETCTL+1),A    ; 'GET' LINE
0444'  3E 04                      LD    A,4             ; SET H/W TO 64-COL.MODE
0446'  47                SETM01   LD    B,A             ; SAVE MODE
0447'  3A 0CC2                    LD    A,(VIDMOD)
044A'  A7                         AND   A               ; TEST CURRENT MODE
044B'  20 10                      JR    NZ,SETM02
044D'  B8                         CP    B               ; CURRENT MODE=0  TEST IF NEW MODE=0
044E'  CA 0142'                   JP    Z,GOODRET       ; RETURN BC=0
0451'  21 12C0                    LD    HL,INIEAT11
0454'  11 0040                    LD    DE,MOVES1
0457'  19                         ADD   HL,DE           ; HL=TOTAL ROOM NEEDED
0458'  ED 5B 0C05                 LD    DE,(STKEND)
045C'  19                         ADD   HL,DE           ; ADD MEMORY IN USE
045D'  DA 04A7'                   JP    C,EXTNPM        ; NOT ENOUGH MEMORY TO OPEN OP2
0460'  ED 5B 1C02                 LD    DE,(RAMTOP)
0464'  A7                         AND   A
0465'  ED 52                      SBC   HL,DE
0467'  D2 04A7'                   JP    NC,EXTNAM       ; NOT ENOUGH MEMORY
046A'  CD 0402'          SETM02   CALL  ENBLEXT         ; ENABLE ROM EXT.
046D'  78                         LD    A,B             ; MODE TO A
046E'  CD 044C'                   CALL  GETVAL          ; H/W VALUE TO B; VIDMOD VALUE TO C
0471'  C5                         PUSH  BC              ; SAVE
0472'  78                         LD    A,B
0473'  CD 0E00                    CALL  CHNGVID         ; CALL RTN. TO OPEN/CLOSE 2ND DF
0476'  CD 040F'                   CALL  ENBLHOME        ; REENABLE HOME BANK
0479'  C1                         POP   BC              ; VALUE FOR VIDMOD
047A'  79                         LD    A,C
047B'  32 0CC2                    LD    (VIDMOD),A      ; UPDATE VIDMOD
047E'  CB 77             UPDATE   BIT   6,A             ; UPDATE SYSTEM VARS. BASED ON MODE
0480'  CA 0142'                   JP    Z,GOODRET       ; DONE IF MODE 0
0483'  3E C1             SXT4     LD    A,CCIN          ; 64/80-COL. MODE
0485'  B8                         CP    B               ; STORE ATTRIBUTE BYTE BASED
0489'  2F                         CPL                   ; ON INK SELECTION
```

```
0487'  C8 3E              902           SRL   8
0489'  C8 38              903           SRL   8
0488'  C8 38              904           SRL   8
048D'  80                 905           OR    R
048E'  32 0039'           906           LD    (ATTRBYT),A       ! ATTRIBUTE BYTE
0491'  32 0011'           907           LD    (ATTCTL),A        ! BASIC PARAMETER
0494'  3A 0033'           908           LD    A,(LINLEN)
0497'  3C                 909           INC   A
0498'  4F                 910           LD    C,A
0499'  06 1E              911           LD    B,I8H             ! BC=HOME POSITION CLN.0/COL.0)
049B'  21 8000            912           LD    HL,0
049E'  11 SCTD            913           LD    CCOORDS),HL       ! PLOT POSITION
04A1'  CD 0537'           914           CALL  UPDPOSN           ! UPDATE AND STORE HOME POSITION
04A4'  C3 0142'           915           JP    GOCORET           ! RETURN BC=0
04A7'  0E 02              916   ERTNRM  LD    C,2               ! RETURN BC=1 FOR NO BOOM
04A9'  C3 0144'           917           JP    ERRRET
                          918   !
                          919           SUBTTL INTERNAL SUB-RTNS.
                          920   !
                          921   !
                          922   !
                          923                                   ! INPUT:  VIDEO MODE IN R
                          924                                   ! OUTPUT: H/W SETTING IN A
                          925                                   !         VIDMOD SETTING IN C
                          926                                   ! RETURNS NZ STATUS IF MODE INVALID
                          927                                   !
                          928                                   !  VALID VALUES:  INPUT     H/W      VIDMOD
                          929                                   !                 -----     ---      ------
                          930                                   !   0=DISPLAY FILE ACTIVE AT SCREEN
                          931                                   !
                          932                                   !    OP1 ONLY       0        0        0
                          933                                   !   OP1=& OP1       80       0        80
                          934                                   !   OP1 & OP2=      1        1        81
                          935                                   !  HIGH RES.GR.     1        2        2
                          936                                   !  64/80-COLUMN   08 - 3E  06 - 1E   48 - 78
04AC'  4F                 937   GETVAL  LD    C,A
04AD'  CB 47              938           BIT   0,A
04AF'  20 16              939           JR    NZ,TSTGI
04B1'  FE 80              940           CP    80H
04B3'  28 1A              941           JR    Z,SETO
04B5'  CB 7F              942           BIT   7,A
04B7'  C0                 943           RET   NZ
04B8'  47                 944           LD    B,A
04B9'  FB 01              945           CP    1
04BB'  CB                 946           RET   Z
04BC'  B4 C6              947           AND   0C6H
04BE'  EE 06              948           XOR   6
04C0'  C0                 949           RET   NZ
04C1'  3E 40              950           LD    A,40H
04C3'  E1                 951           POP   H
04C4'  4F                 952           LD    C,A
04C5'  AF                 953           XOR   A
04C6'  C9                 954           RET
04C7'  FE 01              955   TSTOI   CP    1
04C9'  C1                 956           RET   NZ
04CA'  0E 01              957           LD    B,I
04CC'  0E A1              958           LD    C,0IH
04CE'  C9                 959           RET
04CF'  AF                 960   SETO    XOR   A
04D0'  47                 961           LD    B,A
04D1'  C9                 962           RET
                          963   !
                          964   !
                          965                                   ! ROUTINES TO ENABLE CHUNK 0 IN ROM EXT.
                          966                                   ! FOR ACCESS TO CHNGVID ROUTINE
04D2'  F5                 967   ENBLEXT PUSH  AF
04D3'  DB FF              968           IN    A,(MBEXPT)
04D5'  F6 E0              969           OR    80H         ! SELECT ROM EXT.
04D7'  D3 FF              970           OUT   (MREXPT),A
04D9'  3E 03              971           LD    A,3
04DB'  D3 F4              972           OUT   (DRMSPT),A
04DD'  F1                 973           POP   AF
04DE'  C9                 974           RET
                          975   !
04DF'  F1                 976   ENBLMCME  PUSH  AF
04E0'  AF                 977             XOR   A
04E1'  D3 F4              978             OUT   (DRMSPT),A
04E3'  DB FF              979             IN    A,(MBEXPT)
04E5'  E6 3F              980             AND   03FH
04E7'  D3 FF              981             OUT   (MREXPT),A
04E9'  F1                 982             POP   AF
04EA'  C9                 983             RET
                          984   !
                          985                                   ! TEST PARAMETER VALIDITY
                          986                                   ! IF NOT VALID, DISCARDS RETURN
                          987                                   ! AND EXITS VIA INVPAR
04EB'  3E 17              988   TSTPAR  LD    A,23              ! TEST LINE >23
04ED'  BB                 989           CP    B
04EE'  30 04              990           JR    NC,TSTPRI
04F0'  F1                 991   PARERR  POP   AF
04F1'  CI 0I6T'           992           JP    INVPAR
04F4'  3A 0033'           993   TSTPRI  LD    A,(LINLEN)
04F7'  3D                 994           DEC   A
04F8'  B9                 995           CP    C           ! TEST COL. >LINE LENGTH-1
04F9'  38 F1              996           JR    C,PARERR
04FB'  C9                 997           RET
                          998   !
                          999                                   ! CONVERT LINE/COL. FROM USER TO
                         1000                                   ! INTERNAL FORMAT AND VICE VERSA
                         1001                                   ! USER FORMAT: LINES 0-23
                         1002                                   !              COL. 0-79(84)
                         1003                                   ! INTERNAL FORMAT:
                         1004                                   !              LINES 24-1
                         1005                                   !              COL. C0638I-2
                         1006                                   !              (COL.1 AND UP LINE)
                         1007   !
                         1008   !
04FC'  3A 0033'          1009   CONVRM  LD    A,(LINLEN)
```

```
0AFF'  3C                          INC    A
0500'  91                          SUB    C
0501'  4F                          LD     C,A
0502'  3E 50                       LD     A,SCOSI           ; CONVERT
0504'  90                          SUB    B
0505'  47                          LD     B,A
0506'  C9                          RET
                           ;
                           ;
                           ;
0507'                      UPDPOSN:                         ; ENTER HERE WITH BC=LINE/COL.IN
                                                            ; INTERNAL FORMAT
0507'  CD 050C'                    CALL   CALCPOS           ; CALCULATE POSITION
050A'  18 44                       JR     STPOSN            ; STORE UPDATED POSITION AND RETURN
                           ;
                           ;                                ; ROUTINE TO CALCULATE POSITION IN DF
                           ;                                ; UPDATES BIT POSITION IN VARIABLE DFBIT
                           ;                                ; RETURNS DF ADDRESS IN HL. PRESERVES
                           ;                                ; LINE/COL.POSITION IN BC
                           ;
050C'  CD 053F'            CALCPOS CALL   LNOU              ; GET DISPLAY FILE ADRS.FOR LINE,
050F'  3A 0033'                    LD     A,(LINLEN)        ; TEST WHICH DF
0512'  3C                          INC    A
0513'  91                          SUB    C
0514'  F5                          PUSH   AF                ; SAVE COL.NO.(0-(LINLEN-1))
0515'  5F                          LD     E,A
0516'  87                          ADD    A,A
0517'  83                          ADD    A,E               ; 3*(80-COL.POS.) / 4 = 64-COL.POS.
0518'  CB 3F                       SRL    A
051A'  CB 3F                       SRL    A
051C'  CB 47                       BIT    0,A
051E'  28 06                       JR     Z,CALCF1
0520'  F5                          PUSH   AF                ; ODD COLS.ARE IN DF2
0521'  3E 20                       LD     A,20H
0523'  B4                          OR     H                 ; IN DF2
0524'  67                          LD     H,A
0525'  F1                          POP    AF
0526'  B7                  CALCF1  AND    A                 ; CLEAR CARRY
0527'  1F                          RRA                      ; COL/2 IS POSITION IN DISPLAY FILE
0528'  5F                          LD     E,A
0529'  3A 0035'                    LD     A,(MARGIN)        ; GET ADJUSTMENT VALUE
052C'  83                          ADD    A,E
052D'  5F                          LD     E,A
052E'  16 00                       LD     D,0               ; DE CONTAINS OFFSET INTO LINE
0530'  19                          ADD    HL,DE             ; HL CONTAINS DF ADDRESS
0531'  1E 07                       LD     E,7
0533'  F1                          POP    AF                ; A=COL.POSN.FOR 80 COLS.
0534'  E6 03                       AND    3                 ;
0536'  28 03                       JR     Z,CALCFS          ; STARTS AT BIT 7
0538'  5F                          LD     E,A               ; STARTS AT BIT 1,3 OR 5
0539'  3D                          DEC    A
053A'  83                  CALCFS  ADD    A,E
053B'  32 0032'                    LD     (DFBIT),A
053E'  C9                          RET                      ; CURPOS. HL=DF ADDRESS
                           ;
                           ;
053F'                      LNOU:                            ; GET DISPLAY FILE ADDRESS
                                                            ; FOR START OF LINE IN B
053F'  3E 18                       LD  A,BCAS1
0541'  90                          SUB B
0542'  57                          LD  D,A
0543'  0F                          RRCA
0544'  0F                          RRCA
0545'  0F                          RRCA
0546'  E6 E0                       AND 0E0H
0548'  6F                          LD  L,A
0549'  7A                          LD  A,D
054A'  E6 18                       AND 18H
054C'  F6 40                       OR  40H
054E'  67                          LD  H,A
054F'  C9                          RET
                           ;
0550'                      STPOSN:                          ; STORE CURSOR POSITION
0550'  ED 43 0034'                 LD  (CURPOS),BC
0554'  22 0036'                    LD  (DFADRS),HL
0557'  C9                          RET
                           ;
0558'                      LCPOSN:                          ; LOAD CURSOR POSITION
0558'  ED 4B 0034'                 LD  BC,(CURPOS)
055C'  2A 0036'                    LD  HL,(DFADRS)
055F'  C9                          RET
                           ;
                           ;
                           ;                                ; RTN. USED BY GTCHAR TO PUT
                           ;                                ; 6 PIXELS FROM DF IN B REG.BITS 7-2
                           ;                                ; FOR MATCHING AGAINST CHAR.SET
                           ;
0560'  E5                  GTC2SC  PUSH   HL                ; SAVE DF ADRS.
0561'  46                          LD     B,(HL)            ; SCAN LINE FROM DF
0562'  7C                          LD     A,H
0563'  EE 20                       XOR    20H
0565'  67                          LD     H,A               ; TOGGLE TO ALTERNATE DF
0566'  CB 6F                       BIT    5,A               ; TEST WHICH DF
0568'  20 01                       JR     NZ,GTC101
056A'  23                          INC    HL
056B'  4E                  GTC101  LD     C,(HL)            ; BC CONTAINS CHAR.AT DFBIT
056C'  3A 0032'                    LD     A,(DFBIT)
056F'  C6 07                       SUB    7
0571'  ED 44                       NEG                      ; A=NO.OF BITS OFFSET IN BC
0573'  28 07                       JR     Z,GTC102          ; STARTS AT BIT 7
0575'  CB 11               GTCSHFT RL     C
0577'  CB 10                       RL     B                 ; SHIFT 6 PIXELS FOR CHARACTER
0579'  3D                          DEC    A                 ; TO B, BITS 7-2
057A'  20 F9                       JR     NZ,GTCSHFT
057C'  E1                  GTC102  POP    HL                ;
057D'  C9                          RET                      ; CHAR.IN B, BITS 7-2
                           ;

                                   INCLUDE CHSTEO
                                   SUBTTL CHAR.SET FOR 40/80 COL.MODES
```

```
                        1121
                        1122  ;
                        1123  ; Dot patterns for characters on the TV screen!
                        1124  ;       character with code n is at offsets 8n (too line) to 8n+7 (bottom line)
                        1125  ;       0 = white, 1 = black, no margins between character spaces either
                        1126  ;               vertically or horizontally
                        1127
0570' 00                1128  CHRST   defb 00000000b          ; code 20       space
0571' 00                1129          defb 00000000b
0572' 00                1130          defb 00000000b
0573' 00                1131          defb 00000000b
0574' 00                1132          defb 00000000b
0575' 00                1133          defb 00000000b
0576' 00                1134          defb 00000000b
0577' 00                1135          defb 00000000b
                        1136
0578' 00                1137          defb 00010000b          ; code 21       !
0579' 10                1138          defb 00010000b
057A' 10                1139          defb 00010000b
057B' 10                1140          defb 00010000b
057C' 10                1141          defb 00010000b
057D' 00                1142          defb 00000000b
057E' 10                1143          defb 00010000b
057F' 00                1144          defb 00000000b
                        1145
0580' 00                1146          defb 00000000b          ; code 22       "
0581' 44                1147          defb 01001000b
0582' 48                1148          defb 01001000b
0583' 00                1149          defb 00000000b
0584' 00                1150          defb 00000000b
0585' 00                1151          defb 00000000b
0586' 00                1152          defb 00000000b
0587' 00                1153          defb 00000000b
                        1154
0588' 00                1155          defb 00000000b          ; code 23       #
0589' 28                1156          defb 00101000b
058A' 7C                1157          defb 01111100b
058B' 28                1158          defb 00101000b
058C' 28                1159          defb 00101000b
058D' 7C                1160          defb 01111100b
058E' 28                1161          defb 00101000b
058F' 00                1162          defb 00000000b
                        1163
0590' 00                1164          defb 00000000b          ; code 24       $
0591' 10                1165          defb 00010000b
0592' 7C                1166          defb 01111100b
0593' 90                1167          defb 01010000b
0594' 7C                1168          defb 01111100b
0595' 14                1169          defb 00010100b
0596' 7C                1170          defb 01111100b
0597' 10                1171          defb 00010000b
                        1172
0598' 00                1173          defb 00000000b          ; code 25       %
0599' 44                1174          defb 01000100b
059A' 46                1175          defb 01000100b
059B' 10                1176          defb 00010000b
059C' 10                1177          defb 00010000b
059D' 20                1178          defb 00100000b
059E' 44                1179          defb 01000100b
059F' 00                1180          defb 00000000b
                        1181
05A0' 00                1182          defb 00000000b          ; code 26       &
05A1' 10                1183          defb 00010000b
05A2' 28                1184          defb 00101000b
05A3' 10                1185          defb 00010000b
05A4' 2A                1186          defb 00101100b
05A5' 44                1187          defb 01001000b
05A6' 36                1188          defb 00110100b
05A7' 00                1189          defb 00000000b
                        1190
05A8' 00                1191          defb 00000000b          ; code 27       '
05A9' 10                1192          defb 00010000b
05AA' 20                1193          defb 00100000b
05AB' 00                1194          defb 00000000b
05AC' 00                1195          defb 00000000b
05AD' 00                1196          defb 00000000b
05AE' 00                1197          defb 00000000b
05AF' 00                1198          defb 00000000b
                        1199
05B0' 00                1200          defb 00000000b          ; code 28       (
05B1' 08                1201          defb 00001000b
05B2' 10                1202          defb 00010000b
05B3' 10                1203          defb 00010000b
05B4' 10                1204          defb 00010000b
05B5' 10                1205          defb 00010000b
05B6' 08                1206          defb 00001000b
05B7' 00                1207          defb 00000000b
                        1208
05B8' 00                1209          defb 00000000b          ; code 29       )
05B9' 20                1210          defb 00100000b
05BA' 10                1211          defb 00010000b
05BB' 10                1212          defb 00010000b
05BC' 10                1213          defb 00010000b
05BD' 10                1214          defb 00010000b
05BE' 20                1215          defb 00100000b
05BF' 00                1216          defb 00000000b
                        1217
05C0' 00                1218          defb 00000000b          ; code 2A       *
05C1' 00                1219          defb 00000000b
05C2' 28                1220          defb 00101000b
05C3' 10                1221          defb 00010000b
05C4' 7C                1222          defb 01111100b
05C5' 10                1223          defb 00010000b
05C6' 28                1224          defb 00101000b
05C7' 00                1225          defb 00000000b
                        1226
05C8' 00                1227          defb 00000000b          ; code 2B       +
05C9' 00                1228          defb 00000000b
05CA' 10                1229          defb 00010000b
05CB' 10                1230          defb 00010000b
05CC' 7C                1231          defb 01111100b
```

```
05DB'  10        defb 00010000b
05DC'  10        defb 00010000b
05DD'  00        defb 00000000b

05DE'  00        defb 00000000b      ; code 2C      ,
05DF'  00        defb 00000000b
05E0'  00        defb 00000000b
05E1'  00        defb 00000000b
05E2'  00        defb 00000000b
05E3'  10        defb 00010000b
05E4'  10        defb 00010000b
05E5'  20        defb 00100000b

05E6'  00        defb 00000000b      ; code 2D      -
05E7'  00        defb 00000000b
05E8'  00        defb 00000000b
05E9'  00        defb 00000000b
05EA'  7C        defb 01111100b
05EB'  00        defb 00000000b
05EC'  00        defb 00000000b
05ED'  00        defb 00000000b

05EE'  00        defb 00000000b      ; code 2E      .
05EF'  00        defb 00000000b
05F0'  00        defb 00000000b
05F1'  00        defb 00000000b
05F2'  00        defb 00000000b
05F3'  30        defb 00110000b
05F4'  30        defb 00110000b
05F5'  00        defb 00000000b

05F6'  00        defb 00000000b      ; code 2F      /
05F7'  00        defb 00000000b
05F8'  04        defb 00000100b
05F9'  08        defb 00001000b
05FA'  10        defb 00010000b
05FB'  20        defb 00100000b
05FC'  40        defb 01000000b
05FD'  00        defb 00000000b

05FE'  00        defb 00000000b      ; code 30      0
05FF'  38        defb 00111000b
0600'  4C        defb 01001100b
0601'  54        defb 01010100b
0602'  54        defb 01010100b
0603'  64        defb 01100100b
0604'  38        defb 00111000b
0605'  00        defb 00000000b

0606'  10        defb 00010000b      ; code 31      1
0607'  30        defb 00110000b
0608'  50        defb 01010000b
0609'  10        defb 00010000b
060A'  10        defb 00010000b
060B'  10        defb 00010000b
060C'  7C        defb 01111100b
060D'  00        defb 00000000b

060E'  00        defb 00000000b      ; code 32      2
060F'  38        defb 00111000b
0610'  44        defb 01000100b
0611'  04        defb 00000100b
0612'  38        defb 00111000b
0613'  40        defb 01000000b
0614'  7C        defb 01111100b
0615'  00        defb 00000000b

0616'  00        defb 00000000b      ; code 33      3
0617'  38        defb 00111000b
0618'  44        defb 01000100b
0619'  18        defb 00011000b
061A'  04        defb 00000100b
061B'  44        defb 01000100b
061C'  38        defb 00111000b
061D'  00        defb 00000000b

061E'  00        defb 00000000b      ; code 34      4
061F'  08        defb 00001000b
0620'  18        defb 00011000b
0621'  28        defb 00101000b
0622'  48        defb 01001000b
0623'  7C        defb 01111100b
0624'  08        defb 00001000b
0625'  00        defb 00000000b

0626'  00        defb 00000000b      ; code 35      5
0627'  7C        defb 01111100b
0628'  40        defb 01000000b
0629'  78        defb 01111000b
062A'  04        defb 00000100b
062B'  44        defb 01000100b
062C'  38        defb 00111000b
062D'  00        defb 00000000b

062E'  00        defb 00000000b      ; code 36      6
062F'  38        defb 00111000b
0630'  40        defb 01000000b
0631'  78        defb 01111000b
0632'  44        defb 01000100b
0633'  44        defb 01000100b
0634'  38        defb 00111000b
0635'  00        defb 00000000b

0636'  00        defb 00000000b      ; code 37      7
```

```
0b37' 7C          1336          defb 01111100b
0b38' 04          1337          defb 00000100b
0b39' 0b          1338          defb 000010000
0b3A' 10          1339          defb 000100000
0b3B' 20          1340          defb 001000000
0b3C' 30          1341          defb 001100000b
0b3D' 00          1342          defb 000000000b
                  1343
0b3E' 00          1344          defb 000000000b       1 code 3E      0
0b3F' 38          1345          defb 001110000b
0b40' 44          1346          defb 010001000
0b41' 33          1347          defb 001110000
0b42' 44          1348          defb 010001000
0b43' 44          1349          defb 010001000
0b44' 38          1350          defb 001110000b
0b45' 00          1351          defb 00000000b
                  1352
0b46' 00          1353          defb 000b0000b        1 code 39      9
0b47' 38          1354          defb 001110000b
0b48' 44          1355          defb 010001000
0b49' 44          1356          defb 010001000
0b4A' 3C          1357          defb 001111000
0b4B' 04          1358          defb 000001000
0b4C' 38          1359          defb 001110000
0b4D' 00          1360          defb 000000000b
                  1361
0b4E' 00          1362          defb 000000000b       1 code 3b      1
0b4F' 00          1363          defb 000000000
0b50' 00          1364          defb 000000000b
0b51' 10          1365          defb 000100000
0b52' 00          1366          defb 000000000b
0b53' 00          1367          defb 000000000b
0b54' 10          1368          defb 000100000
0b55' 00          1369          defb 000000000b
                  1370
0b56' 00          1371          defb 000000000b       1 code 3D      1
0b57' 00          1372          defb 000000000b
0b58' 10          1373          defb 000100000
0b59' 03          1374          defb 000000000
0b5A' 00          1375          defb 000000000
0b5B' 10          1376          defb 000100000b
0b5C' 13          1377          defb 000100000
0b5D' 30          1378          defb 001100000
                  1379
0b5E' 00          1380          defb 000000000        1 code 3C      <
0b5F' 00          1381          defb 000000000b
0b60' 00          1382          defb 000000000b
0A61' 10          1383          defb 000100000b
0b62' 20          1384          defb 001000000b
0b63' 10          1385          defb 000100000b
0b64' 00          1386          defb 000000000b
0b65' 00          1387          defb 000000000b
                  1388
0b66' 00          1389          defb 000000000b       1 code 3D      =
0b67' 00          1390          defb 000000000b
0b68' 00          1391          defb 000000000b
0b69' 7C          1392          defb 01111100b
0b6A' 40          1393          defb 000000000
0b6B' 7C          1394          defb 01111100b
0b6C' 00          1395          defb 000000000b
0b6D' 00          1396          defb 000000000b
                  1397
0b6E' 00          1398          defb 000000000b       1 code 3E      >
0b6F' 40          1399          defb 000000000b
0b70' 20          1400          defb 001000000b
0b71' 10          1401          defb 000100000b
0b72' 08          1402          defb 000010000b
0b73' 10          1403          defb 000100000b
0b74' 20          1404          defb 001000000b
0b75' 00          1405          defb 000000000b
                  1406
0b76' 00          1407          defb 000000000        1 code 3F      ?
0b77' 38          1408          defb 001110000b
0b78' 44          1409          defb 010001000
0b79' 08          1410          defb 000100000b
0b7A' 10          1411          defb 000100000
0b7B' 00          1412          defb 000000000
0b7C' 10          1413          defb 000100000
0b7D' 00          1414          defb 000000000b
                  1415
0b7E' 00          1416          defb 000000000        1 code 40      0
0b7F' 38          3b17          defb 001110000b
0b80' 44          1418          defb 010001000
0b81' 5C          1419          defb 010111000
0b82' 5C          1420          defb 010111000b
0b83' 40          1421          defb 010000000
0b84' 3C          1422          defb 001111000
0b85' 00          1423          defb 000000000b
                  1424
0b86' 04          1425          defb 000000000b       1 code 41      A
0b87' 38          1426          defb 001110000b
0b88' 44          1427          defb 010001000
0b89' 44          1428          defb 010001000
0b8A' 7C          1429          defb 011111000
0b8B' 44          1430          defb 010001000
0b8C' 44          1431          defb 010001000
0b8D' 40          1432          defb 000000000b
                  1433
0b8E' 00          1434          defb 000000000b       1 code 42      B
0b8F' 78          1435          defb 011110000
0b90' 44          1436          defb 010001000
0b91' 78          1437          defb 011110000
0b92' 44          1438          defb 010001000
0b93' 44          1439          defb 010001000
```

```
0694' 78                1440        defb 01111000b
0698' 00                1441        defb 00000000b
                        1442
0696' 00                1443        defb 00000000b          ; code 43       C
0697' 38                1444        defb 00111000b
0698' 44                1445        defb 01000100b
0699' 40                1446        defb 01000000b
069A' 40                1447        defb 01000000b
069B' 44                1448        defb 01000100b
069C' 38                1449        defb 00111000b
069D' 00                1450        defb 00000000b
                        1451
069E' 00                1452        defb 00000000b          ; code 44       D
069F' 78                1453        defb 01111000b
06A0' 48                1454        defb 01001000b
06A1' 44                1455        defb 01000100b
06A2' 44                1456        defb 01000100b
06A3' 48                1457        defb 01001000b
06A4' 78                1458        defb 01111000b
06A5' 00                1459        defb 00000000b
                        1460
06A6' 00                1461        defb 00000000b          ; code 45       E
06A7' 7C                1462        defb 01111100b
06A8' 40                1463        defb 01000000b
06A9' 78                1464        defb 01111000b
06AA' 40                1465        defb 01000000b
06AB' 40                1466        defb 01000000b
06AC' 7C                1467        defb 01111100b
06AD' 00                1468        defb 00000000b
                        1469
06AE' 00                1470        defb 00000000b          ; code 46       F
06AF' 7C                1471        defb 01111100b
06B0' 40                1472        defb 01000000b
06B1' 78                1473        defb 01111000b
06B2' 40                1474        defb 01000000b
06B3' 40                1475        defb 01000000b
06B4' 40                1476        defb 01000000b
06B5' 00                1477        defb 00000000b
                        1478
06B6' 00                1479        defb 00000000b          ; code 47       G
06B7' 38                1480        defb 00111000b
06B8' 44                1481        defb 01000100b
06B9' 40                1482        defb 01000000b
06BA' 4C                1483        defb 01001100b
06BB' 44                1484        defb 01000100b
06BC' 38                1485        defb 00111000b
06BD' 00                1486        defb 00000000b
                        1487
06BE' 00                1488        defb 00000000b          ; code 48       H
06BF' 44                1489        defb 01000100b
06C0' 44                1490        defb 01000100b
06C1' 7C                1491        defb 01111100b
06C2' 44                1492        defb 01000100b
06C3' 44                1493        defb 01000100b
06C4' 44                1494        defb 01000100b
06C5' 00                1495        defb 00000000b
                        1496
06C6' 00                1497        defb 00000000b          ; code 49       I
06C7' 7C                1498        defb 01111100b
06C8' 10                1499        defb 00010000b
06C9' 10                1500        defb 00010000b
06CA' 10                1501        defb 00010000b
06CB' 10                1502        defb 00010000b
06CC' 7C                1503        defb 01111100b
06CD' 00                1504        defb 00000000b
                        1505
06CE' 00                1506        defb 00000000b          ; code 4A       J
06CF' 04                1507        defb 00000100b
06D0' 04                1508        defb 00000100b
06D1' 04                1509        defb 00000100b
06D2' 44                1510        defb 01000100b
06D3' 44                1511        defb 01000100b
06D4' 3C                1512        defb 00111100b
06D5' 00                1513        defb 00000000b
                        1514
06D6' 00                1515        defb 00000000b          ; code 4B       K
06D7' 48                1516        defb 01001000b
06D8' 50                1517        defb 01010000b
06D9' 60                1518        defb 01100000b
06DA' 60                1519        defb 01100000b
06DB' 50                1520        defb 01010000b
06DC' 48                1521        defb 01001000b
06DD' 44                1522        defb 01000100b
06DE' 00                1523        defb 00000000b
                        1524
06DF' 00                1525        defb 00000000b          ; code 4C       L
06E0' 40                1526        defb 01000000b
06E1' 40                1527        defb 01000000b
06E2' 40                1528        defb 01000000b
06E3' 40                1529        defb 01000000b
06E4' 7C                1530        defb 01111100b
06E5' 00                1531        defb 00000000b
                        1532
06E6' 00                1533        defb 00000000b          ; code 4D       M
06E7' 44                1534        defb 01000100b
06E8' 6C                1535        defb 01101100b
06E9' 54                1536        defb 01010100b
06EA' 44                1537        defb 01000100b
06EB' 44                1538        defb 01000100b
06EC' 44                1539        defb 01000100b
06ED' 00                1540        defb 00000000b
                        1541
06EE' 00                1542        defb 00000000b          ; code 4E       N
06EF' 44                1543        defb 01000100b
```

```
06C2'  44        1492        defb 01000100b
06C3'  44        1493        defb 01000100b
06C4'  44        1494        defb 01000100b
06C5'  00        1495        defb 00000000b
                 1496
06C6'  00        1497        defb 00000000b          ; code 49      I
06C7'  7C        1498        defb 01111100b
06C8'  10        1499        defb 00010000b
06C9'  10        1500        defb 00010000b
06CA'  10        1501        defb 00010000b
06CB'  10        1502        defb 00010000b
06CC'  7C        1503        defb 01111100b
06CD'  00        1504        defb 00000000b
                 1505
06CE'  00        1506        defb 00000000b          ; code 4A      J
06CF'  04        1507        defb 00000100b
06D0'  04        1508        defb 00000100b
06D1'  04        1509        defb 00000100b
06D2'  44        1510        defb 01000100b
06D3'  44        1511        defb 01000100b
06D4'  38        1512        defb 00111000b
06D5'  00        1513        defb 00000000b
                 1514
06D6'  00        1515        defb 00000000b          ; code 4B      K
06D7'  48        1516        defb 01001000b
06D8'  50        1517        defb 01010000b
06D9'  60        1518        defb 01100000b
06DA'  50        1519        defb 01010000b
06DB'  48        1520        defb 01001000b
06DC'  44        1521        defb 01000100b
06DD'  00        1522        defb 00000000b
                 1523
06DE'  00        1524        defb 00000000b          ; code 4C      L
06DF'  40        1525        defb 01000000b
06E0'  40        1526        defb 01000000b
06E1'  40        1527        defb 01000000b
06E2'  40        1528        defb 01000000b
06E3'  40        1529        defb 01000000b
06E4'  7C        1530        defb 01111100b
06E5'  00        1531        defb 00000000b
                 1532
06E6'  00        1533        defb 00000000b          ; code 4D      M
06E7'  44        1534        defb 01000100b
06E8'  6C        1535        defb 01101100b
06E9'  54        1536        defb 01010100b
06EA'  44        1537        defb 01000100b
06EB'  44        1538        defb 01000100b
06EC'  44        1539        defb 01000100b
06ED'  00        1540        defb 00000000b
                 1541
06EE'  00        1542        defb 00000000b          ; code 4E      N
06EF'  44        1543        defb 01000100b
06F0'  64        1544        defb 01100100b
06F1'  54        1545        defb 01010100b
06F2'  54        1546        defb 01010100b
06F3'  4C        1547        defb 01001100b
06F4'  44        1548        defb 01000100b
06F5'  00        1549        defb 00000000b
                 1550
06F6'  00        1551        defb 00000000b          ; code 4F      O
06F7'  38        1552        defb 00111000b
06F8'  44        1553        defb 01000100b
06F9'  44        1554        defb 01000100b
06FA'  44        1555        defb 01000100b
06FB'  44        1556        defb 01000100b
06FC'  38        1557        defb 00111000b
06FD'  00        1558        defb 00000000b
                 1559
06FE'  00        1560        defb 00000000b          ; code 50      P
06FF'  78        1561        defb 01111000b
0700'  44        1562        defb 01000100b
0701'  44        1563        defb 01000100b
0702'  78        1564        defb 01111000b
0703'  40        1565        defb 01000000b
0704'  40        1566        defb 01000000b
0705'  00        1567        defb 00000000b
                 1568
0706'  00        1569        defb 00000000b          ; code 51      Q
0707'  38        1570        defb 00111000b
0708'  44        1571        defb 01000100b
0709'  44        1572        defb 01000100b
070A'  44        1573        defb 01000100b
070B'  54        1574        defb 01010100b
070C'  38        1575        defb 00111000b
070D'  00        1576        defb 00000000b
                 1577
070E'  00        1578        defb 00000000b          ; code 52      R
070F'  78        1579        defb 01111000b
0710'  44        1580        defb 01000100b
0711'  44        1581        defb 01000100b
0712'  78        1582        defb 01111000b
0713'  48        1583        defb 01001000b
0714'  44        1584        defb 01000100b
0715'  00        1585        defb 00000000b
                 1586
0716'  00        1587        defb 00000000b          ; code 53      S
0717'  38        1588        defb 00111000b
0718'  40        1589        defb 01000000b
0719'  38        1590        defb 00111000b
071A'  04        1591        defb 00000100b
071B'  44        1592        defb 01000100b
071C'  38        1593        defb 00111000b
071D'  00        1594        defb 00000000b
                 1595
```

```
071E'  00        1596      defb 00000000b      ; code 54    T
071F'  7C        1597      defb 01111100b
0720'  10        1598      defb 00010000b
0721'  10        1599      defb 00010000b
0722'  10        1600      defb 00010000b
0723'  10        1601      defb 00010000b
0724'  10        1602      defb 00010000b
0725'  00        1603      defb 00000000b
                 1604
0726'  00        1605      defb 00000000b      ; code 55    U
0727'  44        1606      defb 01000100b
0728'  44        1607      defb 01000100b
0729'  44        1608      defb 01000100b
072A'  44        1609      defb 01000100b
072B'  44        1610      defb 01000100b
072C'  38        1611      defb 00111000b
072D'  00        1612      defb 00000000b
                 1613
072E'  00        1614      defb 00000000b      ; code 56    V
072F'  44        1615      defb 01000100b
0730'  44        1616      defb 01000100b
0731'  44        1617      defb 01000100b
0732'  44        1618      defb 01000100b
0733'  28        1619      defb 00101000b
0734'  10        1620      defb 00010000b
0735'  00        1621      defb 00000000b
                 1622
0736'  00        1623      defb 00000000b      ; code 57    W
0737'  44        1624      defb 01000100b
0738'  44        1625      defb 01000100b
0739'  44        1626      defb 01000100b
073A'  54        1627      defb 01010100b
073B'  54        1628      defb 01010100b
073C'  28        1629      defb 00101000b
073D'  00        1630      defb 00000000b
                 1631
073E'  00        1632      defb 00000000b      ; code 58    X
073F'  44        1633      defb 01000100b
0740'  28        1634      defb 00101000b
0741'  10        1635      defb 00010000b
0742'  10        1636      defb 00010000b
0743'  28        1637      defb 00101000b
0744'  44        1638      defb 01000100b
0745'  00        1639      defb 00000000b
                 1640
0746'  00        1641      defb 00000000b      ; code 59    Y
0747'  44        1642      defb 01000100b
0748'  44        1643      defb 01000100b
0749'  28        1644      defb 00101000b
074A'  10        1645      defb 00010000b
074B'  10        1646      defb 00010000b
074C'  10        1647      defb 00010000b
074D'  00        1648      defb 00000000b
                 1649
074E'  00        1650      defb 00000000b      ; code 5A    Z
074F'  7C        1651      defb 01111100b
0750'  04        1652      defb 00000100b
0751'  08        1653      defb 00001000b
0752'  10        1654      defb 00010000b
0753'  20        1655      defb 00100000b
0754'  7C        1656      defb 01111100b
0755'  00        1657      defb 00000000b
                 1658
0756'  00        1659      defb 00000000b      ; code 5B    [
0757'  38        1660      defb 00111000b
0758'  20        1661      defb 00100000b
0759'  20        1662      defb 00100000b
075A'  20        1663      defb 00100000b
075B'  20        1664      defb 00100000b
075C'  38        1665      defb 00111000b
075D'  00        1666      defb 00000000b
                 1667
075E'  00        1668      defb 00000000b      ; code 5C    \
075F'  00        1669      defb 00000000b
0760'  40        1670      defb 01000000b
0761'  20        1671      defb 00100000b
0762'  10        1672      defb 00010000b
0763'  08        1673      defb 00001000b
0764'  04        1674      defb 00000100b
0765'  00        1675      defb 00000000b
                 1676
0766'  00        1677      defb 00000000b      ; code 5D    ]
0767'  70        1678      defb 01110000b
0768'  10        1679      defb 00010000b
0769'  10        1680      defb 00010000b
076A'  10        1681      defb 00010000b
076B'  10        1682      defb 00010000b
076C'  70        1683      defb 01110000b
076D'  00        1684      defb 00000000b
                 1685
076E'  00        1686      defb 00000000b      ; code 5E    ^
076F'  10        1687      defb 00010000b
0770'  38        1688      defb 00111000b
0771'  54        1689      defb 01010100b
0772'  10        1690      defb 00010000b
0773'  10        1691      defb 00010000b
0774'  10        1692      defb 00010000b
0775'  00        1693      defb 00000000b
                 1694
0776'  00        1695      defb 00000000b      ; code 5F    _
0777'  00        1696      defb 00000000b
0778'  00        1697      defb 00000000b
0779'  00        1698      defb 00000000b
077A'  00        1699      defb 00000000b
```

```
0778' 00              1700          defb 00000000b
077C' 00              1701          defb 00000000b
077D' FC              1702          defb 11111100b
                      1702
0778' 00              1704          defb 00000000b          ; code 60       pound sign
077F' 18              1703          defb 00011000b
0780' 24              1706          defb 00100100b
0781' 70              1707          defb 01110000b
0782' 20              1708          defb 00100000b
0783' 20              1709          defb 00100000b
0784' 7C              1710          defb 01111100b
0783' 00              1711          defb 00000000b
                      1712
0788' 00              1713          defb 00000000b          ; code 61       a
0787' 00              1714          defb 00000000b
0788' 38              1713          defb 00111000b
0789' 04              1716          defb 00000100b
078A' 3C              1717          defb 00111100b
0788' 04              1718          defb 01000100b
078C' 3C              1719          defb 00111100b
0780' 00              1720          defb 00000000b
                      1721
0788' 00              1722          defb 00000000b          ; code 62       b
078F' 20              1723          defb 00100000b
0790' 20              1724          defb 00100000b
0731' 28              1723          defb 00111000b
0792' 24              1726          defb 00100100b
0793' 24              1727          defb 00100100b
0794' 38              1728          defb 00111000b
0793' 00              1729          defb 00000000b
                      1730
0794' 00              1731          defb 00000000b          ; code 62       b
0797' 00              1722          defb 00000000b
0798' 1C              1732          defb 00011100b
0739' 20              1734          defb 00100000b
073A' 20              1723          defb 00100000b
0790' 20              1730          defb 00100000b
073C' 1C              1737          defb 00011100b
0790' 00              1729          defb 00000000b
                      1739
0798' 00              1740          defb 00000000b          ; code 64       d
079F' 08              1741          defb 00001000b
07A0' 0b              1742          defb 00001000b
07A1' 38              1743          defb 00111000b
07A2' 48              1744          defb 01001000b
07A3' 48              1745          defb 01001000b
07A4' 38              1740          defb 00111000b
07A3' 00              1747          defb 00000000b
                      1748
07A4' 00              1749          defb 00000000b          ; code 63       e
07A7' 00              1790          defb 00000000b
07A0' 38              1731          defb 00111000b
                      1732          defb 01000100b
07A9' 44              1733          defb 01111000b
07A6' 78              1734          defb 01000000b
07A8' 40              1735          defb 00111100b
07AC' 3C              1736          defb 00000000b
07A0' 00              1737
                      1738          defb 00000000b          ; code 66       f
07A0' 00              1739          defb 00010000b
07AF' 10              1760          defb 00101000b
8780' 28              1761          defb 01110000b
8781' 70              1762          defb 00100000b
8782' 20              1762          defb 00100000b
8782' 20              1764          defb 00100000b
8784' 20              1763          defb 00000000b
8783' 00              1766
                      1767          defb 00000000b          ; code 67       g
8786' 00              1768          defb 00000000b
8787' 00              1769          defb 00111000b
8788' 38              1770          defb 01001000b
8789' 48              1771          defb 01001000b
878A' 48              1772          defb 00111000b
878B' 28              1773          defb 00001000b
878C' 08              1774          defb 00110000b
878D' 30              1773
                      1778          defb 00000000b          ; code 68       h
878E' 00              1777          defb 01000000b
878F' 40              1778          defb 01000000b
87C0' 40              1773          defb 01110000b
87C1' 70              1780          defb 01001000b
87C2' 48              1781          defb 01001000b
87C3' 48              1782          defb 01001000b
87C4' 48              1782          defb 00000000b
87C3' 00              1784
                      1783          defb 00020000b          ; code 69       i
87C7' 10              1786          defb 00010000b
87C0' 00              1787          defb 00000000b
87C3' 20              1788          defb 00110000b
87CA' 10              1783          defb 00010000b
87CB' 10              1790          defb 00010000b
87CC' 38              1791          defb 00111000b
87CD' 00              1732          defb 00000000b
                      1733
87CE' 00              1734          defb 00000000b          ; code 6A       j
87CF' 08              1735          defb 00001000b
8700' 00              1798          defb 00000000b
8701' 08              1797          defb 00001000b
8702' 08              1798          defb 00001000b
8702' 08              1793          defb 00001000b
8704' 48              1800          defb 01001000b
8703' 30              1801          defb 00110000b
                      1802
8706' 00              1802          defb 00000000b          ; code 6B       k
```

```
07D7'  20        1004        defb 00100000b
07D8'  28        1005        defb 00101000b
07D9'  30        1006        defb 00110000b
07DA'  30        1007        defb 00110000b
07DB'  28        1008        defb 00101000b
07DC'  24        1009        defb 00100100b
07DD'  00        1010        defb 00000000b
                 1011
07DE'  00        1012        defb 00000000b       ; code 6C     l
07DF'  20        1013        defb 00100000b
07E0'  20        1014        defb 00100000b
07E1'  20        1015        defb 00100000b
07E2'  20        1016        defb 00100000b
07E3'  20        1017        defb 00100000b
07E4'  18        1018        defb 00011000b
07E5'  00        1019        defb 00000000b
                 1020
07E6'  00        1021        defb 00000000b       ; code 6D     m
07E7'  00        1022        defb 00000000b
07E8'  28        1023        defb 00101000b
07E9'  54        1024        defb 01010100b
07EA'  54        1025        defb 01010100b
07EB'  54        1026        defb 01010100b
07EC'  54        1027        defb 01010100b
07ED'  00        1028        defb 00000000b
                 1029
07EE'  00        1030        defb 00000000b       ; code 6E     n
07EF'  00        1031        defb 00000000b
07F0'  70        1032        defb 01110000b
07F1'  48        1033        defb 01001000b
07F2'  48        1034        defb 01001000b
07F3'  48        1035        defb 01001000b
07F4'  48        1036        defb 01001000b
07F5'  00        1037        defb 00000000b
                 1038
07F6'  00        1039        defb 00000000b       ; code 6F     o
07F7'  00        1040        defb 00000000b
07F8'  38        1041        defb 00111000b
07F9'  44        1042        defb 01000100b
07FA'  44        1043        defb 01000100b
07FB'  44        1044        defb 01000100b
07FC'  38        1045        defb 00111000b
07FD'  00        1046        defb 00000000b
                 1047
07FE'  00        1048        defb 00000000b       ; code 70     p
07FF'  00        1049        defb 00000000b
0800'  70        1050        defb 01110000b
0801'  48        1051        defb 01001000b
0802'  48        1052        defb 01001000b
0803'  70        1053        defb 01110000b
0804'  40        1054        defb 01000000b
0805'  40        1055        defb 01000000b
                 1056
0806'  00        1057        defb 00000000b       ; code 71     q
0807'  00        1058        defb 00000000b
0808'  38        1059        defb 00111000b
0809'  48        1060        defb 01001000b
080A'  48        1061        defb 01001000b
080B'  38        1062        defb 00111000b
080C'  08        1063        defb 00001000b
080D'  0C        1064        defb 00001100b
                 1065
080E'  00        1066        defb 00000000b       ; code 72     r
080F'  00        1067        defb 00000000b
0810'  18        1068        defb 00011000b
0811'  20        1069        defb 00100000b
0812'  20        1070        defb 00100000b
0813'  20        1071        defb 00100000b
0814'  20        1072        defb 00100000b
0815'  00        1073        defb 00000000b
                 1074
0816'  00        1075        defb 00000000b       ; code 73     s
0817'  00        1076        defb 00000000b
0818'  30        1077        defb 00110000b
0819'  40        1078        defb 01000000b
081A'  30        1079        defb 00110000b
081B'  08        1080        defb 00001000b
081C'  70        1081        defb 01110000b
081D'  00        1082        defb 00000000b
                 1083
081E'  00        1084        defb 00000000b       ; code 74     t
081F'  20        1085        defb 00100000b
0820'  70        1086        defb 01110000b
0821'  20        1087        defb 00100000b
0822'  20        1088        defb 00100000b
0823'  20        1089        defb 00100000b
0824'  18        1090        defb 00011000b
0825'  00        1091        defb 00000000b
                 1092
0826'  00        1093        defb 00000000b       ; code 75     u
0827'  00        1094        defb 00000000b
0828'  48        1095        defb 01001000b
0829'  48        1096        defb 01001000b
082A'  48        1097        defb 01001000b
082B'  48        1098        defb 01001000b
082C'  30        1099        defb 00110000b
082D'  00        1100        defb 00000000b
                 1101
082E'  00        1102        defb 00000000b       ; code 76     v
082F'  00        1103        defb 00000000b
0830'  44        1104        defb 01000100b
0831'  44        1105        defb 01000100b
0832'  28        1106        defb 00101000b
0833'  28        1107        defb 00101000b
```

```
0834' 10                1908            defb 00010000b
0835' 00                1909            defb 00000000b
                        1910
0836' 00                1911            defb 00000000b          ; code 77       w
0837' 00                1912            defb 00000000b
0838' 44                1913            defb 01000100b
0839' 54                1914            defb 01010100b
083A' 54                1915            defb 01010100b
083B' 54                1916            defb 01010100b
083C' 28                1917            defb 00101000b
083D' 00                1918            defb 00000000b
                        1919
083E' 00                1920            defb 00000000b          ; code 78       x
083F' 00                1921            defb 00000000b
0840' 44                1922            defb 01000100b
0841' 28                1923            defb 00101000b
0842' 10                1924            defb 00010000b
0843' 28                1925            defb 00101000b
0844' 44                1926            defb 01000100b
0845' 00                1927            defb 00000000b
                        1928
0846' 00                1929            defb 00000000b          ; code 79       y
0847' 00                1930            defb 00000000b
0848' 48                1931            defb 01001000b
0849' 48                1932            defb 01001000b
084A' 48                1933            defb 01001000b
084B' 38                1934            defb 00111000b
084C' 08                1935            defb 00001000b
084D' 70                1936            defb 01110000b
                        1937
084E' 00                1938            defb 00000000b          ; code 7A       z
084F' 00                1939            defb 00000000b
0850' 7C                1940            defb 01111100b
0851' 08                1941            defb 00001000b
0852' 10                1942            defb 00010000b
0853' 20                1943            defb 00100000b
0854' 7C                1944            defb 01111100b
0855' 00                1945            defb 00000000b
                        1946
0856' 00                1947            defb 00000000b          ; code 7B       {
0857' 1C                1948            defb 00011100b
0858' 10                1949            defb 00010000b
0859' 60                1950            defb 01100000b
085A' 10                1951            defb 00010000b
085B' 10                1952            defb 00010000b
085C' 1C                1953            defb 00011100b
085D' 00                1954            defb 00000000b
                        1955
085E' 00                1956            defb 00000000b          ; code 7C       |
085F' 10                1957            defb 00010000b
0860' 10                1958            defb 00010000b
0861' 10                1959            defb 00010000b
0862' 10                1960            defb 00010000b
0863' 10                1961            defb 00010000b
0864' 10                1962            defb 00010000b
0865' 00                1963            defb 00000000b
                        1964
0866' 00                1965            defb 00000000b          ; code 7D       }
0867' 70                1966            defb 01110000b
0868' 10                1967            defb 00010000b
0869' 08                1968            defb 00001000b
086A' 10                1969            defb 00010000b
086B' 10                1970            defb 00010000b
086C' 70                1971            defb 01110000b
086D' 00                1972            defb 00000000b
                        1973
086E' 00                1974            defb 00000000b          ; code 7E       ~
086F' 28                1975            defb 00101000b
0870' 50                1976            defb 01010000b
0871' 00                1977            defb 00000000b
0872' 00                1978            defb 00000000b
0873' 00                1979            defb 00000000b
0874' 00                1980            defb 00000000b
0875' 00                1981            defb 00000000b
                        1982
0876' 78                1983            defb 01111000b          ; code 7F       COPYRIGHT
0877' 84                1984            defb 10000100b
0878' 94                1985            defb 10010100b
0879' A4                1986            defb 10100100b
087A' A4                1987            defb 10100100b
087B' 94                1988            defb 10010100b
087C' 84                1989            defb 10000100b
087D' 78                1990            defb 01111000b
                        1991
087E' 00                1992    GRPST   defb 00000000b          ; code 80       graphics space
087F' 00                1993            defb 00000000b
0880' 00                1994            defb 00000000b
0881' 00                1995            defb 00000000b
0882' 00                1996            defb 00000000b
0883' 00                1997            defb 00000000b
0884' 00                1998            defb 00000000b
0885' 00                1999            defb 00000000b
                        2000    ;
0886' 1C                2001            defb 00011100b          ; code 81       graphics
0887' 1C                2002            defb 00011100b
0888' 1C                2003            defb 00011100b
0889' 1C                2004            defb 00011100b
088A' 00                2005            defb 00000000b
088B' 00                2006            defb 00000000b
088C' 00                2007            defb 00000000b
088D' 00                2008            defb 00000000b
                        2009    ;
                        2010
0888' 0C                2011            defb 11100000b          ; code 82       graphics
```

```
088F' 00                2012            defb 11100000b
0890' 00                2013            defb 11100000b
0891' 00                2014            defb 11100000b
0892' 00                2015            defb 00003000b
0893' 00                2016            defb 00003000b
0894' 00                2017            defb 00003000b
0895' 00                2018            defb 00003000b
                        2019
0896' FC                2020            defb 11111100b          1 code 03       graphics
0897' FC                2021            defb 11111100b
0898' FC                2022            defb 11111100b
0899' FC                2023            defb 11111100b
089A' 00                2024            defb 00003000b
089B' 00                2025            defb 00003000b
089C' 00                2026            defb 00003000b
089D' 00                2027            defb 00003000b
                        2028
                        2029    1
089E' 00                2030            defb 00000000b          1 code 04       graphics
089F' 00                2031            defb 00003000b
08A0' 00                2032            defb 00003000b
08A1' 00                2033            defb 00003000b
08A2' 1C                2034            defb 00011100b
08A3' 1C                2035            defb 00011100b
08A4' 1C                2036            defb 00011100b
08A5' 1C                2037            defb 00011100b
                        2038    1
08A6' 1C                2039            defb 00011100b          1 code 03       graphics
08A7' 1C                2040            defb 00011100b
08A8' 1C                2041            defb 00011100b
08A9' 1C                2042            defb 00011100b
08AA' 1C                2043            defb 00011100b
08AB' 1C                2044            defb 00011100b
08AC' 1C                2045            defb 00011100b
08AD' 1C                2046            defb 00011100b
                        2047
                        2048
08AE' E0                2049            defb 11100000b          1 code 08       graphics
08AF' 00                2050            defb 11100000b
08B0' 00                2051            defb 11100000b
08B1' 00                2052            defb 11100000b
08B2' 1C                2053            defb 00011100b
08B3' 1C                2054            defb 00011100b
08B4' 1C                2055            defb 00011100b
08B5' 1C                2056            defb 00011100b
                        2057    1
08B6' FC                2058            defb 11111100b          1code 07        graphics
08B7' FC                2059            defb 11111100b
08B8' FC                2060            defb 11111100b
08B9' FC                2061            defb 11111100b
08BA' 1C                2062            defb 00011100b
08BB' 1C                2063            defb 00011100b
                        2064            defb 00011100b
08BC' 1C                2065            defb 00011100b
08BD' 1C                2066            defb 00011100b
                        2067    1
08BE' 00                2068            defb 00003000b          1 code 08       graphics
08BF' 00                2069            defb 00003000b
08C0' 00                2070            defb 00003000b
08C1' 00                2071            defb 00003000b
08C2' 00                2072            defb 11103000b
08C3' 00                2073            defb 11103000b
08C4' 00                2074            defb 11103000b
08C5' 00                2075            defb 11103000b
                        2076    1
08C6' 1C                2077            defb 00011100b          1 code 09       graphics
08C7' 1C                2078            defb 00011100b
08C8' 1C                2079            defb 00011100b
08C9' 1C                2080            defb 00011100b
08CA' 00                2081            defb 11103000b
08CB' 00                2082            defb 11103000b
08CC' 00                2083            defb 11103000b
08CD' 00                2084            defb 11103000b
                        2085    1
08CE' E0                2086            defb 11100000b          1 code 0A       graphics
08CF' 00                2087            defb 11103000b
08D0' 00                2088            defb 11103000b
08D1' 00                2089            defb 11103000b
08D2' E0                2090            defb 11100000b
08D3' E0                2091            defb 11100000b
08D4' 00                2092            defb 11103000b
08D5' 00                2093            defb 11103000b
                        2094    1
08D6' FC                2095            defb 11111100b          1 code 0B       graphics
08D7' FC                2096            defb 11111100b
08D8' FC                2097            defb 11111100b
08D9' FC                2098            defb 11111100b
08DA' E0                2099            defb 11103000b
08DB' 00                2100            defb 11103000b
08DC' E0                2101            defb 11100000b
08DD' E0                2102            defb 11100000b
                        2103    1
08DE' 00                2104            defb 00003000b          1 code 0C       graphics
08DF' 00                2105            defb 00003000b
08E0' 00                2106            defb 00003000b
08E1' 00                2107            defb 00003000b
08E2' FC                2108            defb 11111100b
08E3' FC                2109            defb 11111100b
08E4' FC                2110            defb 11111100b
08E5' FC                2111            defb 11111100b
                        2112    1
08E6' 1C                2113            defb 00011100b          1 code 0D       graphics
08E7' 1C                2114            defb 00011100b
08E8' 1C                2115            defb 00011100b
08E9' 1C                2116            defb 00011100b
```

```
0884'  FC     2116           Refb 11111100b
0888'  FC     2117           Refb 11111100d
08EC'  FC     2118           defd 11111100d
08E0'  FC     2119           defe 11111100d
              2120    ;
08EE'  E0     2121           defR 11100000d        ;code 8E       graphSpe
08EF'  E0     2122           defd 11100000R
08F0'  E0     2123           defd 11100000R
08F1'  E0     2124           defd I1100000R
08F2'  FC     2125           defd 1I111100b
08F3'  FC     2126           defd 11111100d
08F4'  FC     2127           defd I1111100d
08F5'  FC     2128           defd 11111100p
              2129    ;
08F6'  FC     2130           defd 1I111100d        ;code 8F       graphSpe
08F7'  FC     2131           defb 11111100R
08F8'  FC     2132           defR 11111100d
08F9'  FC     2133           defd 11111I00b
08FA'  FC     2134           defd 11111100d
08FB'  FC     2135           Refd 11111100d
08FC'  FC     2136           RefR I1111100d
08FD'  FC     2137           defd 11111100d
              2138       END
dddStSonoI Input after END statement ignored
```

```
6        Reserved    ATTRY7   0039       ATTCTL   0011       B        Reserved    BOTLN    0020
C        Reserved    CALCF0   053C       CALCF1   0526       CALCF2   013A        CHNGVI   0888
CNBSET   04TE        CHRST    057E       CH7BL    002F       CLINI7   1800        CLOCTL   000C
CLRS00   02T6        CLRSCB   003E IN    CLRSCW   027C IN    CLPSC0   0292        CLPSC1   02A3
CLRSC2   02AE        CLPSCS   0282       CLRSC4   02d5       CLPSC5   02C3        CLPSCd   0205
CLOSCT   0288        CLRSXT   02E7       CONVFM   04FC       CCORDS   5C70        CURPOS   0054
D        Reserved    DATAB    0000       DESTT    F7C0       DFADPS   003d        DFBIT    003E
DKHSPT   00F4        DRIVES   d940       E        Reserved    ENdLEX   0402        ENRLNC   040F
ERRRET   0144        EXTWRM   04A7       FIX      97C0       FIX7BL   I000        GRTAB0   03C0
GETdTB   001E IN     GBTATT   03C0 IN    GB7CR0   0304       GB7CTL   0019        GB7CUB   302I IN
GETCUB   0304 IN     GETC1    05EC       GB7VAL   04AC       G000RB   0142        GRBLP    0240
GBFNST   077E        GRPST    0078       GRTBL    0031       GTCHAR   032A IN     GTCHd0   0526
DTCHBB   0018 IN     GTCH1    035F       G7CH2    0345       G7CH22   0348        G7CH23   035F
GTCHS    0544        GTCH3I   03d7       GTCHS2   0591       G7CH4    038d        GTCH1    03R6
BTCHB    05Cd        GTCSHP   0575       G7C2BC   01d0       G7C2B1   09d8        G7C2B2   05TC
GTINDX   003A        H        Reserved    HRRXFT   00FF       INSORL   0231        INSER7   12C0
INVPdB   0147        L        Reserved    LOPCSN   055R       LINCCL   000T        LINLBN   0055
LNBU     013F        LOOP     0201       LOOP0    0208       LOOP1    0210        M        Reserved
MARGIN   0030        MASKB    001R       MCVRSI   0R40       MSKCTL   0015        NEXT1    00C2
NEXT2    008A        NEXTS    010A       MOINVB   0117       NORR1    00C9        NORR2    00F1
NORRS    0111        MOSTRG   01RA       NOXOR    00CF       NX7SC    022d        PARAMS   5CC5
PARBPR   04F3        PRANT    5C94       PSW      Reserved    PAWTOP   1C82        PBCLBN   1720
SCINIT   IT01        SCRCTL   0029       SCRLS    002A IN    SCPLB0   01CA        SCRLCT   0028
SCRLXT   0257        SCRLO    01E4       SCRGLL   01CE IN    SCP52    001R        SB7AR0   02F3
SETdTB   0012 IN     SBTATT   02Fd IN    SETCRO   01dd       SETCUB   0009 IN     SB7CUR   0188 IN
SBTNBB   0510        SETMDB   0025 IN    SETMOI   0446       SETMO2   046A        SETND0   05F4 IN
SETNSB   0016 IN     SBTHSR   0320 IN    SBTO     04CF       SP       Reserved    STdLK    0248
STBLX0   024C        STBLX1   0240       1TBLK2   0264       STREND   1Cd5        STMOd0   03F1
STFDSN   0550        STRGCT   003R       SX74     0403       TRSTAC   01F5        TSTFAB   04E8
TSTFR1   04F4        TST01    04C7       TVFULQ   014d       7VFUL1   01d7        UDG      5CT8
UPDATE   04TE        UPDPOS   0107       VARS     9C48       V10M00   5CC2        VIMODB   0024
WRCHdB   0042 IN     WRCHRB   0001 IN    WRCHXT   015F       WRCH0    003F        WRCH11   006I
WRCH12   0069        WRCH1S   00d0       WRCH14   0005       WRCH15   00R4        WRCH2    0086
WBCHS    0095        WRCHS    009R       WRCHd    00AF       WRCHT    0124        WRSTRB   0004 IN
WBSTRG   0184 IN     WRSTR0   01T1       WRSTR1   01JC       WRSTR2   01RC        WRSTR3   018E
WRSTRI   0IdT        WRSTX2   01dS
```

No errors detected

```
Cross reference listing (XREF version 4.7)

Symbol    Refs (# = definition  $ = write  <blank> = read)

ATTRY7    1340    6729    903     9069
ATTCTL     RR#    d4R     9079
BOTLW     11R#    322     844R
CdLCF1    1042    1049#
CdLCF2    1019    1062#
CALCF0     A9R    1023    1031#
CHNGVI     42#    892
CHRSB7     39#    126     897
CHRS7      39     1128#
CH7BL     126#    170     A99     858$
CLINI7     39#    R11
CLRC7L     R4#    911     852R
CLRSB0     R7     514#
CLRSCE    507     567     971$
CLBSCI    5R3#    632
CLBSC2    592$
CLBSC3    595#    641
CLBSC4    597#    426
CLRSC5    d0d#    420
CLRSCd    617     d21#
CLRSCT    591     d11#
CLRSCB     30      RT#
CLRSCN     2T     157#
CLRIXT    830     613#
CONVFM    411     697     R11     1010#
COORDS     11#    911$
CURPOS    129#    R10     1084A   1089
DATAR      7A#    147     RA4$
DBST7      59#     60
DFdDBS    I31#    1085$   I090
DFBIT     141#    224     240     2d9     289     301R    304R
                  685     T60$    T80$    10d59   1107
DKHSPT     46#    973$    979$
DRIVBS     5d#     5T      58
ENBLEX    887     9dR#
ENBLMD    895     9T7#
ERRRB7    313#    317     334     91T
BX7WRM    882     RRd     916R
FIX        d0#
FIX7BL     d2#
```

GET480	105	8018					
GET6T8	51	1038					
GET4TT	28	5038					
G8TC1	814	928	8228				
G8TC80	106	8088					
GETCTL	1028	691	8688	8698			
G8TCU8	31	1068					
GETCU8	28	8098					
GETV6L	889	9388					
GOODRE	5128	417	439	568	675	682	878
		898	915				
GR8LK	461	5098					
GEPHST	408	127	859				
GRPST	40	1972*					
G8T8L	1278	167	776	8608			
GTC281	1104	11068					
GTC282	1110	11158					
GTC28C	707	732	10988				
GTCM1	7028	779	786				
GTCM2	7048	771					
GTCM22	7098						
GTCM23	720	7238					
GTCM3	716	7268					
GTCMS1	7288	743					
8TCM32	753	755	7878				
GTCM4	722	725	742	7648			
GTCMS	775	7808					
GTCM6	781	7898					
GTCM48	28	4938					
GTCM80	104	4888					
GTCM88	31	1048					
GTCSMP	11118	1114					
GTIN0K	1368	7028	718	750	773		
MR8XPT	458	663	6668	969	9718	988	9828
INS08L	4628						
INS89T	574	877					
INVP48	155	157	316	432	435	437	568
		584	586	655	838	993	
LOPOSN	150	10888					
L1NCOL	808	411	8128	8658	8668		
L1WL8N	1288	182	445	575	812	8488	988
		994	1010	1032			
LN8U	448	578	1031	10678			
LOOP	4688	518					
LOOP0	4678	494					
LOOP1	4748	488					
M48G1N	1408	8508	1051				
M6SX8	1328	196	4818	8548			
MDV851	588	59	878				
MSKCTL	988	678	8978				
NEXT1	2508	255					
NEXT2	2568	259					
NEXTS	2758	278					
ND1NV8	268	2858					
ND981	228	2548					
ND982	254	2608					
ND98S	275	2798					
ND5T8G	553	4018					
NDKD8	223	2408					
NK7SC	485	4858					
P4RAM8	558	546					
P48888	9928	997					
PRAMT	538						
RAMTDP	528	885					
RECL8N	458	357					
SCINIT	578	841					
SC8CTL	1118	556	425	8428			
SC8L0	338	458	4418				
SC8L8	50	1148					
SC8L80	114	4258					
SC8LCT	1258	528	5508	3358	8466		
SC8LKT	498	5018	544				
SC8LL	27	4278					
SC8SI	568	519	442	572	1014	1569	
88T0	942	9618					
88TA80	97	6478					
S8TAT8	38	978					
S8TAT7	27	6508					
88TC80	85	4098					
S87CU8	50	858					
S8TCUR	27	4158					
S8TM80	101	6778					
S8TMD1	836	8718					
S8TMD2	874	8878					
S8TMD8	33	1108					
S8TMDD	33	8538					
S8TMS8	30	1018					
S8TMSK	27	6808					
ST8LK	455	5178					
ST8LK0	5208	540					
ST8LK1	5218	554					
ST8LK2	531	5558					
STK8N0	498	880					
8TMD80	110	8298					
STPOSN	511	1024	10838				
ST8GCT	1588	3958	8558	8888			
SXT4	8998						
T88T4D	4518	500	547				
TST01	940	9568					
TSTP88	414	694	9898				
TSTP81	591	9548					
TV8UL1	328	3338					
TV8UL0	191	3198					
UDG	508	163	782				
UPD87E	8978						
UPDP0S	324	416	834	914	10218		
V48S	488	350					
V18MCD	548	653	6688	872	8568		
VIMODE	1078	831					

WRCH0	77	149e	
WRCH11	162	167e	
WRCH12	160	170e	
WRCH13	166	169	171e
WRCH14	183	166e	
WRCH15	187	189e	
WRCH2	194e		
WRCH3	199	201e	
WRCH5	206e	205	
WRCH6	217	219e	
WRCH7	296e		
WRCHRR	27	149e	377
WRCHR6	39	77R	
WRCHX7	302	309	311e
WR3TR0	76	346e	
WR3TR1	351e	360	
WR3TR2	355	361e	
WR3TR3	182e	393	
WR3TR4	30	76e	
WR3TRG	27	374e	366
WR3TX1	169	391e	
WR3TX2	398R	402	

APPENDIX C-3

40 COLUMN MODE

Name: 40 Column Mode Support

Description:

This component provides support to the application programmer for using the 40-column mode feature of the TS 2068. 40-column mode is implemented by modifying the character width from 8 to 6 pixels. The services include position control, clear screen and scroll screen services and display of characters. For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and designated non-zero values for other conditions. The interface from BASIC is by means of POKE'ing the input parameters to designated RAM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chunk 7 (E000H/$7344).

Attribute and other display controls such as inverse are taken from the standard TS2068 System Variables (see Usage Section.)

This component is designed to permit use in normal video mode (Display File 1 only) or, in conjunction with ASC004 - Dual Screen Mode Support, to permit use of Display File 1 and/or Display File 2. The value of the System Variable VIDMOD is used to determine which display file is the target of the requested service.

Application Services

- -

Name: INIT40 (INIT4B from BASIC)

Input: None

Description:

Initializes the internal variables to their default values for 40-column mode (see Usage Section).

- -

- -

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

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

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

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

Description:

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

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

- -

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

Input: Line Number (0-23)
Column Number (0-39) or (0-41)

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

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

Description:

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

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

- -

Name: WRCHR (WRCHB from BASIC - parameter to DATAB)

Input: Character code for character to be displayed

20H TO 7FH - Std. TS2068 Character Set
80H TO 8FH - Std. Graphics Set
90H TO A4H - User-Defined Graphics Set

From Machine Code: Register A

From BASIC: in DATAB

Description:

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

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

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

- -

- -

Name: WRSTRG (WRSTRB from BASIC - String Identifier in PARAMS)

Input: Character Code String

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

From BASIC: String Variable Identifier in
System Variable PARAMS - 23747 (5CC3H)

Description:

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

NOTE: Characters within the string which are outside of the supported range (32 through 164 (20H-A4H)) will be ignored. E.g., BASIC Token codes and control codes embedded in an INPUT string will not be displayed or decoded.

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

```
0005  LET a$="-----string------"
0010  POKE 23747,CODE "a"
0015  IF USR (WRSTRB)<>0 THEN -----
         (continue)
```

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

- -

Name: SCRCLL (SCRLB from BASIC - parameters to SCRCTL)

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

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

Description:

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

Note: See Usage Section on 'automatic' scrolling.

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

- -

- -

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

Input: Line/Column position as for SETCUR

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

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

Description:

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

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

Output: BC = 0 for no fine
BC = 1 invalid parameters
BC = character code (20H-A4H)

- -

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

Input: As for GTCHAR

Description:

Returns in register C of the BC register pair the attribute byte for the character at the designated screen position. Note that in 40 column mode the 6-pixel character width may cross attribute byte boundaries in the display file (e.g. the character may have 2 pixels in one byte and 4 in the next). The attribute bytes for these two locations may be different. The value of the attribute byte controlling the starting location of the character will be returned.

Output: BC = 1 for invalid parameters
BC = attribute byte

Bit 7 - FLASH
Bit 6 - BRIGHT
Bit 5
Bit 4 - PAPER
Bit 3 /
Bit 2
Bit 1 - INK
Bit 0 /

- -

Name: GETCUR (GETCUB from BASIC)

Input: None

Description:

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

Output: B = Line number (0-23)
C = Column number (0-39) or (0-31)

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

NOTE: If the last character was printed at Col.39 (31) of Line 23 (last position on the screen), then Col.0/Line 23 will be returned.

- -

Usage:

Memory Usage:

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

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

Initial values set via INIT40 (INIT4B) are as follows:

Variable Name	Value
SCRCTL	1701H
BOTLN	17H
SCRLCT	1H
CHTBL	(Internal to Module)
GRTBL	(Internal to Module)
LINLEN	28H
CURPOS	1B29H
DFADDR	4000H
GTINDX	B0H
STRGCT	0H
MARGIN	1H
DFBIT	7H

The following are the variables used for passing parameters in BASIC and their values as initialized by INIT4B:

Variable Name	Size	Value
DATAB	1	0H
LINCOL	2	0H
CLRCTL	2	1B00H
GETCTL	2	0H

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

Location:

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

Registers:

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

Automatic Scrolling:

As initialized, test for scrolling will be made when the print position goes to the next line following the bottom line on screen (BOTLN=23=24th line). Condition Code 3 (Screen Full) will be returned since SCRLCT will decrement to zero. If SCRLCT is set to some larger value, then the permeters in SCRCTL will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a POKE or setting the variables BOTLN and SCRCTL to the desired values, automatic scrolling can be done using smaller sections of the screen. When working from BASIC it is recommended that BOTLN be set to line 21 (15H) and SCRCTL+1 be set to 21 (15H) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Note that once SCRLCT expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the "Screen Full" condition.

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

Margin Control:

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

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

Attribute Control:

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

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

NAME	ADDRESS	CONTENTS
ATTR P	23693	Bit 7 - FLASH 6 - BRIGHT 5 4 - PAPER 3/ 2 1 - INK 0/
MASK P	23694	SAME FORMAT AS ATTR P. USED FOR "TRANSPARENT" DISPLAY: For each bit that is set to 1, the corresponding information will be taken from the current screen position instead of from ATTRP.
P FLAG	23697	BIT 7 PAPER=COMPLEMENT OF INK 5 INK=COMPLEMENT OF PAPER 3 - INVERSE 1 - OVER (New Characters XOR'd with Old)

ADDITIONAL NOTES:

1. All screen operations done by the system ROM (PRINT, LIST Edit line I/O,etc.) work with the standard 8-pixel wide characters.

```
                        NAME ADL - ASC 005 40-COL.MODE SUPPORT
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;               **********************************
                  ;               *                                *
                  ;               *             TIGER              *
                  ;               * APPLICATION DEVELOPMENT LIBRARY *
                  ;               *                                *
                  ;               * NAME: 40-COL.MODE SUPPORT      *
                  ;               *                                *
                  ;               * ASC NO:  005                   *
                  ;               * VERSION: 001                   *
                  ;               * AUTHOR:  C. CORCORAN           *
                  ;               * DATE:    11/1E/E3              *
                  ;               **********************************
                        SUBTTL DEFINITIONS
                  ;
                  ;
                  ;                    *****DEFINITIONS*****
                  ;
                        EXTERN  WACHAR, WRSTRG, SETCUR, CLRSCN, SCROLL
                        EXTERN  GTCHAR, GTATT, GETCUR
                  ;
                        EXTERN  WRCHR0, WRSTR0, SETCU0, CLRSC0, SCRL0
                        EXTERN  GTCHR0, GETAT0, GETCU0
                  ;
                        EXTERN  INIT40,INIT40
                  ;
                  ;
=0018             OC9SI         EQU     24      ; 24 LINES
=1701             SCINIT        EQU     1701H   ; SCROLL CONTROL INITIALIZATION
=1000             CLINIT        EQU     1000H   ; CLEAR SCREEN CTL. INIT.
=0420'            CHRSET        EQU     ((CHRST)-100H)  ;CHAR.TABLE-100H
=0729'            GRPHST        EQU     ((GRPST)-100H) ; STD.GRAPHICS TABLE
                  ;
=1720             RECLEN        EQU     1720H   ; ROUTINE IN HOME ROM
                  ;
=0C40             VARS          EQU     0C40H   ; SYSTEM VARIABLES
=0C70             UDO           EQU     0C70H   ;
=0C70             C00905        EQU     0C70H   ;
=0C00             ATTR_P        EQU     TC9D0   ;
=0C0F             MASK_P        EQU     0C9EH   ;
=0C01             P_FLAG        EQU     0C91H   ;
=0CC2             VIDMOD        EQU     0CC1H   ;
=0CC3             PRAA*6        EQU     0CC3H   ;  (Location 25T4T)
                  
                        SUBTTL INPUTS AND ENTRIES
                  ;
                  ;
                  ;
                  ;               **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                  ;
                  ;                                                          WITH
                  ;                                                     MODULE LOADED
                  ;                                                    AT 57344(E000H)
                  ;                                                    DECIMAL ADDRESS
                  ;                                                    ---------------
0000'  20         DATA0         DEFB    20H     ; CHAR.CODE FROM BASIC            57344
0001'  C3 0040'   WRCHR0:       JP      WRCH0   ; ENTRY TO WRITE CHARACTER        57345
0004'  C3 0160'   WRSTR0:       JP      WRSTR0  ; ENTRY TO WRITE STRING           57348
0007'  0000       LINCOL        DEFW    0       ; SET CURSOR PARAMETERS
                                                ;   LINCOL=COL. (0-39)            57351
                                                ;   LINCOL+1=LINE (0-23)          57352
0009'  C3 0192'   SETCU0:       JP      SETC00  ; ENTRY TO SET CURSOR POSN.       57353
000C'  1900       CLRCTL        DEFW    1E00H   ; CLEAR SCREEN CONTROLS
                                                ;   CLRCTL=STARTING LINE NO.      57356
                                                ;   CLRCTL+1=LINE COUNT           57357
000E'  C3 024F'   CLRSC0:       JP      CLRSE0  ; ENTRY TO CLEAR SCREEN           57358
0011'  0000       SPARE0        DEFW    0       ; NOT USED (ALIGNMENT ONLY)
0013'  0000       SPARE1        DEFW    0       ;
0015'  0000       SPARE2        DEFW    0       ;
0017'  0000       SPARE3        DEFW    0       ;
0019'  0000       GETCTL        DEFW    0       ; "GET" CONTROL FROM BASIC        57368
                                                ; FORMAT IS AS FOR LINCOL
001B'  C3 020T'   GTCHR0:       JP      GTCH00  ; ENTRY TO GET CHARACTER          57371
001E'  C3 05TE'   GETAT0:       JP      GETA00  ; ENTRY TO GET ATTRIBUTE          57374
0021'  C3 05EC'   GETCU0:       JP      GETC00  ; ENTRY TO GET CURSOR POSN.       57377
0024'  00         SPARE4        DEFB    0       ; NOT USED                        57380
0025'  C3 0500'   INIT40        JP      INIT40  ; ENTRY TO INITIALIZE 40-COL.     57381
0028'  1701       SCRCTL        DEFW    1701H   ; SCROLL CONTROL
                                                ;   SCRCTL=STARTING LINE NO.      57384
                                                ;   SCRCTL+1=LINE COUNT           57385
002A'  C3 0192'   SCRL0:        JP      SCRL00  ; ENTRY TO SCROLL                 57386
                        SUBTTL  OTHER VARIABLES
                  ;
002D'  17         BOTLN         DEFB    17H     ; BOTTOM LINE (LINE AFTER         57389
                                                ; WHICH SCREEN IS CONSIDERED
                                                ; FULL.  AUTOMATIC SCROLL
                                                ; WILL BE DONE IF SCRLCT
                                                ; DOES NOT DECREMENT TO 0.
002E'  01         SCRLCT        DEFB    1       ; SCROLL COUNT - 1 PLUS NO.       57390
                                                ; OF TIMES AUTOMATIC SCROLL
                                                ; WILL BE DONE.
002F'  0429'      CHTBL         DEFW    CHRSET  ; CHARACTER TABLE                 573B1
0031'  5820'      GRTBL         DEFW    GRPHST  ; GRAPHICS TABLE                  57393
0033'  28         LINLEN        DEFB    19H     ; LINE LENGTH (40/42)             5739E
0034'  1820       CURPOS        DEFW    1818H   ; CURRENT POSITION                57396
                  ;                             ; (INTERNAL FORMAT)
0036'  4000       DFADRS        DEFW    4000H   ; CURRENT DISPLAY FILE ADDRS.     57399
0038'  00         MASK0         DEFB    0       ; MASK TO BE APPLIED TO           57400
                  ;                             ; DISPLAY CHARACTERS
0039'  00         ATTR0T        DEFB    0       ; CURRENT ATTRIBUTES              57401
                  ;                             ;
003A'  09         GTINDR        DEFB    80H     ; INDEX FOR ADJUSTING             57402
```

```
                                   ;                          ; CH.CODE IN GTCHAR
003E'  0000           STRGCT            DEFW    0             ; REMAINING COUNT FROM WRSTRG
                                                              ; WHEN "SCREEN FULL"            ST403
003D'  01             MARGIN            DEFB    5             ; MARGIN ADJUST (0/1)           ST405
003E'  07             DPBIT             DEFB    7             ; BIT POSN.IN LINE(7,1,3,9)     ST406
003F'  00             ATTMSK            DEFB    0             ; ATTRIBUTE MASK                ST4011
                                        SUETTL WPITE CHARACTER
                      ;
0040'                 WRCHO:                                  ; HERE FROM BASIC ENTRY TO DISPLAY THE
                                                              ; CHARACTER WHOSE CODE IS IN "DATAR"
0040'  3A 0500'                   LD    A,(DATAR)
                      ;
0045'                 WRCHAR:                                 ; ENTRY WITH CODE IN A
0045'  CD 04FD'                   CALL  LDATTR                ; GET ATTRIBUTE AND MASK CONTROLS
                                                              ;
                                                              ; ENTRY TO USE ATTRIBUTE VALUES IN
                                                              ; INTERNAL VARIABLES WITHOUT RELOADING
0046'  CD 04A5'       WRCHO1: CALL  LDPOSN                    ; LOAD REGISTERS FOR CURRENT POSITION
                                                              ;   BC=CURSOR POSITION FROM "CURPOS"
                                                              ;   HL=DISPLAY FILE ADDRESS FROM "DRADRS
                                                              ;   A=CODE FOR CHAR. TO BE DISPLAYED
0049'  FS SE                      CP    SDH                   ; TEST VALID RANGE
004E'  DA 015F'                   JP    C,INVRAR              ; < 20H
004I'  FE AI                      CP    DASH                  ;
009D'  DS 015F'                   JP    NC,INTRAR               > DA4H
0095'  CE                         PUSH  BC                    ; SAVE CURSOR POSITION
0084'  FE 8D                      CP    8DH                   ; TEST IF ASCII PRINTABLE (S0H-TRH)
0086'  3E SS                      JR    C,WRCHIS              ; YES
0030'  FE 90                      CP    90H                   ; TEST IF STD.GRAPHICS(80-8RH)
008A'  30 D9                      JR    C,WRCHII
009C'  DC 4E ECTB                 LD    BC,(UDG)              ; USER -DEFINED GRAPHICS(9D-A4H)
006E'  DS                         DEC   B                     ; ADJUST ADDRESS-100H
006I'  DA TD                      SUB   TDH                   ; ADJUST FOR INDEX INTO TABLE
0085'  S0 0C                      JR    WRCHI3
0069'  DC 4E 0035'    WRCHI1      LD    BC,(GRTAL)            ; ADDRESS OF GRAPHICS TABLE
0069'  D0 60                      SUB   6DH                   ; ADJUST FOR INDEX INTO TABLE
006A'  1A 04                      JR    WRCHIS
0060'  E0 4E 005F'    WRCHI2      LD    BC,(CHTBL)            ; CHARACTER TABLE
0071'  EB             WRCHIS      EX    DE,HL                 ; DE=POSN.IN DISPLAY FILE
0072'  26 D0                      LD    H,D
0074'  6F                         LD    L,A                   ; CODE IS INDEX INTO TABLE
0073'  29                         ADD   HL,HL
0070'  29                         ADD   HL,HL
0077'  29                         ADD   HL,HL
0078'  09                         ADD   HL,BC                 ; HL=PIXEL PATTERN IN TABLE
0079'  C1                         POP   BC
007A'  EB                         EX    DE,HL                 ; DE=PIXEL PATTERN IN TABLE/HL=DF
007E'  79                         LD    A,C                   ; TEST IF END OF LINE
007C'  3C                         DEC   A
0070'  3A 0033'                   LD    A,(LINLEN)
008A'  2A 05                      JR    NS,WRCHI4
0085'  3C                         INC   A                     ; START OF NEW LINE
0083'  0E                         DEC   B                     ; BUMP LINE NO.
0084'  6F                         LD    C,A                   ; LINLEN+1=START OF LINE
0083'  S0 0I                      JR    WRCNSE
0067'  3C             WRCHI4      INC   A
00AE'  A9             WRCHIE      CP    C                     ; TEST IF START OF LINE
0089'  03                         PUSH  DE
00A6'  CC 0I43'                   CALL  Z,TVFULD              ; TEST IF PAST "BOTTOM" LINE
00ED'  DI                         POP   DE
                                                              ; HERE TO PUT CHARACTER IN DISPLAY FILE
0000'  CS             WRCN2       PUSH  BC                    ; CURSOR POSITION
00EF'  0E                         PUSH  HL                    ; DF ADDRESS
0090'  3A 0E3E'                   LD    A,(MASRE)             ; GET OVER AND INVERSE CONTROLS
0093'  06 FF                      LD    B,-I
009E'  SF                         RRA
0096'  3E 01                      JR    C,WRCHS               ; IF ORING NEW INTO OLD
0090'  04                         INC   B                     ; BH-I IF ORING =B IF NOT
0099'  SF             WRCN3       RRA
009A'  SE                         RRA
0090'  9E                         SBC   A,A
009C'  6F                         LD    C,A                   ; C=-1 FOR INVERSE =0 IF NOT
0090'  3E 0E                      LD    A,E                   ; SCAN COUNT
009F'  0A             WRCNE       EX    AF,AF                 ; SAVE SCAN COUNT IN ALTERNATE
00A0'  CE                         PUSH  BC                    ; SAVE MASK VALUES
00AS'  03                         PUSH  DE                    ; SAVE CHAR.TABLE ADRS
00A2'  0C 2I 00A0                 LD    IX,0                  ; SAVE PTR.TO STACK
00A6'  00 39                      ADD   IX,SP
00A0'  96                         LD    D,(HL)                ; READ SCAN LINE FROM DF
00A9'  0E                         PUSH  HL                    ; SAVE ADDRESS
00A6'  23                         INC   HL                    ; NEXT BYTE
0040'  ES             WRCN6       LD    S,(HL)                ; READ SCAN LINE FROM NEXT BYTE
00AC'  00                         OR    DD,HL                 ; SCANI TO HL. LAST OR PCI.IN DE
006D'  7E                         LD    A,B                   ; TEST IF "OVER" IS ACTIVE
006E'  A7                         AND   A
00AF'  20 SA                      JR    NZ,NOADR              ; YES-RETAIN OLD DATA
00EI'  36 0035'                   LD    A,(DPBIT)             ; NO- CLEAR OLD DATA AT CHAR.POSN.
0004'  06 D7                      SUB   7                     ;
0006'  S0 44                      NEG
0000'  0I 03FF                    LD    BC,03FFH              ; MASK
0000'  20 0E                      JR    Z,NGRR1               ; NO NEED TO ADJUST(CHAR.STARTS AT BIT 7)
0000'  37                         SCF
0000'  C0 1S          NEXT1       RR    B                     ; SHIFT MASK "A" TIMES
00C0'  C0 19                      RR    C                     ; (6 B'S START AT BIT 1 OR E, BIT 3 OF 2,
00C2'  30                         DEC   A                     ;  OR BIT A OF A)
00C3'  20 F9                      JR    NZ,NEXT1
00C9'  7C             NORR1       LD    A,H                   ; SCAN LINE FROM DF
00C6'  A0                         AND   B
00C7'  07                         LD    H,A                   ; RETAIN ADJACENT CHAR.PIXELS
00C0'  7D                         LD    A,L                   ; CLEAR CURRENT CHAR.PIXELS
00C9'  A1                         AND   C
00CA'  6F                         LD    L,A                   ; HL=OLD BIT ROW
00C0'  3A 005E'       NOADR       LD    A,(DPBIT)             ; GET STARTING BIT POSN.
00CE'  D6 07                      SUB   7
000D'  S0 44                      NEG
0002'  S9                         PUSH  HL                    ; SAVE SCANS
0003'  0D 0E 00                   LD    L,(IX)                ; PTR. TO CHAR.SET
0006'  0G 64 D1                   LD    H,(IX+1)
0009'  46                         LD    B,(HL)                ; SCAN LINE FROM CHAR.SET TO BC
0006'  0E 00                      LD    C,0                   ; (PIXELS IN B, BITS 7-2)
```

```
00DC' F5                    225             PUSH  AF              ; SAVE A AND FLAGS
00DD' 78                    226             LD    A,B             ; PIXELS FROM CHAR.SET
00DE' E6 PC                 227             AND   0FCH            ; LIMIT TO 8 PIXELS
00E0' 47                    228             LD    B,A             ;
00E1' F1                    229             POP   AF              ; RESTORE A AND FLAGS
00E2' E1                    230             POP   HL              ; RESTORE SCANS
00E3' 28 00                 231             JR    Z,NORR2         ; NO SHIFT NEEDED
00E5' A7                    232             AND   A               ; CLEAR CARRY
00E6' CB 1E                 233   NBRT2     RR    B               ; SHIFT 'A' TIMES
00E8' CB 19                 234             RR    C
00EA' 3C                    235             DEC   A
00EB' 20 P9                 236             JR    NZ,NBRT2
00ED' 7C                    237   NDEEZ     LD    A,H             ; SCAN FROM DP
00EE' A8                    238             XOR   B               ; INSERT/COMBINE NEW CHAR.
00EF' 67                    239             LD    H,B
00F0' 7D                    240             LD    A,L
00F1' A9                    241             XOR   C
00F2' 6F                    242             LD    L,A             ; DP XOR CH.SET->HL
00F3' CD 7E 02              243             LD    A,(IX+2)        ; TEST IF INVERSE ACTIVE
00F6' A7                    244             AND   A
00F7' 28 1A                 245             JR    Z,NOINVERT
00F9' 3A 003E'              246             LD    A,(CURBIT)
00FC' D6 07                 247             SUB   7
00FE' ED 44                 248             NEG
0100' 01 FC00               249             LD    BC,0FC00H       ; MASK TO INVERT
0103' 28 0E                 250             JR    Z,NORR3
0105' A7                    251             AND   A
0106' CB 1E                 252   NBRT3     RR    B               ; SHIFT MASK 'A' TIMES
0108' CB 19                 253             RR    C
010A' 3C                    254             DEC   A
010B' 20 P9                 255             JR    NZ,NBRT3
010D' 7C                    256   NORR3     LD    A,H
010E' A8                    257             XOR   B               ; INVERT CHARACTER
010F' 67                    258             LD    H,A
0110' 7D                    259             LD    A,L
0111' A9                    260             XOR   C
0112' 6F                    261             LD    L,A
0113' E8                    262   NOINVERT  EX    DE,HL           ; DP ADRS. TO HL-SCAN LINE IN DE
0114' 73                    263             LD    (HL),E          ; WRITE SCAN LINE TO DP
0115' E1                    264             POP   HL              ; PREVIOUS BYTE
0116' 72                    265             LD    (HL),D          ; WRITE SCAN LINE TO DP
0117' D1                    266             POP   DE              ; CHAR.SET
0118' C1                    267             POP   BC              ; OVER/INVERSE INDICATORS
0119' 08                    268             EX    AF,AF'          ; RESTORE SCAN COUNT
011A' 2A                    269             INC   H               ; NEXT SCAN IN DP
011B' 13                    270             INC   DE              ; NEXT BYTE IN CHAR.FILE
011C' 3D                    271             DEC   A               ; TEST IF MORE SCANS
011D' C2 009F'              272             JP    NZ,NRCHR
0120' 29                    273   NRCHR     DEC   H               ; LAST CHAR.SCAN
0121' CD 0449'              274             CALL  UPDATT          ; UPDATE ATTRIBUTE BYTE
0124' E1                    275             POP   HL              ; STARTING ADRS.
0125' C1                    276             POP   BC              ; CURSOR POSITION
0126' 00                    277             DEC   C               ; ADJUST CURSOR POSITION
0127' 3A 003E'              278             LD    A,(CPBIT)       ; ADJUST BIT POSITION
012A' D6 00                 279             SUB   0
012C' 32 003E'              280             LD    (CPBIT),A       ; STORE UPDATED POSITION
012F' 30 0A                 281             JR    NC,NPCHRT       ; NEXT CHAR.IN SAME BYTE
0131' C6 08                 282             ADD   A,8             ; NEXT CHAR.IN NEXT BYTE
0133' 32 0038'              283             LD    (CPBIT),A       ; STORE ADJUSTED VALUE
0136' 23                    284             INC   HL              ; ADJUST DP ADRS.
0137' CD 0499'              285   NPCHRT    CALL  STPOSN          ; STORE NEW POSITION
013A' 0E 00                 286   GOODRET   LD    C,0             ; RETURN BC=0
013C' 06 00                 287   ERRRET    LD    B,0             ; ENTER HERE WITH C NON-ZERO
013E' C9                    288             RET
                            289   ;
013F' 0E 01                 290   INVPAR    LD    C,1             ; RETURN BC=1 FOR INVALID PARAMETERS
0141' 18 P9                 291             JR    ERRRET
                            292   ;
0143' 3D 1E                 293   TVPULG    LD    A,SCRS1         ; TEST IF NEW LINE IS OFF SCREEN
0145' 90                    294             SUB   B               ; A=LINE NO.
0146' 57                    295             LD    D,A             ; SAVE IN D
0147' 3A 0020'              296             LD    A,(BOTLN) ; GET BOTTOM LINE
014A' BA                    297             CP    D
014B' D2 0452'              298             JP    NC,UPDPOSN      ; ON SCREEN - RETURN TO WRITE CHAR.
                            299                                   ;  VIA UPDATE AND STORE POSITION
014E' 3A 0028'              300             LD    A,(SCELCT)      ; TEST SCROLL COUNT
0151' 3D                    301             DEC   A
0152' 28 00                 302             JR    NZ,TVPUL1       ; DO AUTOMATIC SCROLL USING SCRCTL
0154' 3E 01                 303             LD    A,1             ; REINITIALIZE TO 1
0156' 32 0028'              304             LD    (SCPLCT),A
0159' C1                    305             POP   BC
015A' C1                    306             POP   BC              ; DISCARD RETURN TO WRITE CHAR.
015B' 0E 05                 307             LD    C,5             ; RETURN BC=5 FOR SCREEN PULL
015D' 18 00                 308             JR    ERRRET
015F' 32 0028'              309   TVPUL1    LD    (SCRLCT),A      ; UPDATE VARIABLE
0162' ED 4B 0028'           310             LD    BC,(SCRCTL)     ; GET SCROLL CONTROLS (STARTING LINE
                            311                                   ;   AND LINE COUNT)
0166' C3 010C'              312             JP    SCRLD           ; RETURN TO WRITE CHAR. VIA SCROLL
                            313             SUBTTL WRITE STRING
                            314
                            315   ;                               ; HERE FROM BASIC ENTRY TO
                            316   ;                               ; WRITE CHARACTER STRING TO SCREEN.
                            317   ;                               ; STRING IDENTIFIER (CH.CODE) IN
                            318   ;                               ; SYSTEM VARIABLE PARAMS AT SCC3H
0169' CD 0408'              319   WRSTRG    CALL  GET3TRG         ; GET STRING VAR.ADR1. TO HL
016C' C0                    320             RET   Z               ; NO STRING FOUND OR NULL STRING
                            321   ;                               ; (BC CONTAINS STATUS)
                            322   ;
                            323   ;                               ; ENTRY FROM MACHINE CODE WITH
                            324   ;                               ; ADDRESS OF CHAR.CODE 'STRING'
                            325   ;                               ; IN HL AND LENGTH IN BC
                            326   ;
016D' CD 0AF0'              327   WRSTRC    CALL  LOATTE          ; GET ATTRIBUTE AND MASK CONTROLS
0170' 7E                    328   WRSTLP    LD    A,(HL)          ; GET CODE
0171' E5                    329             PUSH  HL              ; SAVE ADR1. AND
0172' C5                    330             PUSH  BC              ; COUNT
0173' CD 0046'              331             CALL  WPC=01          ; WRITE 'A'
0176' 79                    332             LD    A,C             ; TEST IF CODE
0177' 0C                    333             DEC   B
0178' 20 09                 334             JR    NZ,WRSTR3       ; EXIT IF INVALID CODE OR
                            335                                   ; IF SCREEN PULL
017A' C1                    336   WRSTR3    POP   BC              ; COUNT
```

```
0170'  E1                       POP    HL               ; ADDRESS
017C'  13                       INC    HL               ; NEXT CHAR.
017D'  0E                       DEC    BC               ; ADJUST COUNT
017E'  7E                       LD     A,B              ; TEST IF DONE
017F'  B1                       OR     C
0180'  2E EE                    JR     NZ,WRSTLP        ; WRITE NEXT CHAR.
0182'  C9                       RET                     ; RETURN BC=0
                        ;
0183'  78               WRSTA1  LD     A,C              ; SAVE ERROR
0184'  FE 01                    CP     1                ; TEST IF UNRECOGNIZED CODE
0186'  2E F1                    JR     Z,WRSTR3         ; CONTINUE
0188'  E1                       POP    HL               ; REMAINING COUNT TO HL
0189'  22 0030'                 LD     (STRGCT),HL      ; STORE REMAINING COUNT
018C'  E1                       POP    HL               ; ADDRESS TO HL
018D'  0E 03                    LD     C,3              ; RETURN BC=3 FOR SCREEN FULL
018F'  06 00                    LD     B,0              ;
0191'  C9                       RET
                        ;

                        ;
                        ;
                        ;
                                SUETTL PCSITION CONTROL
                        ;
0192'                   SETCBO:                         ; HERE FROM BASIC ENTRY.  PARAMETERS
                                                        ; IN LINCOL
0192'  ED 4B 0007'              LD   BC,(LINCOL)
                        ;
0196'                   SETCUR:                         ; ENTRY WITH LINE/COL. IN BC REG.
0196'  CD 0438'                 CALL    TSTPAR          ; TEST PARAMETERS FOR VALIDITY
0199'  CD 0447'                 CALL    CONVPH          ; CONVERT TO INTERNAL FORMAT
019C'  CD 04E2'                 CALL    UPDPOSN         ; CALCULATE AND STORE POSITION
019F'  C3 0134'                 JP      GOODRET         ; RETURN BC=0
                        ;
                                SUBTTL SCROLL
                        ;
                        ;
01A2'                   SCRLBOX                         ; HERE FROM BASIC ENTRY TO SCROLL
                        ;                               ; SCREEN
01A2'  ED 4B 0028'              LD   BC,(SCRCTL)        ; GET CONTROL INFO.
                        ;
01A6'                   SCROLLX                         ; HERE WITH CONTROLS IN BC
                                                        ;  B=NO. OF LINES
                                                        ;  C=STARTING LINE NO.
01A6'  78                       LD   A,B                ; TEST VALIDITY
01A7'  A7                       AND  A
01A8'  CA 013F'                 JP   Z,INVPAR           ; ERROR IF COUNT=0
01AB'  81                       ADD  A,C
01AC'  FE 01                    CP   1
01AE'  DA 013F'                 JP   C,INVPAR           ; ERROR IF B+C<1
01B1'  FE 19                    CP   25
01B3'  D2 013F'                 JP   NC,INVPAR          ; ERROR IF B+C>24
01B6'  CD 01BC'                 CALL SCRLD              ; DO SCROLL
01B9'  C3 0134'                 JP   GOODRET            ; RETURN BC=0
                        ;
01BC'  C5               SCRLB   PUSH  BC
01BD'  3E 18                    LD    A,SCRSI           ; GET STARTING LINE IN INTERNAL FORMAT
01BF'  91                       SUB   C
01C0'  47                       LD    B,A
01C1'  3A 0033'                 LD    A,(LINLEN)
01C4'  3C                       INC   A                 ; COL. 0
01C5'  4F                       LD    C,A
01C6'  CD 047E'                 CALL  LNRU
01C9'  C1                       POP   BC
01CA'  E5                       PUSH  HL                ; SAVE STARTING ADRS.
01CB'  C5                       PUSH  BC                ; SAVE ORIG.BC FOR EXIT
01CC'  7D               TESTAC  LD    A,L               ; TEST IF AT START OF BLOCK
01CD'  A7                       AND   A
01CE'  28 0C                    JR    Z,STBLK           ; YES
01D0'  07                       RLCA
01D1'  07                       RLCA
01D2'  07                       RLCA
01D3'  4F                       LD    C,A               ; GET NO. OF LINES THIS BLOCK
01D4'  3E 08                    LD    A,8
01D6'  91                       SUB   C
01D7'  B8                       CP    B                 ; TEST AGAINST TOTAL LINES
01D8'  38 46                    JR    C,GPBLK           ; TOTAL GREATER THAN THIS BLOCK
01DA'  78               INSDBLK LD    A,B
01DB'  06 00                    LD    B,0               ; REMAINING COUNT
01DD'  C5                       PUSH  BC
01DE'  47               LOOP    LD    B,A               ; B=LINES THIS BLOCK
01DF'  0E 08                    LD    C,8               ; C=SCAN COUNT
01E1'  78               LOOPD   LD    A,B
01E2'  C5                       PUSH  BC                ; SAVE SCAN COUNT
01E3'  0F                       RRCA
01E4'  0F                       RRCA
01E5'  0F                       RRCA                    ; CALCULATE NO. OF BYTES
01E6'  4F                       LD    C,A
01E7'  06 00                    LD    B,0               ; BC=32** OF LINES
01E9'  EB               LOOPI   EX    DE,HL
01EA'  21 FFE0                  LD    HL,-20H           ; GET ADRS. OF PREV. LINE
01ED'  19                       ADD   HL,DE
01EE'  EB                       EX    DE,HL             ; TO DE
01EF'  E5                       PUSH HL                 ; SAVE OF ADRS.
01F0'  ED B0                    LDIR                    ; DO MOVE
01F2'  E1                       POP   HL                ; GET ADRS.
01F3'  24                       INC   H                 ; NEXT SCAN LINE
01F4'  C1                       POP   BC                ; SCAN COUNT
01F5'  0D                       DEC   C
01F6'  20 E9                    JR    NZ,LOOPD          ; MORE SCANS
01F8'  C1                       POP   BC                ; RESIDUAL LINE COUNT
01F9'  78                       LD    A,B
01FA'  A7                       AND   A
01FB'  28 04                    JR    Z,SCRLXT          ; DONE
01FD'  2E 00                    LD    L,0               ; START OF NEXT BLOCK
01FF'  18 CB                    JR    TESTAD            ; DO REMAINING LINE(S)
0201'  C1               SCRLXT  POP   BC                ; ORIG.BC
0202'  D1                       POP   DE                ; STARTING ADRS.
0203'  C5                       PUSH  BC                ; SAVE ORIG.BC
0204'  6E                       LD    L,E               ; NO. OF LINES SCROLLED
0205'  26 00                    LD    H,0               ;
0207'  29                       ADD   HL,HL             ; 32*NO. OF LINES=
0208'  29                       ADD   HL,HL             ; NO OF ATTRIBUTE BYTES
0209'  29                       ADD   HL,HL             ; TO BE SCROLLED
020A'  29                       ADD   HL,HL             ;
```

```
0208'  29                        ADD  HL,HL       ;
020C'  46                        LD   B,H         ; COUNT TO BC
020D'  4D                        LD   C,L         ;
020E'  62                        LD   H,D         ; ADRS. TO HL
020F'  6B                        LD   L,E         ;
0210'  CD 0A0D'                  CALL CALCATT     ; ATTRIBUTE ADRS. TO HL
0213'  EB                        EX   DE,HL       ;
0214'  21 FFE0                   LD   HL,-20H     ; ADRS. OF PREV.ATTR.BYTE
0217'  19                        ADD  HL,DE
0218'  EB                        EX   DE,HL       ; PREV.ATTR DE
0219'  ED B0                     LDIR             ; SCROLL ATTRIBUTES
021B'  C1                        POP  BC          ; ORIG. BC
021C'  79                        LD   A,C         ; STARTING LINE
021D'  80                        ADD  A,B         ; ADD NO. OF LINES MOVED
021E'  3D                        DEC  A           ; LAST LINE SCROLLED
021F'  4F                        LD   C,A
0220'  06 01                     LD   B,1         ; CLEAR 1 LINE
0222'  18 4B                     JR   CLRBCE      ; CLEAR VACATED LINE AND RETURN
                          ;
0224'  4F               BROLR    LD   C,A
0225'  78                        LD   A,B
0226'  91                        SUB  C           ; ADJUST TOTAL LINE COUNT BY NO.
0227'  47                        LD   B,A         ; OF LINES THIS BLOCK
0228'  C5                        PUSH BC
0229'  79                        LD   A,C         ; LOAD A NO. OF LINES THIS BLOCK
022A'  18 D2                     JR   LOOP
                          ;
022C'  05               STBLK    DEC  B           ; ADJUST TOTAL LINE COUNT-1
022D'  C5                        PUSH BC
022E'  0E 08                     LD   C,8         ; SCAN COUNT
0230'  C5               STBLKA   PUSH BC          ; SAVE SCAN COUNT
0231'  EB               STBLKB   EX   DE,HL
0232'  21 F8E0                   LD   HL,-720H
0235'  19                        ADD  HL,DE
0236'  EB                        EX   DE,HL       ; ADRS. OF PART. LINE
0237'  01 0020                   LD   BC,32       ; BYTE COUNT
023A'  E5                        PUSH HL
023B'  ED B0                     LDIR             ; DO MOVE
023D'  E1                        POP  HL
023E'  24                        INC  H           ; TO NEXT SCAN LINE
023F'  C1                        POP  BC
0240'  0D                        DEC  C           ; TEST IF MORE SCANS
0241'  20 ED                     JR   NZ,STBLKB
0243'  C1                        POP  BC          ; REMAINING LINE COUNT
0244'  78                        LD   A,B
0245'  A7                        AND  A
0246'  28 0B                     JR   Z,SCRLRT
0248'  11 F820                   LD   DE,-7E0H    ; ADJUST TO LINE 1
024B'  19                        ADD  HL,DE
024C'  C3 01CC'                  JP   TOSTAD      ; BC=LINE COUNT
                                                  ; HL=OP ADDRESS
                          .S
                                 SUBTTL CLEAR SCREEN
                          ;
                          ;
024F'                   CLRSAD:                   ; HERE FROM BASIC ENTRY TO CLEAR SCREEN
024F'  ED 4B 090C'               LD   BC,(CLRCTL) ; GET CONTROL INFO.
                          ;
0253'                   CLRSCN:                   ; ENTRY WITH BC=LINE COUNT AND
                                                  ; STARTING LINE NO. FOR CLEAR
0253'  78                        LD   A,B
0254'  A7                        AND  A           ; TEST VALIDITY
0255'  CA 013F'                  JP   Z,INVPAR    ; ERROR IF COUNT=0
0258'  81                        ADD  A,C
0259'  FE 19                     CP   E           ;
025B'  DA 013F'                  JP   C,INVPAR    ; ERROR IF B<C+1
025E'  FE 1A                     CP   25
0260'  D2 013F'                  JP   NC,INVPAR   ; ERROR IF B+C>24
0263'  CD 026A'                  CALL CLRSCA      ; DO SERVICE
0266'  C3 0E3A'                  JP   SBCSRET     ; RETURN BC=0
                          ;
                          ;
0269'  C9 04FB'         CLRSCA   CALL LDATTR      ; GET ATTRIBUTE CTL
026C'  C5                        PUSH BC          ; ORIGINAL BC
026D'  3E 18                     LD   A,SCRSZ     ; CONVERT TO INTERNAL FORMAT
026F'  91                        SUB  C
0270'  47                        LD   B,A
0271'  3A 0092'                  LD   A,(LINLEN)
0274'  3C                        INC  A           ; SET TO COL.0
0275'  4F                        LD   C,A
0276'  CD 0479'                  CALL LNOU        ; GET ADDRESS
0279'  50                        LD   D,B
027A'  59                        LD   E,C         ; SAVE POSITION
027B'  C1                        POP  BC          ; ORIG. BC
027C'  D5                        PUSH DE          ; SAVE CURSOR POSITION
027D'  7D               CLRSC1   LD   A,L         ; GET A OF LINES THIS BLOCK
027E'  07                        RLCA
027F'  07                        RLCA
0280'  07                        RLCA
0281'  4F                        LD   C,A
0282'  3E 05                     LD   A,B
0284'  91                        SUB  C
0285'  B8                        CP   B           ; COMPARE TO TOTAL LINE COUNT
0286'  38 A7                     JR   C,CLRSC7
0288'  78               CLRSC2   LD   A,B         ; NO. OF LINES
0289'  06 00                     LD   B,0         ; REMAINING LINES
028B'  C5                        PUSH BC
028C'  47               CLRSC3   LD   B,A
028D'  0E 08                     LD   C,A         ; SCAN COUNT
028F'  78               CLRSC4   LD   A,B
0290'  C5                        PUSH BC          ; SAVE LINE CNT./SCAN CNT.
0291'  E6 07                     AND  7
0293'  0F                        RRCA
0294'  0F                        RRCA
0295'  0F                        RRCA
0296'  4F                        LD   C,A         ; BC=32*NO. OF LINES
0297'  06 00                     LD   B,0         ; (=0 IF 256 FOR 8 LINES)
0299'  0D                        DEC  C           ; ADJUST
029A'  E5               CLRBCE   PUSH HL
```

```
0298'  56                  564          LD    D,M
029C'  5E                  565          LD    E,L
029D'  36 00               566          LD    (HL),00          ; CLEAR DP
029F'  I3                  567          INC   DE
02AD'  ED B0               568          LDIR
02A2'  E5                  569          POP   HL
02A3'  24                  570          SNC   H                ; NEXT SCAN ROW
02A4'  C5                  575          POP   BC               ; POP LINE/SCAN COUNT
02A5'  DC                  572          DEC   C                ; DECR. SCAN COUNT
02A6'  20 E7               573          JR    NI,CLR5C4        ; NEXT SCAN
02A8'  E5                  574          PUSH  HL               ; SAVE NEXT DP ADRS.
02A9'  25                  575          DEC   H                ; LAST SCAN OF CHAR.
02AA'  C0 04ED'            576          CALL  CALCATT          ; GET ADRS. OF ATTRIBUTE FILE
02AD'  78                  577          LD    A,B              ; NO. OF LINES THIS BLOCK
02AE'  E6 07               578          AND   7
02B0'  0F                  576          RRCA
02B5'  0F                  580          RRCA
02B2'  0F                  58I          RRCA
02B3'  4F                  582          LD    C,A              ; BC=32*NO. OF LINES
02B4'  06 00               583          LD    B,0
02B6'  0B                  584          DEC   C                ; ADJUST
02B7'  54                  585          LD    D,H
02B8'  50                  586          LD    E,L
02B9'  3A 0039'            587          LD    A,(ATTBYT)       ; ATTRIBUTE VALUE
02BC'  77                  588          LD    (HL),A           ; UPDATE ATTRIBUTE FILE
02BD'  53                  589          INC   DE               ;
02BE'  ED B0               560          LDIR                   ;
02C0'  EI                  59I          POP   HL               ; DP ADRS.(LAST SCAN+I00H)
02C5'  CI                  562          POP   BC               ; TOTAL LINE COUNT
02C2'  78                  563          LD    A,B
02C3'  A7                  594          AND   A
02C4'  28 05               595          JR    I,CLRSX7
02C6'  2E 00               596          LD    L,0              ; HL=START OF NEXT BLOCK
02C8'  C3 02D0'            597          JP    CLASCS           ; B=REMAINING LSNE COUNT
02CB'  CI                  598  CLBSRT  POP   BC               ; CURSOR POSITION
02CC'  C3 0492'            599          JP    UPCPOSN          ; RETURN VIA UPDATE AND STORE POSITION
02CF'  4F                  600  CLBSCT  LD    C,A              ; SAVE NO. LINES THIS BLOCK
0200'  78                  605          LD    A,B
0201'  4I                  602          SUB   C                ; ADJUST TOTAL LINE COUNT
0202'  47                  603          LD    B,A
0203'  C5                  604          PUSH  BC               ; REMAINING LINE COUNT
0204'  76                  605          LD    A,C              ; LINES THII BLOCK
0205'  I8 B5               606          JR    CLR5C3           ;
                           607  ;
                           609          SUBTTL 'GET' ROUTINES
                           650  ;
                           6II  ;
0207'                      652  G7CH80I                        ; HERE FROM BASIC ENTRY TO GET CHARACTER
                           6I3                                 ; (RETURNS CODE FOR CHARACTER AT
                           6I4                                 ;  POSITION SPECIFIED IN GETCTL)
0207'  ED 48 00I6'         6I5          LD    BC,(GETCTL)      ; GET CONTROL INFO.
                           6I6  ;
0208'                      6I7  G7CHAB3                        ; ENTRY WITH POSITION IN BC
020B'  C0 0436'            6I8          CALL  TSTPAR           ; TEST PARAMETERS
020E'  3A 003E'            6I9          LD    A,(CPBIT)        ; SAVE CURRENT BIT POSITION
02EI'  F5                  620          PUSH  AP
02E2'  CD 044T'            62I          CALL  CONVPM           ; CONVERT TO INTERNAL FORMAT
02E5'  C0 045T'            622          CALL  CALCPOS          ; GET DP ADDRESS AND BIT POS.
02E8'  ED 5B 002P'         623          LD    DE,(CHTBL)       ; CHAR.TABLE
02EC'  06 6D               624          LD    B,66             ; NO. OF PRINTABLE CHARACTERS
02EE'  3E E0               625          LD    A,ECH            ; SET ADJUSTMENT INDEX
02P0'  32 003A'            626  GTCHI   LD    (GTINDX),A
02P3'  54                  627          INC   D                ; ADJUST TO START OF TABLE
02P4'  C5                  628  G7CH2   PUSH  BC               ; SAVE COUNT
02P5'  E5                  625          PUSH  HL               ; SAVE ADRS. IN DP
02P6'  D5                  630          PUSH  DE               ; SAVE ADRS. IN CHAR.TABLE
02P7'  C0 05I3'            63I          CALL  GTC2EC           ; GET SCAN LINE FROM DP TO REG.B
                           632                                 ; BITS 7-2
02P6'  7E                  633  GTCH22  LD    A,B
02P0'  E6 PC               636          AND   0FCH             ; 6 BITS TO A
02P0'  47                  635          LD    B,A              ; DP CHAR.SCAN ROW IN B
02PE'  EB                  636          EX    DE,HL            ; CHAR.SET PTR. TO HL
02PP'  7E                  637          LD    A,(HL)           ; PIXEL ROW FROM CHAR.SET
0300'  E6 PC               638          AND   0FCH             ; MASK TO 6 BITS
0302'  A0                  639          XOR   B                ; TEST AGAINS DP IMAGE
0303'  28 I0               640          JR    I,GTCH3          ; MATCH
0305'  F5                  64I          PUSH  AP
0306'  3A 0036'            642          LD    A,(GTINDX)       ; TEST IF LOOKING AT ITO.GRAPHICI
0309'  PE 90               6A3          CP    90H
030J'  20 03               644          JR    NZ,GTCH23
0330'  PI                  643          POP   AP
030E'  I8 30               646          JR    GTCH4            ; DO NOT TEST INVERSE IF STO.GR.
03I0'  PI                  647  GTCH23  POP   AP
03II'  PE PC               648          CP    0FCH             ; TEST IF INVERSE
03I3'  20 36               649          JR    NZ,GTCH4         ; DOES NOT MATCH
0355'  4F                  650  GTCH3   LD    C,A              ; C=0 FOR MATCH  -4 FOR INVERSE
03I6'  06 07               65I          LD    B,7              ; SCAN COUNT
03I8'  EB                  652  GTCH3I  EX    DE,HL            ; DP ADRS.TO HL
03I9'  C5                  653          PUSH  BC       ; SAVE MATCH AND SCAN COUNT
03IA'  24                  654          INC   H        ; NEXT SCAN IN DP
03IB'  I3                  655          INC   DE       ; NEXT ROW IN CHAR.SET
03IC'  CD 05I3'            656          CALL  GTC2BC           ; NEXT SCAN ROW TO BC
03IP'  78                  A57          LD    A,B
0320'  E6 PC               658          AND   0FCH
0322'  47                  6I6          LD    B,A
0323'  EB                  660          EX    DE,HL            ; CHAR.SET PTR. TO HL
0324'  7E                  66I          LD    A,(HL)           ; PIXEL ROW TO A
0329'  E6 PC               662          AND   0FCH
032T'  66                  663          ADD   B                ; TEST MATCH
0328'  CI                  66A          POP   BC               ; GET PREV.MATCH AND SCAN COUNT
0329'  A6                  665          OR    C
032A'  20 I*               666          JR    NI,GTCH4         ; NO MATCH-START AGAIN AT NEXT
032C'  50 EA               667          DJNI  GTCH3S           ; DO NEXT ICAN
032E'  79                  668          LD    A,C              ; A=-4 IF MATCH ON INVERSE
032P'  C6 03               669          ADD   A,3              ; IET TO -I IF MATCH ON INVERSE
033I'  CI                  670          POP   BC               ; HERE IF MATCH- (SP)=PTR.TO CHAR.
0332'  CI                  67I          POP   BC               ; IN TABLE:(SP+2)=CHAR.IN DP:
0333'  CI                  672          POP   BC               ; (SP+4)=CHAR.COUNT
0334'  4F                  673          LD    C,A              ; B=COUNT C=-3 IF MATCH ON INVERSE
0333'  3A 0036'            676          LD    A,(GTINDX)       ; CONVERT MATCH TO CHAR.CODE
0338'  90                  675          SUB   B
0339'  PE 20               676          CP    20H              ; TEST IF MATCH ON SPACE
033B'  20 05               677          JR    NZ,GTCH32        ; NO
033D'  0C                  678          INC   C                ; TEST IF MATCH ON INVERSE
033E'  20 02               679          JR    NI,GTCH32
```

```
0340'  3E 8F                   LD    A,8FH         ; LOAD CODE FOR GRAPHICS BLOCK
0342'  4F            GTCH12    LD    C,A           ; RETURN IN REG. C OF BC PAIR
0343'  06 00                   LD    B,0           ; AND IN REG. A
0345'  F1                      POP   AF
0346'  32 003E'                LD    (OPBIT),A     ; RESTORE OPBIT
0349'  79                      LD    A,C           ; CHAR.CODE TO A
034A'  C9                      RET
                     ;
034B'  E1            GTCH4     POP   HL            ; PTR. TO CHR. SET
034C'  11 0008                 LD    DE,8          ; MOVE ON TO NEXT CHARACTER
034F'  19                      ADD   HL,DE
0350'  5D                      LD    E,L
0351'  54                      LD    D,H
0352'  E1                      POP   HL            ; OP ADDRESS
0353'  C1                      POP   BC            ; CHAR. COUNT
0354'  10 90                   DJNZ  GTCH2         ; DECREMENT CHAR.COUNT AND LOOP AGAIN
                     ;
0356'  3A 003A'                LD    A,(GTINDX)    ; TEST IF DONE
0359'  FE 80                   CP    80H           ; TEST 1ST STD.CHR.SET
035B'  30 0A                   JR    NC,GTCH5      ; TRY GRAPHICS
035D'  ED 5B 0031'             LD    DE,(GRTBL)    ; GRAPHICS TABLE
0361'  06 10                   LD    B,16          ; NO. OF ENTRIES
0363'  3E 80                   LD    A,80H
0365'  18 89                   JR    GTCH1
0367'  FE 90         GTCH5     CP    90H           ; TEST IF STD.GRAPHICS
0369'  30 0C                   JR    NC,GTCH6      ;
036B'  ED 5B 5C7B              LD    DE,(UDG)      ; TRY USER-DEFINED GRAPHICS
036F'  15                      DEC   D             ; ADJUST ADDRESS-100H
0370'  06 15                   LD    B,21          ; NO. OF ENTRIES
0372'  3E A5                   LD    A,0A5H        ; INDEX ADJUST
0374'  C3 02F0'                JP    GTCH1
                     ;
                     ;                             ; NO MATCH AT ALL
0377'  F1            GTCH6     POP   AF            ; RESTORE ORIG. OPBIT
0378'  32 003E'                LD    (OPBIT),A
037B'  48                      LD    C,B           ; BC=0
037C'  AF                      XOR   A             ; A=0
037D'  C9                      RET

                                              ; GET ATTRIBUTE BYTE
                     ;
                     ;                        ; HERE FROM BASIC ENTRY
037E'  ED 4B 0019'   GETA80: LD    BC,(GETCTL)
                     ;                        ; HERE WITH LINE/COL. IN BC
0382'                GETATT:
0382'  CD 0436'                CALL  TSTPRM     ; TEST PARAMETERS
0385'  3A 003E'                LD    A,(OPBIT)  ; SAVE CURRENT BIT POSITION
0388'  F5                      PUSH  AF
0389'  CD 0847'                CALL  CONVRM     ; CONVERT TO INTERNAL FORMAT
038C'  CD 0857'                CALL  CALCPOS    ; CALCULATE OF POSITION
038F'  CD 0480'                CALL  CALCATT    ; CALCULATE ATTRIBUTE POSITION
0392'  7E                      LD    A,(HL)     ; ATTRIBUTE BYTE TO A
0393'  4F                      LD    C,A        ; PASS BACK IN BC
0394'  06 00                   LD    B,0
0396'  F1                      POP   AF         ; RESTORE (OPBIT)
0397'  32 003E'                LD    (OPBIT),A  ;
039A'  79                      LD    A,C        ; ATTRIBUTE ALSO IN A
039B'  C9                      RET
                     ;
                     ;
                     ;                        ; GET CURSOR POSITION
039C'                GETC80:                  ; HERE FROM BASIC ENTRY
039C'                GETCUR:
039C'  ED 4B 0034'             LD    BC,(CURPOS)     ; GET INTERNAL POSITION
03A0'  CD 0447'                CALL  CONTFM          ; CONVERT TO USER FORMAT
03A3'  3A 0033'                LD    A,(LINLEN)      ; TEST IF END OF LINE
03A6'  B9                      CP    C               ; IF =40 OR 80
03A7'  30 0B                   JR    NC,GETC1        ; NO
03A9'  0E 00                   LD    C,0             ; NEXT POSN.START OF NEXT LINE
03AB'  78                      LD    A,B
03AC'  3C                      INC   A               ; BUMP TO NEXT LINE
03AD'  47                      LD    B,A
03AE'  FE 18                   CP    24              ; TEST IF OFF SCREEN
03B0'  38 02                   JR    C,GETC1         ;    NO
03B2'  06 17                   LD    B,23            ; POSN. IS AT BOTTOM LINE
03B4'  ED 43 0007'   GETC1     LD    (LINCOL),BC     ; STORE FOR BASIC PROGRAM
03B8'  C9                      RET                   ; VALUES ALSO IN BC
                     ;
                               SUBTTL INITIALIZE 80-COL.MODE
                     ;
                     ;
                     ;
03B9'                INIT80:
03B9'  21 1701                 LD  HL,SCINIT          ; INITIALIZE VARIABLES
03BC'  22 0026'                LD  (SCRCTL),HL        ; SCROLL CONTROL
03BF'  3E 17                   LD  A,23               ;
03C1'  32 002D'                LD  (BOTLIN),A         ; BOTTOM LINE
03C4'  3E 01                   LD  A,1                ;
03C6'  32 002E'                LD  (SCRLCT),A         ; SCROLL COUNT
03C9'  3E 28                   LD  A,40               ;
03CB'  32 0031'                LD  (LINLEN),A         ; LINE LENGTH
03CE'  3E 01                   LD  A,1                ;
03D0'  32 0050'                LD  (MARGIN),A         ; MARGIN
03D3'  21 1800                 LD  HL,CLINIT          ;
03D6'  22 000C'                LD  (CLRCTL),HL        ; CLEAR SCREEN CONTROL
03D9'  AF                      XOR A
03DA'  32 0036'                LD   (STRGCT),A        ; STRING COUNT
03DD'  32 005C'                LD   (STRGCT+1),A      ;
03E0'  21 0829'                LD   HL,CHRSET         ; STD. 80-COL.CHAR.SET
03E3'  22 002F'                LD   (CHTBL),HL        ;
03E6'  21 0729'                LD   HL,GRPHST         ; STD. 40-COL.GRAPHICS SET
03E9'  22 0011'                LD   (GRTBL),HL        ;
                     ;                                ; INITIALIZE BASIC INPUTS
                     ;
03EC'  32 0000'                LD   (DATAB),A         ; DATA BYTE
03EF'  32 0007'                LD   (LINCOL),A        ; COLUMN
03F2'  32 0008'                LD   (LINCOL+1),A      ; LINE
03F5'  32 0019'                LD   (GETCTL),A        ; "GET" COLUMN
03F8'  32 001A'                LD   (GETCTL+1),A      ; "GET" LINE
                     ;
```

```
03F8'  21 0000              LD   HL,0                ;
03FB'  22 5CTD              LD   (COORDS),HL         ; PLOT POSITION
0401'  3A 0033'             LD   A,(LINLEN)          ;
0404'  3C                   INC  A                   ; HOME POSITION TO BC
0405'  4F                   LD   C,A
0406'  06 18                LD   B,SCRS1             ; = LINE 0/COL.0
0408'  CD 0452'             CALL UPDPOSN             ; CALCULATE AND STORE HOME POSITION
040B'  C3 015A'             JP   GOODRST             ; RETURN BC=0
                       ;
                       ;
                       ;
                            SUBTTL INTERNAL SUB-RTNS.
                       ;
                       ;
040E'  3A 5CC3     GETSTRG LD   A,(PARAM8)          ; GET STRING ID
0411'  E6 1F               AND  1FH                 ; MASK OFF UPPER BITS
0413'  F6 60               OR   60H                 ; STRING VARIABLE IDENTIFIER
0415'  57                  LD   D,A                 ; SAVE IN D
0416'  2A 5C4B             LD   HL,(VARS)           ; FIND STRING
0419'  7E          GTSTR1  LD   A,(HL)              ;
041A'  E6 7F               AND  07FH                ; TEST IF END OF VARS AREA
041C'  28 13               JR   Z,NOSTRG            ; NO FIND - RETURN BC=2
041E'  BA                  CP   D                   ; TEST IF MATCH
041F'  28 08               JR   Z,GTSTR2            ; FOUND STRING
0421'  D5                  PUSH DE                  ; SAVE STRING ID
0422'  CD 19B8             CALL NEXT ONE            ; RETURNS ADRS. OF NEXT VAR. IN DE
0425'  EB                  EX   DE,HL               ; ADRS. TO HL
0426'  D1                  POP  DE                  ; RESTORE STRING ID
0427'  18 F0               JR   GTSTR1              ; LOOK AGAIN
0429'  23          GTSTR2  INC  HL                  ; GET LENGTH
042A'  4E                  LD   C,(HL)
042B'  23                  INC  HL
042C'  46                  LD   B,(HL)
042D'  23                  INC  HL                  ; FIRST TEXT CODE
042E'  78                  LD   A,B                 ; TEST IF NULL STRING
042F'  B1                  OR   C
0430'  C9                  RET                      ; RETURN (Z IF NULL STRING/NZ IF NOT)
                       ;
0431'  0E 02       NOSTRG  LD    C,2
0433'  06 00               LD    B,0
0435'  C9                  RET                      ; (Z AND BC=2 FOR NO FIND)
                       ;
                       ;

                       ;
                       ;
                                                    ; TEST PARAMETER VALIDITY
                                                    ; IF NOT VALID, DISCARDS RETURN
                                                    ; AND EXITS VIA INVPAR
0436'  3E 17       TSTPAR  LD   A,23                ; TEST LINE >23
0438'  B8                  CP   B
0439'  30 04               JR   NC,TSTPR1
043B'  F1          PARSER  POP  AF
043C'  C3 013F'            JP   INVPAR
043F'  3A 0033'    TSTPR1  LD   A,(LINLEN)
0442'  3D                  DEC  A
0443'  B9                  CP   C          ; TEST COL. >LINE LENGTH-1
0444'  38 F5               JR   C,PARSER
0446'  C9                  RET
                       ;
                                                    ; CONVERT LINE/COL. FROM USER TO
                                                    ; INTERNAL FORMAT AND VICE VERSA
                                                    ; USER FORMAT: LINES 0-23
                                                    ;              COL. 0-39(41)
                                                    ; INTERNAL FORMAT:
                                                    ;              LINES 24-1
                                                    ;              COL. (43)41-2
                                                    ;              (COL.1 END OF LINE)
                       ;
                       ;
0447'  3A 0033'    CONVPM  LD    A,(LINLEN)
044A'  3C                  INC   A
044B'  91                  SUB   C
044C'  4F                  LD    C,A                ; CONVERT
044D'  3E 18               LD    A,SCRS1
044F'  90                  SUB   B
0450'  47                  LD    B,A
0451'  C9                  RET
                       ;
                       ;
                       ;
0452'              UPDPOSN:                         ; ENTER HERE WITH BC=LINE/COL.IN
                                                    ; INTERNAL FORMAT
0452'  CD 0457'            CALL  CALCPOS            ; CALCULATE POSITION
0455'  18 42               JR    STPOSN             ; STORE UPDATED POSITION AND RETURN
                       ;
                       ;                            ; ROUTINE TO CALCULATE POSITION IN DF
                       ;                            ; UPDATES BIT POSITION IN VARIABLE DFBIT
                       ;                            ; RETURNS DF ADDRESS IN HL.  PRESERVES
                       ;                            ; LINE/COL.POSITION IN BC
                       ;
0457'  CD 0A7E'    CALCPOS CALL  LNBU               ; GET DISPLAY FILE ADRS.FOR LINE
045A'  3A 0033'            LD    A,(LINLEN)
045D'  3C                  INC   A
045E'  91                  SUB   C                  ; GET COL. NO.
045F'  F5                  PUSH  AF                 ; SAVE COL. NO.(0-(LINLEN-1))
0460'  5F                  LD    E,A
0461'  87                  ADD   A,A
0462'  83                  ADD   A,E                ; E 3(40-COL.POS) /4 = 32-COL.POS
0463'  CB 3F               SRL   A
0465'  CB 3F               SRL   A
0467'  5F                  LD    E,A                ;
0468'  3A 005D'            LD    A,(MARGIN)         ; GET ADJUSTMENT VALUE
046B'  83                  ADD   A,E
046C'  5F                  LD    E,A
046D'  16 00               LD    D,0                ; DE CONTAINS OFFSET INTO LINE
046F'  19                  ADD   HL,DE              ; HL CONTAINS DF ADDRESS
0470'  1E 07               LD    E,7
0472'  F1                  POP   AF                 ; A=COL.POSN.FOR 40 COLS.
0473'  E6 03               AND   3                  ;
0475'  28 02               JR    Z,CALCP2           ; STARTS AT BIT 7
0477'  3F                  LD    E,A                ; STARTS AT BIT 1,3 OR 5
0478'  3D                  DEC   A
0479'  83          CALCP2  ADD   A,E
047A'  32 005E'            LD    (DFBIT),A
```

```
0470'  C0                      EET                        ; BC=POS. HL=CP 6DDRESS
                        ;
                        ;
0470'                   LNEU:                             ; GOT DISPLAY FILE ADDRESS
                                                          ; FOR START OF LINE IN B
0478'  30 SE                    LD   A,SCRS3
0A00'  0E                       SUE  E
0401'  S7                       LD   B,A
0403'  0P                       RECA
0403'  0P                       ORC6
0404'  0P                       RRCA
0485'  SE SE                    ANE  EE0H
048T'  EP                       LD   L,A
0408'  7A                       LD   A,B
04E6'  S6 S0                    ANE  IEH
04EE'  P6 40                    OE   AEH
04E3'  6T                       LD   H,A
04EE'  3A SCC2                  LE   A,(VSONOO)           ; TOET SP U3SNS DP1
0491'  C0 4T                    OST  E,6
0493'  CE                       OOT  3
0494'  SE 2E                    LD   6,20H                ; UESNS OPS
0A66'  E6                       OD   H
0467'  0T                       LD   H,6
0A60'  C6                       RET
                        
0406'                   DTPOSN:                           ; STEOO CURSOO POSSTSON
0499'  EE 43 0E3A'              LD   (CURPOS),BC
049D'  22 EE36'                 LE   (OPAOPE),HL
04AE'  C9                       OST
                        ;
04A1'                   LDPOSN:                           ; LOAC CURSOR POSITION
04A1'  S0 4E EE34'              LD   BC,(CUPPOE)
04AE'  SA EE36'                 LD   HL,(OPADRS)
04A0'  C9                       RET
                        ;
                        ;
04A9'                   UPOATT:                       ; UPOATE ATTRIBUTE ETTE FOR CHARACTER
                        ;                             ; JUST POINTED
                        ;                             ; HL = ANT ICAN OF CHAR.IN OP
                        ;                             ;
                        ;                             ; SP CHAPACTEP EPPNI ATTRIRUTE ETTE
                        ;                             ; EOUNDART, EOTH ATTRIRUTE OTTES WILL
                        ;                             ; OE ADJUSTED
                        ;
                        ;
EAA6'  C0 E450'                 CALL CALCATT              ; ADDRE. OP ATIRIEUTR RTTE TO HL
E4AC'  3A 0E3E'                 LD   A,(OPOIT)            ; EAVE OIT PCEITION
E4AP'  PE                       PUSH AP                   ;
04E6'  3A EE3E'                 LD   A,(ATTETT)           ; CURRENT ATTRIEUTE ETTE VALUR
E4E3'  EP                       LD   E,A                  ; TO E
04E6'  3A EE3P'                 LD   A,(ATTNEP)           ; MPIK ID D
E4E7'  S7                       LD   D,A
E4B0'  3A 0030'                 LD   A,(MASKR)            ; PLPOS
04B0'  4T                       LE   B,A                  ; TJ B
E4BC'  TE               UPDATE  LD   P,(HL)               ; OTTE PRON ATTRIEUTE PILE
EABD'  A0                       ROR  E
E4BE'  A2                       PND  D
E4BP'  A0                       ROR  E                    ; NEW ATTR.PCR BI OLD POR S
04CE'  CE 6E                    EST  A,E                  ; SET INK COMPLRMRNT OP PPPRPT
04C2'  20 0E                    JR   Z,UPDPTS             ;
04C4'  E6 P0                    AND  0P0H                 ; SET INP TO RLPCK
E4C6'  CR 4P                    OIT  E,A                  ; IS PAPER 4-T
E4C9'  20 R2                    JR   NI,UPDPTS            ; TES - USE RLACP INP
04CA'  P6 0T                    DR   T                    ; NO  - USE WHITE INP
E4CC'  C0 T0            UPDATS  OIT  6,F                  ; SRT PPPRR COMPLEMENT OP INKT
04CE'  20 ER                    JR   Z,UPOATZ             ;
E4DE'  E6 CT                    AND  RCTH                 ; SET PAPRR TO OLACK
E4B2'  C0 ET                    OIT  2,P                  ; IS INK 4-T
E4D4'  2E E2                    JR   NZ,UPOATZ            ; TRE - USR RLACP PPPEP
E4D6'  P6 EE                    OR   ERH                  ; NO  - USR WHITE PAPRR
040E'  TT               UPOATZ  LD   (HL),A               ; TO PTTRIRUTR PILE
E409'  3A 00EE'                 LD   A,(DPOIT)            ; TRST EIT POSITION
E4DC'  PE E9                    CP   E
040E'  30 00                    J0   NC,UPDATE            ; CNPR.IN SINGLE RTTE IP STPPTS
                        ;                                 ; IN OIT T OR S
0400'  23                       INC  HL                   ; CHAR.SPANS 2 RTTOS IP STARTS
E40S'  3E ET                    LD   A,T                  ; IN OIT 3 CP S
E4EE'  E2 EE3E'                 LD   (OPOIT),A            ; TEMPORAPILT SET TO T
E4E6'  S0 D4                    JR   UPDPTE               ; UPDPTE NEXT OTTE
                                                          ; (OPRIT=T WILL PORCE ERIT PND
                                                          ;  ORIGINAL VALUE OP OPEIT WILL
                                                          ;  OO RESTORED)
                        ;
E4E0'  PS               UPOATE  POP  AP                   ; ORIGINPL DPOIT
04E9'  E2 003S'                 LD   (OPOIT),A            ; RESTORE
E4EC'  C9                       RET
                        ;
                        
                        ;                                 ; SUB-RTN. TO CALCULATE ADR3. OP
                        ;                                 ; ATTRIOUTE OTTE PROM POPS. OP
                        ;                                 ; ANT 3CPN POW IN DP
                        ;
                        ;                                 ; USEO ET UPOATT AND OETATT
                        ;
04E0'  7C               CALCATT LD   A,H                  ; UPPER OTTE EP ADDRE3E
E40E'  0P                       RRCA
E6EP'  EP                       PRCA
0APE'  EP                       RRCA
E4PS'  EE EE                    ANE  EEH
04P3'  P6 E0                    OR   SEH                  ; ADR3. OP PTTRIBUTE OTTE
04PS'  C6 AC                    OIT  E,H                  ; TEST WHICH OP
R4PT'  2R 02                    JR   Z,CPLCPI             ; OP1
04PE'  P6 2E                    OR   20H                  ; OP2
04PB'  ET               CALCAI  LE   H,A                  ; ADR3. OP ATTRIOUTE
E4PC'  C9                       RET
                        ;
04PD'                   LEATTR:                           ; LOGO INTERNAL ATTRIRUTE VARIAOLEE
                                                          ; POON SYSTEM VAPIARLEE
                                                          ;
E4PE'  PE                       PUEH AP                   ; EAVE A
04PE'  3A SC60                  LD   A,(ATTR_P)           ;
0SE1'  32 E039'                 LD   (ATTRTT),A           ;
0SE4'  3A ECEE                  LE   A,(PP3K_P)           ;
030T'  32 E03P'                 LD   (ATTMSK),A           ;
0E0A'  3A ECEI                  LD   A,(P_PLAG)
0S0D'  EP                       RRC6                      ; SHIPT ODE OITS TO ETEN
```

```
050E'  32 003R'          1023          LO    (MASKR),A           1
0511'  *1                1024          POP   AP
0512'  C9                1025          RET
                         1026   1
                         1027   1                                 1 RTN. USEO ET GTCHAR TO PUT
                         102R   1                                 1 6 PIRELS PROM OP 1M R PEG.BITS 7-2
                         1029   1                                 1 POR MATCHING AGAINST CHAR.SET
                         1030   1
0513'  P5                1031   GTC2EC  PUSH  HL                  1 SAVE OP AORS.
0514'  4A                1032          LO    R,(HL)              1 SCAN LINE PROM 0*
0515'  23                1033          INC   HL                  1 NEXT BYTE
0E16'  4E                103A   GTC2E1  LO    C,(HL)              1 BC CONTAINS CHAR.AT OPEIT
0E17'  3A 003E'          1035          LO    A,(OPBIT)
051A'  D6 07             1036          SUE   7
051C'  EC 44             1037          NEG                       1 A=NO.OP EITS OPPSET IN RC
051E'  26 07             103E          JR    2,GTC2E2            1 STARTS AT RIT 7
0520'  CE 11             1039   GTCSHPT RL    C
0522'  CR 10             1040          RL    E                   1 SHIPT 6 PIRELS POR CHARACTER
0524'  30                1041          OEC   A                   1 TO E, EITS 7-2
0525'  20 P9             1042          JR    NI,GTCSHPT
0527'  E1                1043   GTC2E2  POP   HL                  1
052E'  C9                1044          RET                       1 CHAR.IN E, EITS 7-2
                         1045   1
                         1046
                         10A7          INCLUOE CHSTRO
                         104E          SUETTL CHAR.SET POR 40/E0 COL.MOOES
                         1049
                         1050   1
                         1051   1 Get patterns for characters on the TV screen!
                         1052   1       character with code x is at offsets 8x (top line) to 8x+7 (bottom line)
                         1053   1       0 = white, 1 = black, no margins between character spaces either
                         1054   1               vertically or horizontally
                         1055
0529'  00                105A   CHRST   defb 00000000b            1 code 20      space
052A'  00                1057          defb 00000000b
052B'  00                105E          defb 00000000b
052C'  00                1059          defb 00000000b
052D'  00                10A0          defb 00000000b
052E'  00                1061          defb 00000000b
052F'  00                1062          defb 00000000b
```

See APPENDIX C-2 for Character Set

```
AOL - ASC 003 40-COL.MODE SUPPORT        CP1R0/11 version 13.36.14       1A-Mar-R4  12:43:23
CHAR.SET FOR 40/80 COL.MODES    CHST80.SRC

*        Reserved     ATTSYT   0039         ATTHSK   C03P         ATTRSP   5CR0         E        Reserved
EOTLN    0020         C        Reserved     CALCAT   04E0         CALCA1   04PR         CALC*0   045T
CALCP2   04T9         CHRSET   0429         CHRST    0529         CHTRL    002P         CLINIT   1900
CLRCTL   000C         CLRSRO   024P         CLPSC8   000E IN      CLPSCN   0253 IN      CLPSCO   0269
CLRSC1   0270         CLRSC2   02E8         CLRSC3   02RC         CLRSCA   02RF         CLRSC5   029A
CLRSCT   02CP         CLPSKT   02CR         CONVPM   D44T         COOROS   5C70         CURPOS   003A
O        Reserved     OATAR    0000         OPAORS   0336         OPRIT    003E         E        Reserved
ERRRET   013C         GETABO   03TE         GETATE   C01E IN      GETATT   03E2 IN      GETCEO   039C
GETCTL   0019         GETCUB   0021 IN      GETCUR   C39C IN      GETC1    0384         GETSTR   0A0E
GOOORE   013A         GRRLK    0224         GRPHST   0T29         GRPST    0R29         GRTBL    0031
GTCHAR   020R IN      GTCHBO   020T         GTCHRB   001R IN      GTCH1    02P0         GTCH2    02P4
GTCH22   02PA         GTCH23   0310         GTCH3    C315         GTCH31   0318         GTCH32   0342
GTCHA    0348         GTCH5    036T         GTCH6    03TT         GTCSHP   0520         GTC2EC   0513
GTC2E1   0516         GTC2E2   0527         GT1NOX   003A         GTSTR1   0419         GTSTR2   0429
H        Reserved     INIT4R   0025 IN      INIT40   0389 IN      INSOEL   010A         INVPAR   013P
L        Reserved     LOATTR   04P0         LO*OSN   04A1         LINCOL   000T         LINLEN   0033
LNBU     0A7E         LOOP     010E         LOOPO    01E1         LOOP1    0189         M        Reserved
MARGIN   0030         MAS*R    003E         MASKSP   5CRE         MERT1    00EE         MEXT2    0086
MERT3    0136         MOINVE   0113         MORR1    03C5         MORR2    00E0         MORR3    0100
MOSTRG   0431         MOXOR    00CR         PARAMP   5CC3         PARERR   0438         PSW      Reserved
PSFLAG   5C91         RECLEN   1720         SCINIT   1701         SCRCTL   002R         SCRLE    002A IN
SCRLEO   01A2         SCRLCT   002E         SCRLKT   C201         SCRLO    01RC         SCRCLL   01A6 IN
SCRS1    0018         SETCBO   0192         SETCUB   0009 IN      SETCUR   0196 IN      SP       Reserved
SPAREO   0011         SPARE1   0013         SPARE2   0015         SPARE3   001T         SPAPE4   002A
STELR    022C         STELRO   0230         STELK1   0231         STPOSN   049E         STRGCT   003E
TESTAO   01CC         TSTPAR   0436         TSTPR1   043P         TVPULO   0143         TVPUL1   015P
UOG      5CT8         UPOATT   04A9         UPOATO   C4RC         UPOAT1   04CC         UPOAT2   0408
UPOAT3   04E8         UOOPOS   0452         VORS     5C48         V1OMOO   5CC2         WRCHAR   0043 IN
WRCHRE   0001 IN      WRCHRT   013T         WRCHO    0040         WRCH01   0046         WRCH11   0065
WRCH12   00A0         WRCH13   00T1         WRCH1A   00ET         WRCH15   00EE         WRCH2    00EE
WRCH3    0099         WRCH5    009P         WRCH6    03AE         WRCHT    0120         WRSTLP   0170
WRSTRE   0004 IN      WRSTRG   0160 IN      WRSTRO   0169         WRSTR3   01TA         WRSTK1   0183

     No errors detected
```

```
Cross reference listing (XREF version 4.7)

Symbol     Refs (# = definition  R = write  <blank> = read)

ATTETT     111P     SET      955      101E#
ATTHSR     119*     PST     1020R
ATTR_P      4E*    1017
ROTLN       95*     296      T67R
CALCA1    100R     1010*
CALCAT     459      576      730      952     1001P
CALCP2     902      905*
CALC*0     622      T29      ET5      E83*
CHRSET      40*     103      TT9
CHRST       40     1056*
CHTEL     103*      152      623      7E0#
CLINIT      39*     TT4
CLRCTL      73*     511      T75R
CLRSEO      T6      510*
CLRSCO     AT1      523      52TP
CLRSC1     540*     597
CLRSC2     549*
CLRSC3     552*     606
CLRSCA     554*     573
CLRSC5     565A
CLRSCT     54E      600*
```

CLR3C8	31	T6e					
CLR5CH	28	9130					
CLR5X7	595	5980					
CONVPR	368	821	728	766	8828		
COORDS	470	7928					
CURPD3	1066	743	9336	938			
D6768	666	125	T898				
DP6DR3	1088	9346	939				
DP817	1180	201	317	246	278	2808	2838
		619	6848	T148	T26	T358	9088
		953	978	9848	9918	1035	
BRRR8T	287R	291	308				
G8T690	86	T22e					
G8T678	32	846					
G8T677	29	7246					
G8TC1	747	T53	T598				
G87C80	89	7416					
G8TCTL	816	615	722	7888	T858		
G8TCU8	32	856					
G87CUR	29	T42R					
G8T37R	319	8066					
GDDDR8	2866	379	392	926	799		
GR8LK	419	4736					
GR6H9T	416	104	783				
GRP3T	43	1020e					
GR78L	1046	149	709	7828			
G7C281	10346						
GTC282	1038	10436					
GTC28C	633	656	10336				
GTCM3	6266	703	718				
G7CM2	6286	689					
GTCM82	6338						
G7CM23	664	6676					
G7CM3	648	6506					
GTCM31	6526	667					
G7CM32	677	679	6816				
G7CM4	646	649	666	6886			
G7CM5	689	7046					
G7CM6	709	7136					
GTCMAR	28	6176					
GTCM8D	83	6126					
GTCMR8	32	836					
G7C9MP	30386	1042					
G71MDR	1116	6266	642	674	697		
G797R1	8116	926					
GT37R2	919	5216					
INI746	34	87	7636				
IN1749	34	876					
IN3D8L	4166						
INVPAR	137	139	2906	389	388	398	937
		529	522	843			
LO677R	128	327	927	10136			
LOPD9N	132	9376					
LINCOL	696	364	7998	7868	7878		
LINLEN	1036	164	398	532	745	7716	793
		866	862	884			
LN9U	401	535	883	9106			
LOOP	4198	479					
LOOP8	4236	438					
LOOP1	4286						
M6RGTN	117e	7736	396				
M6SA8	1086	178	939	10236			
M69R_P	496	1016					
M8X71	2076	210					
M8X72	2316	236					
M8X73	2326	255					
MD1MV8	245	2626					
MORR1	205	2116					
MORR2	231	2376					
MORR3	230	2566					
MO37RG	813	8106					
MORDR	200	2176					
P6R6M8	326	806					
P6R88R	8446	869					
P_PL6G	906	1021					
R8CL8N	436	817					
SC1NI7	386	764					
3CRCTL	886	510	376	7696			
3CRL0	312	391	3946				
3C0L8	31	816					
SCRL80	91	3766					
SCRLC7	1006	300	3048	3099	7696		
SCRLR7	442	4496	900				
SCROLL	28	3806					
SCR5I	376	293	383	929	796	888	912
S87C80	72	3626					
387CU8	31	726					
S87CUR	28	3666					
SPARR0	776						
SP6RR1	786						
3P6RR2	796						
3P6RR3	806						
SP6RR4	866						
ST8LK	407	4816					
ST8LK0	4846	496					
ST8LK1	4856						
S7P03N	289	876	326				
STRGC7	1156	1496	7776	7786			
T83760	4036	444	303				

```
TSTPAR    367     618     725     8410
TSTPR1    R63     R46e
TVPUL1    302     309e
TVPULQ    173     293e
UOG        46e    145     706
UPOGT0    961e    985
UPOAT1    966     969     9713
UPOAT2    972     975     977e
UPOAT3    980     990e
UPOAT7    2F4     943e
UPOP05    29R     369     599     797     873e
V6RS       45e    R17
VIOMOO     51e    926
WRCH0      87     123e
WRCH01    132e    331
WRCH11    164     149e
WRCH12    142     152e
WRCH13    148     151     153e
WRCH16    165     1T0e
WRCH15    169     171e
WRCH2     176e
WRCH3     1R1     1R3e
WRCH5     18Re    272
WRCH6     196e
WRCH7     2T3S
WRCHAR     2R     12Te
WRCHRR     31      6Te
WRCHXT    281     285e
WRSTLP    32Rs    342
WRSTR0     6R     319e
WRSTR3    336e    347
WRSTRR     31      6Re
WRSTRG     28     32Te
WRSTR1    334     345s
```

APPENDIX C-4

DUAL SCREEN MODE

Name: Dual Screen Mode Support

Description:

This component provides support to the application programmer for using the dual screen capability of the TS 2068. The services include opening/closing the second display file (moving the machine stack, OS ROM routines and BASIC structures), position control, clear screen and scroll screen services, and display of characters. In addition, services are provided to control which display file is active at the screen and which is the target of the current screen operation, as well as a Copy Service and an Exchange Service to transfer screen data/attributes within or between the two display files. "Get" services are provided to return the current display position, the character code for a specified display position, or the attribute byte for a specified display position.

For ease of use from BASIC, status is returned in the BC register pair, usually zero for successful completion and designated non-zero values for other conditions. The interface from BASIC is by means of POKE'ing the input parameters to designated ROM locations prior to invoking the service via the USR function. See page 4 of the code listing for the memory addresses to be used from BASIC with the support code loaded starting at Chunk 7 (E000H/57344).

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

- -

Assembler Services

- -

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

Input:

MODE	0	=	Normal (Display File 1 only)
	128(80H)	=	DF1 Active (DF2 Open)
	1	=	DF2 Active
	4	=	Screen Operations to DF1
	5	=	Screen Operations to DF2

From Machine Code: Mode Parameter in A

From BASIC: Mode Parameter in VIMODE

Description:

Mode values of 0, 128 or 1 will cause an update of the video mode hardware and the opening or closing of the second display file as needed. This involves determining if there is enough free RAM to open the second display file and if so, moving the BASIC structures and machine language variables area up and the UDG (User-Defined Graphics) area down to make space for the machine stack and DS RAM routines at the top of memory. The second display file is cleared to zeroes. The affected system and internal variables are updated or initialized (see Usage Section). When returning to Mode 0 the structures are returned to their normal locations. In these modes the designated Display File is active at the screen and is the target of all screen operations.

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

The screen position is set to "home" (Line 0/Col.0) whenever SETMODE (SETMD3) is executed.

NOTE: All screen operations of the TS 2068 System BASIC interpreter including program entry, system messages, LIST, PRINT, PLOT, DRAW, etc. work only in Display File 1.

- -

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

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

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

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

Description:

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

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

- -

Name: SETCUR (SETCUE from BASIC - parameters to LINCOL)

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

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

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

Description:

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

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

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

- -

- -

Name: WRCHR (WPCHR from BASIC - parameter to DATAB)

Input: Character code for character to be displayed

20H TO 7FH - Std. TS2068 Character Set
80H TO 8FH - Std. Graphics Set
90H TO A4H - User-Defined Graphics Set

From Machine Code: Register A

From BASIC: in DATAB

Description:

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

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

- -

Name: WRSTRG (WRSTRB from BASIC - String identifier in PARAMS)

Input: Character Code String

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

From BASIC: String Variable identifier in
System Variable PARAMS - 23747 (5CC3H)

Description:

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

NOTE: Characters within the string which are outside of the supported range (32 through 164 (20H-A4H)) will be ignored. E.g., BASIC Token codes and control codes embedded in an INPUT string will not be displayed or decoded.

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

```
0005  LET a$="-----string-------"
0010  POKE 23747,CODE "a"
0015  IF USR (WRSTRB)<>0 THEN -----
          (continue)
```

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

- -

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

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

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

Description:

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

Note: See Usage Section on 'automatic' scrolling.

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

- -

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

Input: Line/Column position as for SETCUR

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

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

Description:

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

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

Output: BC = 0 for no find
BC = 1 invalid parameters
BC = character code (20h-A4H)

- -

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

Input: As for GTCHAR

Description:

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

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

- -

Name: GETCUR (GETCUB from BASIC)

Input: None

Description:

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

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

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

NOTE: If the last character was printed at Col.31 of Line 23 (last position on the screen), then Col.0/Line 23 will be returned.

- -

Name: COPYSC (COPYSB from BASIC - string identifier in PARAMS)

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

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

From BASIC: Parameters in string variable, separated by commas.
String Identifier in System Variable PARAMS - 23747 (5CC3H)

Description:

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

From BASIC, the following example would copy lines 8 thru 20 from DF1 to lines 0 thru 12 in DF2:

```
LET a$="8,0,1,2,13"        8=Starting Line
                           0=Destination Line
                           1=Source DF
                           2=Dest.DF
                          13=Line Count

POKE 23747,CODE "a"     (String Identifier to PARAMS)
LET cc=USR (COPYSB)
IF cc<>0 THEN.............
```

Output: BC=0 Successful
BC=1 Invalid Parameters (DF<>1 or 2; Line No.<>0-23; Line+Count<>1-24
BC=2 BASIC - String Not Found

- -

Name: EXCHSC (EXCHSB from BASIC)

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

Free Machine Code: Source '1' DF in B
Source '1' Line in C
Source '2' DF in D
Source '2' Line in E
Line Count in A

From BASIC: Parameters in string variable, separated by commas, in the order Source '1' Line, Source '2' Line, Source '1' DF, Source '2' DF, Line Count. String Identifier in System Variable PARAMS - 23747 (5CC3H)

Description:

Exchanges the specified number of lines between Source '1' and Source '2'. Both Sources may be in the same Display File. The exchange is done by XOR'ing the two sources together three times. The operation is done a line at a time, accessing the two sources in a top to bottom fashion; e.g. to exchange lines 8-12 in DF1 with lines 16-20 in DF2 would first exchange line 8 (DF1) with line 16 (DF2), then line 9 with line 17 and proceed in this fashion through exchange of line 12 with line 20.

From BASIC, the following example would exchange lines 8 thru 4 in DF1 with lines 8-12 in DF2:

```
LET a$="0,8,1,2,5"         0=Source '1' Line
                           8=Source '2' Line
                           1=Source '1' DF
                           2=Source '2' DF
                           5=Line Count

POKE 23747,CODE "a"     (String Identifier to PARAMS)
LET cc=USR (EXCHSB)
IF cc<>0 THEN .............
```

Output: BC=0 Successful
BC=1 Invalid Parameters (DF<>1 or 2; Line No.<>0-23; Line+Count <>1-24
BC=2 BASIC - String Not Found

- -

- -

Usage:

Memory Usage:

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

Name	Size	Description
SCRCTL	2	Scroll Control LSB = Starting line number MSB = Number of lines to be scrolled
BOTLN	1	Bottom Line - Line number (0-23) after which test for scroll will be made.
SCRLCT	1	Scroll Count- Number of times + 1 that automatic scroll will be done. When decremented to zero, data will not be displayed and a condition code will be returned to user.
CHTBL	2	Character Table (Base Address-100H)
GRTBL	2	Std.Graphics Character Table (Base-100H)
LINLEN	1	Line Length
CURPOS	2	Current Position (Internal Format) LSB = Column Position MSB = Line Position
DFADDR	2	Current Display File Address
MASKB	1	Working Byte - (P FLAG Shifted Right 1) (bit 0 = OVER) (bit 2 = INVERSE) (bit 4 = INK Complement of PAPER (bit 6 = PAPER Complement of INK
ATTBYT	1	Working Byte - (Copy of ATTR P)
GTINDX	1	'Get' Index - Used by GTCHAR
STRGCT	2	String Count - Contains remaining byte count when BC=3 (Screen Full) is returned from the Write String service WRSTRG (WRST02).
ATTMSK	1	Working Byte - (Copy of MASK P)

Initial values set via SETMODE (SETMOB) when the second display file is first opened are as follows:

Variable Name	Value
SCRCTL	1701H
BOTLN	17H
SCRLCT	1H
CHTBL	3D00H
GRTBL	(Internal to Module)
LINLEN	20H
CURPOS	1821H
DFADDR	4000H/6000H
GTINDX	80H
STRGCT	0H

The following are the variables used for passing parameters in BASIC. The * indicates those initialized by SETMOB when the second display file is first opened:

Variable Name	Size	Value
* DATAB	1	0H
* LINCOL	2	0H
* CLRCTL	2	1800H
* GETCTL	2	0H
Y1MODE	1	

In addition, Y1DMOD, PARAMS and other system variables must be available to these routines. At minimum, chunks 0-3 and chunk 7 must be enabled in the Home Bank.

Location:

This package can be incorporated into a machine code program within the memory range from chunk 4 through 7, taking into consideration the remapping of certain structures when the second display file is open

NOTE: Machine code above RAMTOP is not moved.

Registers:

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

Automatic Scrolling:

As initialized, test for scrolling will be made when the print position goes to the next line following the bottom line on screen (BOTLN=23=24th line). Condition Code 3 (Screen Full) will be returned since SCRLCT will decrement to zero. If SCRLCT is set to some larger value, then the parameters in SCRCTL will be used to automatically scroll the designated number of lines beginning at the specified line number. As initialized, this will scroll the entire screen up. By performing a POKE or setting the variables BOTLN and SCRCTL to the desired values, automatic scrolling can be done using smaller sections of the screen. When working from BASIC it is recommended that BOTLN be set to Line 21 (15H) and SCRCTL+1 be set to 21 (15H) to avoid conflict with the Edit Line which uses the bottom two lines of the screen. Note that once SCRLCT expires, it will be set to 1. If a different value is desired, it must be reinitialized by the user after receiving the "Screen Full" condition.

By setting the SCRCTL variable and invoking the SCROLL (SCRL8) routine any portion of the screen may be scrolled at any time.

Attribute Control:

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

NAME	ADDRESS	CONTENTS
ATTR_P	23693	Bit 7 - FLASH 6 - BRIGHT 5 4 - PAPER 3/ 2 1 - INK 0/
MASK_P	23694	SAME FORMAT AS ATTR_P, USED FOR "TRANSPARENT" DISPLAY: For each bit that is set to 1, the corresponding information will be taken from the current screen position instead of from ATTR_P.
P_FLAG	23697	BIT 7 PAPER=COMPLEMENT OF INK 5 INK=COMPLEMENT OF PAPER 3 - INVERSE 1 - OVER (New Characters XOR'd with Old)

Using System ROM Screen Services:

The following technique can be used to build screens using the BASIC commands such as PRINT, PLOT, CIRCLE, and DRAW or other System ROM routines and get them into the second display file:

1. Set Mode 80 (opens 2nd DF)
2. Build Screen in DF1 using System or ADL Services
3. Copy DF1 to DF2 using COPYSC (COPYSB)
4. Set Mode 1 (DF2 to Screen)
5. Set Mode 4 (DF2 at Screen/Operations to DF1)
6. Build Screen in DF1 using System or ADL Services
7. Set Mode 80 (DF1 to Screen)
8. Go to Step 3

The switching of the screen from one display file to the other is transparent to the viewer where the files contain the same data.

W A R N I N G

When the second display file is active at the screen, any system messages such as SCROLL?, error reports, or use of the INPUT command will not be visible and the system may appear to be "hung". Indiscriminately pressing keys may be filling the 'invisible' Edit Line with 'garbage' thus further aggravating the situation. The ON ERR facility can be used to intercept some of these situations and set the views mode to use Display File 1 at the screen in order that the message can be seen and responded to. Otherwise, it is necessary to key in and execute from the Edit Line without being able to see it.

- -

```
ADL - ASC 004 DUAL SCREEN MODE SUPPORT  C8260/II version 10.36.14        16-Mar-84  12:47:30
VERSION LEVEL CONTROL    ADDI.SRC

                            NAME ADL - ASC 004 DUAL SCREEN MODE SUPPORT
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;
                      ;                 ************************************
                      ;                 *                                  *
                      ;                 *              TIMEX               *
                      ;                 * APPLICATION DEVELOPMENT LIBRARY  *
                      ;                 *                                  *
                      ;                 * NAME: DUAL SCREEN MODE SUPPORT   *
                      ;                 *                                  *
                      ;                 * ASC NO:  004                     *
                      ;                 * VERSION: 001                     *
                      ;                 * AUTHOR:  C. CORCORAN            *
                      ;                 * DATE:    1/06/84                 *
                      ;                 ************************************
                            SUBTTL VERSION LEVEL CONTROL
                      ;
                      ;
                      ;     VERSION          DATE              COMMENTS
                      ;     -------          ----              --------
                      ;
                      ;       001            1/8/84            ORIGINAL
                      ;
                            SUBTTL DEFINITIONS
                      ;
                      ;
                      ;
                      ;                          ****DEFINITIONS****
                      ;
                            INTERN  WRCHAR,WRSTRG,SETCUR,CLRSCN,SCROLL
                            INTERN  GTCHAR,GETATT,GETCUR
                            INTERN  WRCHRB,WRSTRB,SETCUB,CLRSCB,SCRLB
                            INTERN  GTCHRB,GETATB,GETCUB
                            INTERN  SETMODE,SETMDB
                            INTERN  COPYSC,COPYSB
                            INTERN  EXCHSC,EXCHSB   .
                      ;
=0018                 SCRLL          EQU     24        ; 24 LINES
=1701                 SCINIT         EQU     1701H     ; SCROLL CONTROL INITIALIZATION
=1800                 CLINIT         EQU     1800H     ; CLEAR SCREEN CTL. INIT.
=1C00                 CHRSET         EQU     1C00H     ; ROM CHAR.TABLE-100H
=01E0'                GRPHIT         EQU     ((GRPIT)-100H) ;TO.GRAPHICS CHARS.
                      ;
=0E8E                 CHNGVID        EQU     0E8EH     ; ROUTINE IN ROM EXTENSION
=1720                 RECLEN         EQU     1720H     ; ROUTINE IN HOME ROM
=30F9                 ININT          EQU     30F9H     ; ROUTINE IN HOME ROM
=3193                 FP2A           EQU     3193H     ; ROUTINE IN HOME ROM
                      ;
=00FF                 HREXPT         EQU     0FFH      ; HOME ROM EXT.SELECT PORT (BIT 7)
=00F4                 DKHSPT         EQU     0F4H      ; DOCK HORIZONTAL SELECT PORT
                      ;
=5C48                 VARS           EQU     5C4BH     ; SYSTEM VARIABLES
=5C50                 CH_ADD         EQU     5C5DH     ;
=5C65                 STKEND         EQU     5C65H     ;
=5C7B                 UDG            EQU     5C7BH     ;
=5C7D                 COORDS         EQU     5C7DH     ;
=5C8D                 ATTR_P         EQU     5C8DH     ;
=5C8E                 MASK_P         EQU     5C8EH     ;
=5C91                 P_FLAG         EQU     5C91H     ;
=5CB2                 RAMTOP         EQU     5CB2H     ;
=5CB4                 PRAMT          EQU     5CB4H     ;
=5CC2                 VIDMOD         EQU     5CC2H     ;
=5CC3                 PARAMS         EQU     5CC3H     ; (LOCATION 23747)
=6840                 DRIVES         EQU     6840H     ;
=12C0                 INSERTSZ       EQU     7B00H-DRIVES
=0840                 MOVESZ         EQU     DRIVES-6000H
=F7C0                 DEST7          EQU     0FFFFH-MOVESZ+1
=97C0                 PIX            EQU     DEST7-6000H

=1000                 PIXTBL         EQU     1000H
```

```
                                82          SUBTTL INPUTS AND ENTRIES
                                83  ;
                                84  ;
                                85  ;
                                8A  ;
                                87  ;
                                88  ;
                                89  ;               **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                                90  ;                                                                  WITH
                                91  ;                                                                MODULE LOADED
                                92  ;                                                                AT 37344(E000H)
                                93  ;                                                                DECIMAL ADDRESS
                                94  ;                                                                ---------------
                                95  ;
0000' 20                        9A  DATA0           DEFB    20H     ; DATA BYTE FROM BASIC          37344
0001' CI 004A'                  97  WRCHR0:         JP      WRCHO   ; ENTRY TO WRITE CHARACTER      37348
0004' C3 0088'                  98  WRSTR0:         JP      WRSTRO  ; ENTRY TO WRITE STRING         3734A
                                99  ;                               ; (see PARAMS above)
0007' 0000                      I00 LINCOL          DEFW    0       ; SET CURSOR PARAMETERS
                                I05                                 ;   LINCOL=COL. (0-31)          37351
                                I02                                 ;   LINCOL+1=LINE (0-23)        37352
0009' C3 0III'                  I03 SETCUR:         JP      SETC00  ; ENTRY TO SET CURSOR POSN.     37353
000C' 1806                      I04 CLRCTL          DEFW    IR00H   ; CLEAR SCREEN CONTROLS
                                I05                                 ;   CLRCTL=STARTING LINE NO.    37356
                                I00                                 ;   CLRCTL+1=LINE COUNT         37357
000E' C3 0304'                  I07 CLRSC0:         JP      CL0S00  ; ENTRY TO CLEAR SCREEN         37388
0011' 0000                      I0E SPARE0          DEFW    0       ; NOT USED (ALIGNMENT ONLY)
0013' 0000                      I09 SPARE1          DEFW    0       ;
0015' 0000                      I10 SPARE2          DEFW    0       ;
0017' 0000                      I11 SPARE3          DEFW    0       ;
0019' 0000                      I12 GETCTL          DEFW    0       ; "GET" CONTROL FROM BASIC      37389
                                I13                                 ; FORMAT IS AS FOR LINCOL
001B' C3 028C'                  I14 GTCHR0:         JP      GTCH00  ; ENTRY TO GET CHARACTER        3738I
001E' C3 020C'                  I18 GETAT0:         JP      GETAB0  ; ENTRY TO GET ATTRIBUTE        37374
0021' C3 02P1'                  I1A GETCUR:         JP      GETC80  ; ENTRY TO GET CURSOR POSN.     37377
0024' 00                        I1F VIMC00          DEFB    0       ; VIDEO MODE CONTROL            37380
0025' C3 0308'                  I18 SETMD0:         JP      STMD00  ; ENTRY TO SET VIDEO MODE       37381
0028' 1701                      I19 SCRCTL          DEFW    1705H   ; SCROLL CONTROL
                                I20                                 ;   SCRCTL=STARTING LINE NO.    37384
                                I25                                 ;   SCRCTL+I=LINE COUNT         37385
002A' C3 012T'                  I22 SCRL0:          JP      SCRL00  ; ENTRY TO SCROLL               37388
                                I23 ;
                                I24 ;COPYS0                         ; ENTRY TO COPY FROM OP         37408
                                I28 ;                               ; (SEE NEXT PAGE)
                                I28 ;EXCHS0                         ; ENTRY TO EXCHANGE SCREEN      3741I
                                I27 ;                               ; (SEE NEXT PAGE)
                                I28 ;
                                I30         SUBTTL  OTHER VARIABLES
                                I31 ;
                                I32 ;
                                I85 ;
                                I84 ;
002C' 17                        I50 BOTLN           DEFB    ITH     ; BOTTOM LINE (LINE AFTER       37389
                                I38                                 ; WHICH SCREEN IS CONSIDERED
                                I37                                 ; FULL.  AUTOMATIC SCROLL
                                I38                                 ; WILL BE DONE IF SCROLL
                                I89                                 ; DOES NOT DECREMENT TO 0.
002D' 01                        I40 SCRLCT          DEFB    1       ; SCROLL COUNT - I PLUS NO.     37E90
                                I41                                 ; OF TIMES AUTOMATIC SCROLL
                                I42                                 ; WILL BE DONE.
                                I48 ;
                                I44 ;
                                I48 ;
                                I46 ;
002F' 3C00                      I47 CHTBL           DEFW    CHBSET  ; ADDRESS OF CHARACTER TABLE    37395
                                I48                                 ; (DEFAULTS TO TS2068 SYSTEM
                                I49                                 ;  ROM)
0031' 0000'                     I50 GRTBL           DEFW    GRPMST  ; ADDRESS OF STD.GRAPHICS TBL.  37397
0033' 20                        I51 LINLEN          DEFB    20H     ; LINE LENGTH (32)              37398
0034' 1025                      I52 CURPOS          DEFW    1025H   ; CURRENT POSITION (INTERNAL    37399
                                I58                                 ;  FORMAT)
0036' 4000                      I54 DFADRE          DEFW    4000H   ; CURRENT DISPLAY FILE ADDRS.   3739B
0038' 00                        I58 MASK0           DEFB    0       ; MASK TO BE APPLIED TO         37400
                                I56                                 ; DISPLAY CHARACTERS
0039' 08                        I57 ATTRTT          DEFB    0H      ; CURRENT ATTRIBUTES            37401
                                I58 ;
003A' 80                        I59 GTINDX          DEFB    80H     ; INDEX FOR ADJUSTING           37402
                                I60                                 ; CH.CODE IN GTCHAR
003B' 0000                      I65 STRGCT          DEFW    0       ; REMAINING COUNT FROM WRSTRG
                                I62 ;                               ; WHEN "SCREEN FULL"            37403
003D' 0000                      I68 DFADE4          DEFW    0       ;                               37408
003F' 00                        I64 ATTNSK          DEFB    0       ; ATTRIBUTE MASK                37407
                                I65 ;
0040' C3 0308'                  I68 COPYS0          JP      COPYS0  ; BASIC ENTRY TO COPY OP        37408
0043' C3 0AP5'                  I67 EXCHS0          JP      EXCHS0  ; BASIC ENTRY TO EXCHANGE OP    37411
                                I69         SUBTTL  WRITE CHARACTER
                                I70 ;
                                I71 ;
0046'                           I72 WRCH0:                          ; HERE FROM BASIC ENTRY TO DISPLAY THE
                                I78                                 ; CHARACTER WHOSE CODE IS IN "DATA0"
0046' 3A 0000'                  I74             LD    A,(DATA0)
                                I78 ;
0049'                           I76 WRCHA0:                         ; ENTRY WITH CODE IN A
0049' CD 0642'                  I77             CALL  LOATTR        ; GET ATTRIBUTE AND MASK CONTROLS
                                I78 ;
                                I79 ;                               ; ENTRY TO USE ATTRIBUTE VALUES IN
                                I80 ;                               ; INTERNAL VARIABLES WITHOUT RELOADING
004C' CD 0050'                  I81 WRCH01: CALL  LOPOSN            ; LOAD REGISTERS FOR CURRENT POSITION
                                I82                                 ;   BC=CURSOR POSITION FROM "CURPOS"
                                I88                                 ;   HL=DISPLAY FILE ADDRESS FROM "DFADRS
                                I84                                 ;   A=CODE FOR CHAR. TO BE DISPLAYED
004F' FE 20                     I85             CP    20H           ; TEST VALID RANGE
0051' DA 00C4'                  I86             JP    C,INVPAR      ; < 20H
0054' FE A8                     I87             CP    0A8H          ;
0056' D2 00C4'                  I88             JP    NC,INVPAR     ; > A4H
0059' C5                        I89             PUSH  BC            ; SAVE CURSOR POSITION
005A' FE 80                     I90             CP    80H           ; TEST IF ASCII PRINTABLE (20H-7FH)
005C' 38 15                     I91             JR    C,WRCH12      ; YES
005E' FE 90                     I92             CP    90H           ; TEST IF STD.GRAPHICS(80H-8FH)
0060' 38 09                     I98             JR    C,WRCH11
0062' DC 48 0C70                I9A             LD    BC,(UDG)      ; USER-DEFINED GRAPHICS(90H-A4H)
```

```
0066'  05                  DEC   B              ; ADJUST ADDRESS-100H
0067'  D6 70               SUB   70H            ; ADJUST FOR INDEX INTO TABLE
0069'  18 0C               JR    WRCH13
006B'  ED 4B 0051'  WRCH11 LD    BC,(GRTBL)     ; ADDRESS OF GRAPHICS TABLE
006F'  D6 80               SUB   80H            ; ADJUST FOR INDEX INTO TABLE
0071'  18 04               JR    WRCH13
0073'  ED 4B 002F'  WRCH12 LD    BC,(CHTBL)     ; CHARACTER TABLE
0077'  EB           WRCH13 EX    DE,HL          ; DE=POSN.IN DISPLAY FILE
0078'  26 00               LD    H,0
007A'  6F                  LD    L,A            ; CODE IS INDEX INTO TABLE
007B'  29                  ADD   HL,HL
007C'  29                  ADD   HL,HL
007D'  29                  ADD   HL,HL
007E'  09                  ADD   HL,BC          ; HL PIXEL PATTERN IN TABLE
007F'  C1                  POP   BC
0080'  EB                  EX    DE,HL
0081'  79                  LD    A,C            ; TEST IF END OF LINE
0082'  3D                  DEC   A
0083'  3A 0035'            LD    A,(LINLEN)
0086'  20 05               JR    NZ,WRCH14
0088'  3C                  INC   A              ; START OF NEW LINE
0089'  05                  DEC   B              ; BUMP LINE NO.
008A'  4F                  LD    C,A            ; LINLEN+1=START OF LINE
008B'  18 01               JR    WRCH15
008D'  3C           WRCH14 INC   A
008E'  B9           WRCH15 CP    C              ; TEST IF START OF LINE
008F'  D5                  PUSH  DE
0090'  CC 00C8'            CALL  Z,TVFULC       ; TEST IF OFF SCREEN
0093'  D1                  POP   DE
                                                ; HERE TO PUT CHARACTER IN DISPLAY FILE
0094'  C5           WRCH2  PUSH  BC             ; CURSOR POSITION
0095'  E5                  PUSH  HL             ; DISPLAY FILE ADRS.
0096'  3A 005B'            LD    A,(MASKB)
0099'  06 FF               LD    B,-1
009B'  1F                  RRA
009C'  38 01               JR    C,WRCH3        ; IF ORING NEW INTO OLD
009E'  04                  INC   B              ; B=-1 IF ORING  =0 IF NOT
009F'  1F           WRCH3  RRA
00A0'  1F                  RRA
00A1'  9F                  SBC   A,A
00A2'  4F                  LD    C,A            ; C=-1 FOR INVERSE  =0 IF NOT
00A3'  3E 08               LD    A,8            ; SCAN COUNT
00A5'  A7                  AND   A
00A6'  EB                  EX    DE,HL
00A7'  08           WRCH5  EX    AF,AF'
00A8'  1A                  LD    A,(DE)         ; OLD BIT ROW
00A9'  A0                  AND   B
00AA'  AE                  XOR   (HL)
00AB'  A9                  XOR   C              ; INVERT IF C=-1
00AC'  12                  LD    (DE),A         ; UPDATE DISPLAY FILE
00AD'  08                  EX    AF,AF'
00AE'  14                  INC   D              ; NEXT SCAN
00AF'  23                  INC   HL             ; NEXT BYTE IN CHAR.FILE
00B0'  3D                  DEC   A              ; TEST IF MORE SCANS
00B1'  20 F4               JR    NZ,WRCH5
00B3'  15           WRCH7  DEC   D              ; ADJUST DF ADRS.
00B4'  EB                  EX    DE,HL          ; DF ADRS. TO HL
00B5'  CD 0665'            CALL  UPDATT         ; UPDATE ATTRIBUTE BYTE
00B8'  E1                  POP   HL             ; STARTING ADDRESS
00B9'  C1                  POP   BC             ; CURSOR POSITION
00BA'  0D                  DEC   C              ; ADJUST CURSOR POSITION
00BB'  23                  INC   HL             ; ADJUST DF ADDRESS
00BC'  CD 0655'     WRCHRT CALL  STPOSN         ; STORE NEW POSITION
00BF'  0E 00        GOODRET LD   C,0            ; RETURN BC=0
00C1'  06 00        ERRRET LD    B,0            ; ENTER HERE WITH C NON-ZERO
00C3'  C9                  RET
                    ;
00C4'  0E 01        INVPAR LD    C,1            ; RETURN BC=1 FOR INVALID PARAMETERS
00C6'  18 F9               JR    ERRRET
                    ;
00C8'  3E 18        TVFULC LD    A,SCRSZ        ; TEST IF NEW LINE IS OFF SCREEN
00CA'  90                  SUB   B              ; A=LINE NO.
00CB'  57                  LD    D,A            ; SAVE IN D
00CC'  3A 002D'            LD    A,(BOTLN) ; GET BOTTOM LINE
00CF'  BA                  CP    D
00D0'  D2 0628'            JP    NC,UPDPOSN     ; ON SCREEN - RETURN TO WRITE CHAR.
                                                ;  VIA UPDATE AND STORE POSITION
00D3'  3A 002B'            LD    A,(SCRLCT)     ; TEST SCROLL COUNT
00D6'  3D                  DEC   A
00D7'  20 0B               JR    NZ,TVFUL1      ; DO AUTOMATIC SCROLL USING SCRCTL
00D9'  3E 01               LD    A,1            ; REINITIALIZE TO 1
00DB'  32 002B'            LD    (SCRLCT),A
00DE'  C1                  POP   BC
00DF'  C1                  POP   BC             ; DISCARD RETURN TO WRITE CHAR.
00E0'  0E 03               LD    C,3            ; RETURN BC=3 FOR SCREEN FULL
00E2'  18 DD               JR    ERRRET
00E4'  32 002B'     TVFUL1 LD    (SCRLCT),A     ; UPDATE VARIABLE
00E7'  ED 4B 002B'         LD    BC,(SCRCTL)    ; GET SCROLL CONTROLS (STARTING LINE
                                                ;   AND LINE COUNT)
00EB'  C3 0141'            JP    SCRLB          ; RETURN TO WRITE CHAR. VIA SCROLL
                    ;
                           SUBTTL WRITE STRING
                    ;                           ; HERE FROM BASIC ENTRY TO
                    ;                           ; WRITE CHARACTER STRING TO SCREEN.
                    ;                           ; STRING IDENTIFIER (CH.CODE) IN
                    ;                           ; SYSTEM VARIABLE PARAMS AT 5CC3H
00EE'  CD 0608'     WRSTRG CALL GETSTRG         ; GET STRING VAR.ADRS. TO HL
00F1'  C8                  RET  Z               ; STRING NOT FOUND OR NULL STRING
                    ;                           ; (BC CONTAINS STATUS)
                    ;
                    ;                           ; ENTRY FROM MACHINE CODE WITH
                    ;                           ; ADDRESS OF CHAR.CODE "STRING"
                    ;                           ; IN HL AND LENGTH IN BC
                    ;
00F2'  CD 06A2'     WRSTRG1 CALL LDATTR         ; GET ATTRIBUTE AND MASK CONTROLS
00F5'  7E           WRSTLP LD    A,(HL)         ; GET CODE
00F6'  E5                  PUSH  HL             ; SAVE ADDRESS
00F7'  C5                  PUSH  BC             ; AND COUNT
00F8'  CD 004C'            CALL  WRCH01         ; WRITE "A"
00FB'  79                  LD    A,C            ; TEST IF GOOD
00FC'  B0                  OR    B
00FD'  20 09               JR    NZ,WRSTX1      ; EXIT IF INVALID CODE OR
                    ;                           ; IF SCREEN FULL
00FF'  C1           WRSTB5 POP   BC             ; COUNT
```

```
0100'  E1                  510          POP   HL              ; ADDRESS
0101'  23                  311          INC   HL              ; NEXT CHAR.
0102'  0E                  312          DEC   BC              ; ADJUST COUNT
0103'  78                  313          LD    A,B             ; TEST IF DONE
0104'  E1                  314          OR    C
0109'  20 EE               313          JR    NZ,WRSTLP       ; WRITE NEXT CHAR.
010T'  C9                  316          RET                   ; RETURN BC=0
                           317  ;
0108'  T9                  318  WRSTRS  LD    A,C             ; SAVE ERROR
0109'  FE 01               319          CP    1               ; TEST IF UNRECOGNIZED CODE
010E'  28 P2               320          JR    Z,WRSTR3        ; CONTINUE
0100'  E1                  321          POP   HL              ; REMAINING COUNT TO HL
010E'  22 009E'            322          LD    (STRGCTI),HL    ; STORE REMAINING COUNT
0111'  E1                  323          POP   HL              ; ADDRESS TO HL
0112'  0E 05               324          LD    C,5             ; RETURN EC=5
0114'  06 00               325  WRSTR2  LD    B,0
0116'  C9                  326          RET
                           327  ;

POSITION CONTROL   I009.SRC

                           329          SUBTTL POSITION CONTROL
                           350  ;
0117'                      351  SETCUR:                       ; HERE FROM BASIC ENTRY.  PARAMETERS
                           352                                ; IN LINCCL
0117'  00 A0 0007'         353          LD    BC,(LINCCL)
                           354  ;
0110'                      355  SETCUP:                       ; ENTRY WITH LINE/COL. IN BC REG.
011E'  CC 040C'            356          CALL  TSTPAR          ; TEST PARAMETERS FOR VALIDITY
0111'  CC 0A10'            357          CALL  CONVFP          ; CONVERT TO INTERNAL FORMAT
0121'  CD 0629'            358          CALL  UPDPOSN         ; CALCULATE AND STORE POSITION
0124'  C3 008P'            359          JP    GOODRET         ; RETURN BC=0
                           360  ;
                           362          SUBTTL SCROLL
                           363  ;
                           364  ;
                           365  ;
0127'                      366  SCRL00:                       ; HERE FROM BASIC ENTRY TO SCROLL
                           367  ;                             ; SCREEN
0127'  ED 40 0020'         368          LD    EC,(SCRCYLI     ; GET CONTROL INFO.
                           369  ;
0120'                      370  SCROLL:                       ; HERE WITH CONTROLS IN BC
                           371                                ;  E=NO. OF LINES
                           372                                ;  C=STARTING LINE NO.
012E'  T0                  373          LD    A,B             ; TEST VALIDITY
012C'  A7                  374          AND   B               ;
0120'  CA 00CA'            375          JP    Z,INVPAR        ; ERROR IF COUNT=0
0130'  EI                  376          CP    B,C
0131'  PE 01               377          CP    1
0133'  DA 00CA'            378          JP    C,INVPAR        ; ERROR IF B+C<1
0136'  PE 19               379          CP    25
013E'  D2 00CA'            380          JP    NC,INVPAR       ; ERROR IF E+C>24
0138'  CD 0141'            381          CALL  SCRL0           ; DO SCROLL
013E'  C3 008P'            382          JP    GOOCRET         ; RETURN BC=0
                           383  ;
0141'  C5                  384  SCPL0   PUSH  BC
0142'  3E I8               385          LD    A,ICRS2         ; GET STARTING LINE IN INTERNAL FORMAT
0144'  9I                  386          SUB   C
0145'  A7                  387          LD    B,A
0146'  3A 0033'            388          LD    B,(CLINLENI
0149'  3C                  389          INC   B               ; COL. 0
014A'  AP                  390          LD    C,A
014E'  CE 065A'            391          CALL  LNBU
014E'  C1                  392          POP   BC
014P'  EI                  393          PUSH  HL              ; SAVE STARTING ADDR.
0190'  C5                  394          PUSH  BC              ; SAVE ORIG.BC FOR EXIT
0191'  T0                  395  T0ITAD  LD    A,L             ; TEST IF AT START OF BLOCK
0192'  B7                  396          AND   A
0191'  28 3C               397          JR    Z,STELK         ; YES
0195   07                  398          RLCB
0196'  07                  399          RLCA
0197'  07                  400          RLCA
0138'  AP                  401          LD    C,A             ; GET NO. OF LINES THIS BLOCK
0199'  IE 0E               402          LD    A,B
019E'  9I                  403          SUB   C
019C'  BE                  404          CP    E               ; TEST AGAINST TOTAL LINES
019D'  IB 4A               405          JR    C,GRBLK         ; TOTAL GREATER THAN THIS BLOCK
019P'  TE                  406  INIDELK LD    A,B
0160'  06 00               407          LD    B,0             ; REMAINING COUNT
0162'  C3                  408          PUSH  BC
0163'  A7                  409  LOOP    LD    B,A             ; B=LINES THIS BLOCK
0164'  0E 0E               410          LD    C,E             ; C=SCAN COUNT
0166'  TE                  411  LOOP0   LD    A,B
0167'  CI                  412          PUSH  BC              ; SAVE SCAN COUNT
0168'  0P                  413          RRCB
016E'  0P                  414          RRCA
01AA'  0P                  415          RRCA                  ; CALCULATE NO. OF BYTES
01AB'  4P                  416          LD    C,A
016C'  0A 00               417          LD    B,0             ; BC=SIZE OF LINES
0168'  E8                  418  LOOPI   EX    DE,HL
01AP'  21 PF80             419          LD    HL,-80H         ; GET ADDR. OF PREV. LINE
0172'  19                  420          ADD   HL,DE
0173'  88                  421          EX    DE,HL           ; TO DE
0174'  E9                  422          PUSH  HL              ; DISPLAY FILE ADR1.
0175'  ED B0               423          LDIR                  ; DO MOVE
0177'  81                  424          POP   HL
0178'  24                  425          INC   H               ; NEXT SCAN LINE
0179'  C1                  426          POP   BC              ; SCAN COUNT
017A'  00                  427          DEC   C
0178'  20 E9               428          JR    NZ,LOOP0        ; MORE SCANS
0170'  C1                  429          POP   BC              ; RESIDUAL LINE COUNT
017E'  78                  430          LD    A,B
017P'  A7                  431          AND   A
0180'  28 04               432          JR    Z,SCRLXT        ; DONE
0182'  2E 00               433          LD    L,0             ; START OF NEXT BLOCK
0184'  18 C8               434          JR    TSTTAD          ; DO REMAINING LINE(S)
018A'  C1                  435  SCRLX7  POP   BC              ; ORIG.BC
018T'  01                  436          POP   DE              ; STARTING ADDR. TO DE
0188'  C9                  437          PUSH  BC              ; SAVE ORIG.BC
01E9'  A8                  438          LD    L,B             ; NO. OF LINES SCROLLED
018A'  26 00               439          LD    H,0             ;
0180'  29                  440          ADD   HL,HL           ; SIZE=NO. OF LINES*
0180'  29                  441          ADD   HL,HL           ; NO. OF ATTRIBUTE BYTES
0188'  29                  442          ADD   HL,HL           ; TO BE SCROLLED
018P'  29                  443          ADD   HL,HL           ;
0190'  29                  444          ADD   HL,HL           ;
```

```
0191'  44
0192'  40
0193'  A2
0194'  6R
0195'  CC 0A92'
019A'  08
019B'  21 FFE0
019C'  19
019D'  E8
019E'  E0 80
01A0'  C1
01A1'  79
01A2'  80
01A3'  10
01A4'  4F
01A5'  08 01
01A7'  18 45

01A9'  4F
01AA'  75
01AB'  91
01AC'  47
01AD'  C5
01AE'  79
01AF'  18 02

01B1'  05
01B2'  C5
01B3'  0E 08
01B5'  C5
01B6'  08
01B7'  21 FBE0
01BA'  19
01BB'  E8
01BC'  81 0020
01BF'  05
01C0'  E0 80
01C2'  81
01C3'  24
01C4'  C1
01C5'  00
01C6'  20 E0
01C8'  C1
01C9'  78
01CA'  A7
01CB'  2B 89
01CD'  11 FB20
0100'  19
0101'  C5 0151'

0104'
0104'  E0 48 000C'

0108'

0108'  7A
0109'  A7
010A'  CA 00C4'
010D'  81
010E'  FE 01
01E0'  0A 00C4'
0103'  FE 19
01E5'  D2 00C4'
01E8'  C0 01EE'
01EB'  C3 008F'

01E0'  CC 06A2'
01F1'  C5
01F2'  5E 18
01F4'  91
01F5'  47
01FA'  5A 0031'
01F9'  5C
01FA'  4F
01FB'  CC 0A5A'
01FE'  50
01FF'  59
0200'  C1
0201'  05
0202'  70
0203'  07
0204'  07
0205'  07
020A'  4F
0207'  5E 08
0209'  91
020A'  08
020B'  5B 47
020D'  78
020E'  06 00
0210'  C5
0211'  47
0212'  08 08
0214'  78
0215'  C5
021A'  EA 07
0215'  0F
3215'  0F
021A'  0F
0218'  4F
021C'  0A 00
021E'  0C
021F'  E5
0220'  54
0221'  5C
0222'  3A 00
0224'  13
```

```
429          LO   B,M          ; COUNT TO BC
426          LO   C,L          ;
427          LO   M,0          ; ADRS. TO ML
428          LO   L,E          ;
429          CALL CALCATT      ; ATTRIBUTE ADRS. TO ML
430          EA   00,ML        ;
411          LO   ML,-20H      ; ADRS. OF PREV.ATTR.BYTE
412          ADD  ML,0E        ;
483          EA   00,ML        ; PREV.BYTE 00
434          LDIR              ; SCROLL ATTRIBUTES
455          POP  BC           ; ORIG. BC
414          LO   A,C          ; STARTING LINE
417          ADD  A,B          ; ADD NO. OF LINES MOVED
418          DEC  A            ; LAST LINE SCROLLED
419          LO   C,A          ;
440          LO   B,1          ; CLEAR 1 LINE
441          JR   CLRSCB       ; CLEAR VACATED LINE AND RETURN
442  ;
448  GRDLR   LO   C,A          ;
444          LD   A,A          ;
445          SUB  C            ; ADJUST TOTAL LINE COUNT BY NO.
44A          LO   B,A          ; OF LINES THIS BLOCK
447          PUSH AC           ;
448          LO   A,C          ; LOAD A NO. OF LINES THIS BLOCK
449          JR   LOOP         ;
450  ;
451  STDLR   DEC  A            ; ADJUST TOTAL LINE COUNT-1
452          PUSH BC           ;
495          LO   C,A          ; SCAN COUNT
454  STALR0  PUSH AC           ; SAVE SCAN COUNT
455  170LA1  EA   0E,ML        ;
456          LO   ML,-720H     ;
457          ADD  ML,C0        ;
458          DA   0E,ML        ; ADRS. OF PREV. LINE
459          LO   BC,12        ; BYTE COUNT
4A0          PUSH ML           ;
461          LDIR              ; DO MOVE
4A2          POP  ML           ;
461          ENC  M            ; TO NEXT SCAN LINE
4A4          POP  BC           ;
465          DEC  C            ; TEST IF MORE SCANS
4AA          JR   NZ,170LR0    ;
4A7          POP  BC           ; REMAINING LINE COUNT
4A8          LO   A,A          ;
4A9          AND  A            ;
470          JR   Z,SCRLAT     ;
471          LO   00,-780H     ; ADJUST TO LINE 1
472          ADD  ML,00        ;
471          JP   7E57A0       ; BC=LINE COUNT
474                            ; ML=OF ADDRESS
475  ;
477          SUBTTL  CLEAR SCREEN
478  ;
479  ;
480  CLRSAB: ;                 ; HERE FROM BASIC ENTRY TO CLEAR SCREEN
481          LO   BC,(CLRCTL)  ; GET CONTROL INFO.
4A2  ;
481  CLR1CN: ;                 ; ENTRY WITH BC=LINE COUNT AND
484                            ; STARTING LINE NO. FOR CLEAR
485          LO   A,B          ;
48A          AND  A            ; TEST VALIDITY
487          JP   Z,INVPAR     ; ERROR IF COUNT=0
4A8          ADD  A,C          ;
4A9          CP   1            ;
490          JP   C,INVPAR     ; ERROR IF B=C<1
491          CP   25           ;
492          JP   NC,INVPAR    ; ERROR IF B=C>24
A95          CALL CLR1C0       ; DO SERVICE
494          JP   GOODRET      ; RETURN BC=0
495  ;
49A  ;
497  CLRSC0: CALL LOATTR       ; GET ATTRIBUTE CTL
498          PUSH BC           ;
499          LO   A,1CH*32     ; CONVERT TO INTERNAL FORMAT
500          SUB  C            ;
501          LO   B,A          ;
502          LD   A,(CLINLEN)  ;
501          INC  A            ; SET TO COL.0
504          LO   C,A          ;
505          CALL LNAD         ; GET ADDRESS
506          LO   D,0          ;
507          LD   B,C          ; SAVE POSITION
508          POP  BC           ; ORIG. BC
509          PUSH DE           ; SAVE CURSOR POSITION
510  CLRSC1  LO   A,L          ; GET # OF LINES THIS BLOCK
511          ALCA              ;
512          ALCA              ;
511          ALCA              ;
514          LO   C,A          ;
515          LO   A,B          ;
516          SUB  C            ;
517          CP   5            ; COMPARE TO TOTAL LINE COUNT
518          JR   C,CLR1C7     ;
519  CLR1C2  LO   A,A          ; NO. OF LINES
520          LO   A,D          ; REMAINING LINES
521          PUSH BC           ;
522  CLRSC1  LO   B,A          ;
525          LO   C,A          ; SCAN COUNT
526  CLRSC4  LO   A,B          ;
525          PUSH BC           ;
924          AND  7            ;
527          RRCA              ;
52E          RRCA              ;
529          RRCA              ;
550          LO   C,A          ; BC=32*NO. OF LINES
551          LO   B,0          ; (=0 IF 256 FOR 8 LINES)
552          DEC  C            ; ADJUST
555  CLRSC5  PUSH ML           ;
51A          LO   D,M          ;
535          LO   E,L          ;
556          LD   (ML),00      ; CLEAR DP
517          INC  DE           ;
```

```
0225'  EG 80                    LOI8
0227'  E1                       POP   HL
0228'  24                       INC   H            ; NEXT SCAN ROW
0229'  C1                       POP   BC           ; POP SCAN COUNT
0226'  00                       OEC   C
0228'  20 ET                    J8    NI,CLBSC6
0220'  EE                       PUSH  HL           ; SAVE NEXT OF ADRS.
022E'  2E                       OEC   H            ; LAST SCAN ROW
022F'  C0 0682'                 CALL  CALCATT      ; GET ADRS. IN ATTRIBUTE FILE
0232'  78                       LO    A,E          ; NO. OF LINES THIS BLOCK
0233'  E6 07                    ANO   T
025E'  0F                       RRCA
0256'  0F                       RRCA
0257'  0F                       RRCA
0258'  AF                       LO    C,A          ; EC=32*NO. OF LINES
0258'  06 0E                    LD    B,0
0258'  00                       OEC   C            ; ADJUST
025C'  54                       LD    D,H
0150'  E0                       LO    E,L
0255'  5A 0055'                 LO    A,(ATTETT)   ; ATTRIBUTE VALUE
0241'  TT                       LO    (HL),A       ; UPDATE ATTRIBUTE FILE
0242'  13                       INC   DE           ; NEXT BYTE
0243'  E0 80                    LOI8               ;
0245'  E1                       POP   HL           ; OF ADRS.(LAST SCAN+100H)
0246'  C1                       POP   BC           ; TOTAL LINE COUNT
0247'  18                       LO    A,E
024E'  AT                       ANO   A
0240'  28 05                    J8    Z,CLBSXT
0248'  2E 00                    LO    L,B          ; HL=START OF NEXT BLOCK
0240'  C3 0202'                 JP    CLRSC1       ; E=REMAINING LINE COUNT
0150'  C1               CLBSXT  POP   AC           ; POSITION
0251'  C3 0620'                 JP    UPOPOSN      ; RETURN VIA UPDATE AND STORE POSITION
0254'  4F               CLBSCT  LO    C,A          ; SAVE NO. LINES THIS BLOCK
0155'  TE                       LO    A,B
0256'  B1                       SUE   C            ; ADJUST TOTAL LINE COUNT
0157'  4T                       LO    B,A
0250'  C1                       PUSH  EC           ; REMAINING LINE COUNT
0155'  T9                       LO    A,C          ; LINES THIS BLOCK
025A'  1E 85                    JR    CLRSC5       ;
                                SUBTTL "GET" ROUTINES
                        ;
                        ;
025C'                   GTCHB0:                    ; HERE FROM BASIC ENTRY TO GET CHARACTER
                                                   ; (RETURNS CODE FOR CHARACTER AT
                                                   ;  POSITION SPECIFIED IN GETCTL)
025C'  E0 4B 0018'              LO    BC,(GETCTL)  ; GET CONTROL INFO.
                        ;
026D'                   GTCHA8:                    ; ENTRY WITH POSITION IN BC
0260'  C0 06DC'                 CALL TSTPAR        ; TEST PARAMETERS
0263'  C0 0610'                 CALL CONVFM        ; CONVERT TO INTERNAL FORMAT
0266'  C0 0620'                 CALL CALCPOS       ; GET DISPLAY FILE ADRS.
0268'  ED 5B 002F'              LD    DE,(CHTBL)   ; CHAR.TABLE
026D'  D6 60                    LD    B,86         ; NO. OF PRINTABLE CHARACTERS
026F'  3E 80                    LO    A,8CH        ; SET ADJUSTMENT INDEX
0271'  32 003A'         GTCH1   LD    (GTINDX),A
0274'  14                       INC   D            ; ADJUST TO START OF TABLE
0275'  EB                       EX    DE,HL        ; OF ADRS.IN DE;CHAR.SET IN HL
0276'  C5               GTCH2   PUSH  BC           ; SAVE COUNT
0277'  0E                       PUSH  DE           ; SAVE ADRS. IN DF
0278'  E5                       PUSH  HL           ; SAVE ADRS. IN CHAR.TABLE
0278'  1A                       LD    A,(DE)       ; SCAN ROW FROM OF
0276'  AE                       XOR   (HL)         ; TEST AGAINST CHAR. SET
0278'  28 10                    JR    Z,GTCH3      ; MATCH
0270'  FE                       PUSH  AF
027E'  3A 003A'                 LD    A,(GTINDX)
0281'  FE 80                    CP    80H          ; TEST IF STD.GRAPHICS
0283'  20 03                    J8    NZ,GTCH23
0285'  F1                       POP   AF
0286'  18 27                    J8    GTCH4        ; DO NOT TEST FOR INVERSE
0288'  F1               GTCH23  POP   AF
0288'  3C                       INC   A            ; TEST INVERSE
028A'  20 23                    JR  NZ,GTCH4
028C'  30                       OEC   A            ; A=-1 FOR INVERSE
018D'  AF               GTCH3   LD    C,A          ; C=0 FOR MATCH  -1 FOR INVERSE
028E'  06 07                    LD    B,7
0290'  1A               GTCHS1  INC   D            ; NEXT SCAN IN OF
028E'  23                       INC   HL           ; NEXT BYTE IN CHAR.SET
0292'  1A                       LO    A,(DE)
0293'  AE                       XOR   (HL)
0294'  A8                       XOR   C
0295'  20 18                    J8    NZ,GTCH4           ; NO MATCH
0297'  10 F7                    OJNZ  GTCHS1    ; FOR NEXT SCAN
                                                ;
                                                ; HERE IF MATCH WITH (SP)=CHAR IN CHAR SET
                                                ; (SP+2)=CHAR.IN DISPLAY FILE (SP+5)=CHAR.
                                                ; COUNT
0299'  T5                       LO    A,C       ; A=-1 IF MATCH ON INVERSE
0286'  C1                       POP   BC
0288'  C1                       POP   EC
028C'  C1                       POP   BC
029D'  AF                       LO    C,A       ; B=COUNT C=-1 IF MATCH ON INVERSE
028E'  3A 003A'                 LO    A,(GTINDX)        ; ADJUST CHAR.CODE
02A1'  ED                       SUB   A         ; A=CHAR.CODE
02A2'  FE 20                    CP    20H       ; TEST IF SPACE
02A4'  20 05                    J8    NZ,GTCH32
01A6'  0C                       INC   C         ; TEST IF MATCH ON INVERSE SPACE
02A7'  20 02                    JR    NZ,GTCH32
02A8'  3E 8F                    LO    A,8FH     ; SET CODE FOR GRAPHICS BLOCK
02AB'  4F               GTCH32  LO    C,A
02AC'  06 00                    LO    B,0
02AE'  C5                       RET             ; CHAR.CODE IN A AND IN BC
                        ;
02AF'  E1               GTCH4   POP   HL        ; HERE WHEN NO MATCH
0280'  11 0008                  LO    DE,8      ; LOOK AT NEXT CHAR. IN SET
0283'  19                       ADD   HL,DE             ; CHAR. SET
0284'  D1                       POP   DE                ; CHAR. IN OF
028E'  C1                       POP   BC                ; CHAR.COUNT
0286'  10 8E                    OJNZ  GTCH2     ; LOOK AGAIN
                                                ;
                                                ; HERE TO TEST IF DONE
028E'  EB                       EX    DE,HL     ; OF ADRS.TO HL
0188'  3A 003A'                 LO    A,(GTINDX)
028C'  FE 40                    CP    80H       ; TEST IF STD.CHR.SET
```

```
02BE'  20 0A                  JB    N2.GTCM3   ; TRY GRAPHICS
02C0'  E0 3B 0031'            LO    OE.(GRTBL); STO.GRAPHICS TABLE
02C4'  06 10                  LO    B.56       ; NO. OF ENTRIES
02C6'  3E 90                  LD    A.90H      ; INDEX
02C8'  18 AT                  JB    GTCM1      ;
02CA'  FE 90           GTCM5  CP    90H        ; TEST IF STD.GRAPHICS
02CC'  20 0B                  JR    NZ.GTCM6   ; DONE IF NOT
02CE'  ED 3B 3CTB             LO    OE.(UOG)   ; TRY USER-DEFINED GRAPHICS AREA
02D2'  15                     DEC   D          ; ADJUST ADDRESS-100H
02D3'  06 15                  LO    E.21       ; NO OF ENTRILS
02D5'  3E A5                  LO    A.0A3H     ; INDEX
02D7'  18 9E                  JR    GTCM1
02D9'  4B              GTCN6  LO    C.B        ; HERE WHEN NO MATCH ANYWHERE
02DA'  AF                     XOR   A          ; SET EC AND A=ZERO
02DB'  C9                     RET

                                               ; GET ATTRIBUTE BYTE
                       ;                       ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                       ;
02DC'  E0 4E 0015'     GETAED: LO   BC.(GBTCTL)
                       ;                       ; HERE WITH LIN/COL IN BC
                       ;
02E0'  C0 060C'        GETATT: CALL TSTPAR     ; TEST PARAMETERS
02E3'  CD 0610'               CALL  CCNVPN     ; CONVERT TO INTERNAL FORMAT
02E6'  CD 0A20'               CALL  CALCPOS    ; CALCULATE OF POSITION
02E9'  CC 0692'               CALL  CALCATT    ; CALCULATE ATTRIBUTE POSITION
02EC'  7E                     LO    A.(HL)     ; ATTRIBUTE BYTE TO A
02ED'  4F                     LO    C.A        ; PASS BACK IN BC
02EE'  06 00                  LO    E.0
02F0'  C9                     RET
                       ;                       ; GET CURSOR POSITION
02F1'                  GBTCEO:                 ; HERE FROM BASIC ENTRY
02F1'                  GETCUR:
02F1'  EG 4E 0034'            LO   EC.(CURPOS)         ; GET INTERNAL POSITION
02F5'  C0 0E10'               CALL CONVPM              ; CONVERT TO USER FORMAT
02F8'  3A 0033'               LO   A.(LINLEN)
02FB'  E5                     CP   C                   ; TEST IF END OF LINE(=32)
02FC'  20 0E                  JR   NZ.GETCS            ; NO
02FE'  0B 00                  LO   C.0                 ; NEXT POSN.START OF NEXT LINE
0300'  7B                     LO   A.E
0301'  3C                     INC  A                   ; BUMP TO NEXT LINE
0302'  47                     LO   E.A
0303'  FE 18                  CP   24                  ; TEST IF OFF SCREEN
0305'  38 02                  JR   C.GETC1             ; NO
0307'  E6 17                  LO   E.23                ; POSN. IS AT BOTTOM LINE
0309'  EC 43 0007'     GETC1  LO   (LINCOL).BC         ; STORE FOR BASIC PROGRAM
030D'  C9                     RET                      ; VALUES ALSO IN BC
                       ;

                              SUBTTL  VIDEO MODE CONTROL
                       ;
                       ;
030E'                  STNG00:                 ; HERE FROM BASIC ENTRY WITH MODE IN
                                               ; "VIMODE"
030E'  36 0026'               LD  6.(VIMODE)
                       ;
0311'                  SETMODE:                ; ENTRY WITH MODE IN 6
                       ;
0311'  67                     LO    E.6        ; MODE TO E
0312'  67                     AND   6
0313'  2E 15                  JB    2.SETM00   ; TEST FOR VALIDITY
0315'  FE 80                  CP    80H        ;
0317'  2E 11                  JR    2.SETMOE   ;
0319'  FE 01                  CP    1
031B'  2E 00                  JR    2.SETM00
031D'  FE E4                  CP    6
031F'  CA 03BA'               JP    2.PRPV     ; HANDLE 6 AND 3
0322'  FE 01                  CP    5
0324'  CA 03BA'               JP    2.PPPV     ;
0327'  C3 0BC4'               JP    INVPAR     ; INVALID PARAMETER
032A'  3A 3CC2         SETM00 LO    6.(VIDMD0) ; TEST CURRENT MODE
032D'  67                     AND   A
032E'  20 5A                  JR    NZ.SETMO2  ; OP2 OPEN
0330'  E0                     OR    B          ; CURRENT MODE=0 - TEST IF NEW MODE=0
0331'  C6 000F'               JP    2.GDOORET  ; NO ACTION
                       ;
                       ;                       ; INITIAL OPENING OF 2ND OF
0334'  21 17D1         SETNGI LO    HL.SCINIT  ; INITIALIZE VARIABLES
0337'  22 002E'               LO    (SCRCTL).HL ; SCROLL CONTROL
033A'  3E 17                  LO    A.23       ;
033C'  32 0020'               LO    (BOTLN).A  ; BOTTOM LINE
033F'  3E 01                  LO    A.1        ;
0341'  32 002E'               LO    (SCRLCT).A ; SCROLL COUNT
0344'  3E 20                  LO    A.32       ;
0346'  32 0033'               LO    (LSHLEN).A ; LINE LENGTH
0349'  21 1800                LO    HL.CLINIT  ;
034C'  22 000C'               LO    (CLRCTL).HL ; CLEAR SCREEN CONTROL
034F'  AF                     XOR   A
0350'  32 003B'               LO    (STRGCT).A ; WRSTPG REMAINING COUNT
0353'  32 003C'               LO    (STRGCT+1).A ;
0356'  21 3C00                LO    HL.CHRSET  ; STD.CHAR.SET
0359'  22 002F'               LO    (CHTBL).HL ;
035C'  21 03E0'               LO    HL.GRPHST  ; STD. GRAPHICS SET
035F'  22 0031'               LO    (GRTBL).HL ;
                       ;                       ;
                       ;                       ; INITIALIZE BASIC INPUTS
0362'  32 0000'               LO    (DATAE).A  ; DATA BYTE
0365'  32 0007'               LO    (LINCOL).A ; COLUMN
0368'  32 0008'               LO    (LINCOL+1).A ; LINE
036B'  32 0019'               LO    (GETCTL).A ; "GET" COLUMN
036E'  32 001A'               LO    (GETCTL+1).A ; "GET" LINE
                       ;                       ; TEST IF ENOUGH MEMORY
0371'  21 12C0                LO     HL.SNSBRTS2
0374'  15 0B40                LO     DE.MOVBS2
0377'  19                     ADD    HL.DE      ; HL=TOTAL ROOM NEEDED
0378'  EC 5E 5C65             LO     DE.(STXEND)
037C'  19                     ADD    HL.DE      ; ADD MEMORY IN USE
037D'  0A 03E3'               JP     C.EXTNEM   ; NOT ENOUGH MEMORY TO OPEN OP2
0380'  ED 3B 5CB2             LO     DE.(RAMTOP)
0384'  A7                     AND    A
0385'  BC 52                  SBC    HL.DE
0387'  D2 0353'               JP     NC.EXTNEM ; NOT ENOUGH MEMORY
038A'  78              SETM02 LO     A.B       ; MODE TO A
038B'  CC 03F3'               CALL   ENBLEXT   ; ENABLE ROM EXT.
038E'  CG 03CD'               CALL   GETVAL    ; H/W VALUE TO B; VIDMOD VALUE TO C
```

```
0391' C5              767         PUSH  BC        ; SAVE
0392' T8              768         LD    A,R
0393' A7              769         AND   A         ; TEST IF 0
0394' 3C 01           770         JR    NZ,SETMZ1
0396' T9              771         LD    A,C       ; VALUE FOR CHNGVID IN C
0397' CD 08R8         772  SETMZ1  CALL  CHNGVID  ; CALL RTN. TO OPEN/CLOSE 2ND DP
039A' CD 0600'        773         CALL  ENABLMSG  ; REENABLE HOME RANG
039D' C1              774         POP   BC
039E' T9              775         LD    A,C
039F' 32 5CC2         776  SETMOS  LD    (VIOMOD),A; UPDATE VIOMCO
03A2' 3A 0033'        777  UPDATE  LD    A,(LINLEN)        ; GET LINE LENGTH
03A5' 9C              778         INC   A
03A6' AF              779         LD    C,A               ; CCLUMN
03A7' 06 18           780         LD    B,SCRSI           ; BC=HOME POSITION (LN.0/CCL.0)
03A9' 21 0000         781         LD    HL,0
03AC' 22 5CT0         782         LD    (CCCORDS),HL      ; INITIALIZE PLOT POSITION
03AF' CD 0628'        783         CALL  UPDPOSN           ; UPDATE AND STORE HOME POSITION
03B2' C3 08RF'        784         JP    GOCORET           ; RETURN BC=0
03B5' 08 02           785  BATMRN  LD    C,2               ; RETURN RC=2 FOR NO ROOM
03B7' C3 00C1'        786         JP    ERPRET
                      787  ;
                      788  ;                               ; MODE = 4 OR 5 USED TO DIRECT DATA/
                      789  ;                               ; ATTRIBUTES TO DISPLAY FILE 1 OR 2
                      790  ;                               ; WHILE THE ALTERNATE FILE IS AT THE
                      791  ;                               ; SCREEN.  NO CHANGE TO H/W SETTING
                      792  ;
                      793  ;                               ; INPUT     CURRENT VIOMCO   NEW VIOMCO
                      794  ;                               ; -----     --------------   ----------
                      795  ;
                      796  ;                               ;   A           80             80
                      797  ;                               ;   4           81             84
                      798  ;                               ;   A           84             8A
                      799  ;                               ;   4           88             88
                      800  ;
                      801  ;                               ;   0           80             85
                      R02  ;                               ;   5           81             51
                      R03  ;                               ;   5           84             R1
                      804  ;                               ;   0           85             85
                      805  ;
03BA' 3A 5CC2         806  PERV    LD    A,(VIOMOD)        ; ONLY VALID IF VIOMOD=80,81,84 OR 85
03BD' AF              807         LD    C,A               ; CURRENT VIOMOD TO C
03BE' C8 TF           808         BIT   T,A
03C0' CA 00C4'        809         JP    I,INVPAR          ; INVALID PARAMETER
03C3' T8              810         LD    A,R               ; MODE 4 OR 5 TO A
03C4' FE 04           811         CP    4
03C6' 20 0A           812         JR    NZ,DTAOP2         ; MODE 5 - DATA TO OP2
                      813  ;                               ; MODE 4 - DATA TO OP1
03C8' C8 41           814  DTAOP1  BIT   0,C             ; TEST IF OP1 ALREADY DEST.
03CA' CA 00RF'        815         JP    I,GOODRET         ; NO ACTION
03CD' AR              816         XOR   C                 ; SET VIOMOD=80 OR 84
03CE' E6 FE           817         AND   0FEH              ; BIT 0=0
03D0' 18 C8           818         JR    SETMOD            ; UPDATE VARIABLES
                      819  ;
                      820  ;                               ; MODE 5 - DATA TO OP2
03D2' C8 A1           821  DTAOP2  BIT   0,C             ; TEST IF OP2 ALREADY DEST.
03D4' C2 00RF'        822         JP    NZ,GOODRET        ; NO ACTION
03D7' A9              823         XOR   C                 ; SET VIOMOD=81 OR 85
03D8' IR C5           824         JR    SETMOS            ; UPDATE VARIABLES
                      825  ;
                      827         SUBTTL  COPY SCREEN
                      82E  ;
                      825  ;
                      850  ;                               ; COPY LINES WITHIN OR BETWEEN
                      851  ;                               ; DISPLAY FILES
                      83I  ;
                      855  ;
03DA'                 856  COPTSOI                         ; USED FROM BASIC ENTRY
03DA' 2A 5C50         855         LD    HL,(CH_ADD)       ; SAVE CURRENT POINTER
03DD' E5              83R         PUSH  HL
03DE' CD 0668'        83T         CALL  GETSTRG           ; GET STRING OF PARAMETERS
03E1' 20 05           83R         JR    NZ,COPTO1         ; FOUND STRING
03E3' E1              835         POP   HL                ; RESTORE CH_ADD AND RETURN
03E4' 22 5C50         840         LD    (CH_ADD),HL       ; ERROR (NO STRING OR NULL STRING)
03E7' C8              841         RET                     ; (BC CONTAINS STATUS)
                      842  ;
03E8' =0004           845  COPTWK  DEFS  4                 ; WORK AREA FOR BASIC PARAMETERS
                      8AA  ;
                      845  ;
03EC' 22 5C50         846  COPTO1  LD    (CH_ADD),HL       ; ADRS. OF STRING TO CH_ADD
03EF' CD 0418'        847         CALL  GETNO             ; GET PARAMETERS
03F2' 32 03ER'        84R         LD    (COPYWK+1),A      ; SOURCE LINE NO.
03F5' CD 0418'        849         CALL  GETNO             ;
03F8' 32 03E8'        850         LD    (COPYWK),A        ; DEST. LINE NO.
03FB' CD 041R'        851         CALL  GETYNO
03FE' 32 03EA'        852         LD    (COPYWK+5),A      ; SOURCE DP
0A01' CD 0418'        855         CALL  GETNO
0A04' 32 03RA'        854         LD    (COPTWK+2),A      ; DEST. DP
0A07' CD 0A30'        855         CALL  GETNO
0A0A' 0I              85R         POP   HL                ; RESTORE CH_ADD
0A0B' 22 3C5D         85T         LD    (CH_ADD),HL
0A0E' E0 4R 03ER'     858         LD    BC,(COPYWK)
0A12' 0C 58 03RA'     855         LD    C8,(COPTWK+2)
0A16' 18 29           860         JR    COPTSC            ; TEST VALIDITY AND DO COPY
                      8R1  ;
0A1R' 2A 5C50         862  GETNO   LD    HL,(CH_ADD)       ; LOAD ADDRESS
0A59' T8              865         LD    A,(HL)            ; GET CHAR. FROM STRING
0A1C' FE 2C           864         CP    2CH               ; TEST IF COMMA
0A1R' 2R 0P           8R5         JR    Z,GETNZ
0A20' FE 30           86R         CP    30H               ; TEST IF DIGIT
0A22' 38 II           86T         JR    C,COPERR          ; INVALID PARAMETER
                      868  ;                               ; OR END OF STRING
0A24' FE 5A           865         CP    3AH
0A2R' 30 0C           8T0         JR    NC,COPERR
0A2R' CC 50P5         8T1  GETNI   CALL  INTNT             ; CONVERT TO BINARY
0A28' C0 5IE5         8T2         CALL  PP2A
0A2E' D0              8T5         RET   NC                ; VALID DIGIT IN A
0A2F' 25              8T4  GETNZ   INC   HL
0A30' 22 5C50         8T5         LD    (CH_ADD),HL
0A33' 58 E9           878         JR    GETNO             ; TEST NEXT CHAR.
                      8TT  ;
0A30' 0I              878  COPERR  POP   HL              ; DISCARD RETURN FROM GETNO
```

```
0456'  01               COP0BI  POP     HL              ; RESTORE CH_ADD
0457'  22 0C80                   LD      (CH_ADDS),HL
045A'  C3 00C4'                  JP      INVPAR          ; RETURN BC=1 FOR INVALID
                                                         ; PARAMETER(S)
                         ;
                         ;
045D'                    COPYSCR                         ; ENTRY WITH PARAMETERS IN REGS.
                         ;                               ; AS NOTED
045D'  B7                        AND     A               ; TEST VALIDITY
0438'  CA 00C4'                  JP      Z,INVPAR        ; ERROR IF LINE COUNT=0
0441'  FE 19                     CP      25              ;
0443'  D2 00C4'                  JP      NC,INVPAR       ; ERROR IF > 24
0446'  F5                        PUSH    AF
0447'  CD 05A0'                  CALL    GTADRS          ; GET OF ADRS. FOR SOURCE
044A'  DA 00C4'                  JP      C,INVPAR        ; INVALID PARAMETER DETECTED
044D'  E5                        PUSH    HL              ; SAVE SOURCE ADDRESS
044E'  41                        LD      B,C             ; DEST. LINE
044F'  93                        LD      D,E             ; DEST. OF
0450'  CD 0560'                  CALL    GTADRS          ; GET OF ADRS. FOR DESTINATION
0453'  DA 00C4'                  JP      C,INVPAR
0456'  D1                        POP     DE              ; SOURCE ADRS.
0457'  EB                        EX      DE,HL           ; HL=SOURCE  DE=DEST.
0458'  F1                        POP     AF              ; GET COUNT
0459'  47                        LD      B,A
045A'  E5                        PUSH    HL              ; SAVE STARTING ADDRESSES
045B'  D5                        PUSH    DE              ;
045C'  C5                        PUSH    BC              ; SAVE ORIG. LINE COUNT
045D'  7D                COPYS1  LD      A,L             ; CALCULATE # OF LINES SOURCE BLOCK
045E'  07                        RLCA
045F'  07                        RLCA
0460'  07                        RLCA
0461'  4F                        LD      C,A             ;
0462'  3E 08                     LD      A,8
0464'  91                        SUB     C
0465'  4F                        LD      C,A             ; # OF LINES THIS BLOCK
0466'  C5                        PUSH    BC              ; SAVE NO.OF LINES/# OF LINES SOURCE BLK.
0467'  B8                        CP      B               ; COMPARE TO TOTAL LINES
0468'  38 72                     JR      C,COPYS3        ; TOTAL LINES > THIS BLOCK
046A'  7B                        LD      A,E             ; CALCULATE # OF LINES DEST.BLOCK
046B'  07                        RLCA
046C'  07                        RLCA
046D'  07                        RLCA
046E'  4F                        LD      C,A
046F'  3E 08                     LD      A,8
0471'  91                        SUB     C               ; A=# OF LINES DEST.BLOCK
0472'  C1                        POP     BC              ; B=TOTAL LINES
0473'  4F                        LD      C,A             ; C=# OF LINES DEST BLOCK
0474'  B8                        CP      B               ; B=TOTAL LINES TO BE COPIED
0475'  38 79                     JR      C,COPYS2        ; TOTAL > DEST.BLOCK - USE DEST.LINE CNT.
                         ;
                         ;                               ; COPY IS WITHIN BOTH SOURCE AND DEST.BLOCKS
                         ;
0477'  78                COPYS2  LD      A,B             ; NO. OF LINES
0478'  0A 08                     LD      B,0             ; REMAINING LINES
047A'  C5                        PUSH    BC              ; SAVE REMAINING LINE COUNT
047B'  47                COPYS3  LD      B,A             ; NO. OF LINES
047C'  0E 08                     LD      C,8             ; SCAN COUNT
047E'  78                COPYS4  LD      A,B             ; NO. OF LINES
047F'  C5                        PUSH    BC              ; SAVE SCAN COUNT
0480'  EA 07                     AND     7
0482'  0F                        RRCA
0483'  0F                        RRCA
0484'  0F                        RRCA
0485'  4F                        LD      C,A             ; BC=32 * NO. OF LINES
0486'  0A 00                     LD      B,0             ; (C=0 IF 256 FOR 8 LINES)
0488'  A7                        AND     A               ; TEST IF 0
0489'  20 02                     JR      NZ,COPY41
048B'  0A 01                     LD      B,1             ; SET TO 256
048D'  E5                COPY41  PUSH    HL              ; SAVE ADDRESSES
048E'  D5                        PUSH    DE
048F'  ED B0                     LDIR                    ; COPY "BC" BYTES
0491'  D1                        POP     DE              ; RESTORE ADDRESSES
0493'  E1                        POP     HL
0493'  14                        INC     D               ; ADJUST FOR NEXT SCAN LINE
0494'  24                        INC     H               ;  (+100H)
0495'  C1                        POP     BC              ; SCAN COUNT
0496'  0D                        DEC     C               ; DECREMENT SCAN COUNT
0497'  20 E8                     JR      NZ,COPYS4       ; NEXT SCAN ROW
0499'  78                        LD      A,B             ; SAVE # OF LINES JUST MOVED
049A'  C1                        POP     BC              ; RESIDUAL LINE COUNT
049B'  4F                        LD      C,A             ;
049C'  78                        LD      A,B             ;
049D'  A7                        AND     A
049E'  28 22                     JR      Z,COPY47        ; DONE
04A0'  C5                        PUSH    BC              ; B=REMAINING LINE COUNT
04A1'  79                        LD      A,C
04A2'  EA 07                     AND     7
04A4'  0F                        RRCA
04A5'  0F                        RRCA
04A6'  0F                        RRCA
04A7'  4F                        LD      C,A
04A8'  06 00                     LD      B,0
04AA'  A7                        AND     A
04AB'  20 02                     JR      NZ,COPY42
04AD'  0A 01                     LD      B,1             ; BC = NO. OF BYTES JUST MOVED
04AF'  25                COPY42  DEC     H               ; ADJUST ADDRESSES TO START
04B0'  09                        ADD     HL,BC           ; OF NEXT LINE IN EACH FILE
04B1'  3E 78                     LD      A,78H
04B3'  A4                        AND     H
04B4'  67                        LD      H,A
04B5'  E5                        PUSH    HL              ; SAVE ADJUSTED SOURCE ADRS
04B6'  15                        DEC     D
04B7'  EB                        EX      DE,HL           ; DEST. ADRS. TO HL
04B8'  09                        ADD     HL,BC
04B9'  3E 78                     LD      A,78H
04BB'  A4                        AND     H
04BC'  67                        LD      H,A
04BD'  D1                        POP     DE              ; SOURCE ADRS. TO DE
04BE'  EB                        EX      DE,HL           ; HL=SOURCE  DE=DEST.
04BF'  C1                        POP     BC
04C0'  18 98                     JR      COPYS1          ; B=REMAINING LINE COUNT
04C2'  C1                COPY47  POP     BC              ; ORIG.LINE COUNT
04C3'  68                        LD      L,B             ; CALCULATE # OF ATTRIBUTE BYTES
```

```
04C4'  2A G0                    LO      H,0
04CA'  I9                       ADD     HL,HL
04C7'  29                       ADD     HL,HL
04CE'  29                       ADD     HL,HL
04G9'  29                       ADD     HL,HL
04CA'  29                       ADD     HL,HL         ; E2 * NO. OF LINES
04C6'  44                       LO      A,H
04CC'  40                       LO      C,L
04C0'  E1                       POP     HL            ; DEST.ADDRESS TO HL
04CE'  CO 0A92'                 CALL    CALCATT       ; ADRS. OF ATTRIBUTE FILE
0431'  EE                       EX      GE,HL         ; DEST.ADRS. TO DE
0402'  E1                       POP     HL            ; SOURCE ADDRESS
040E'  CO 0A92'                 CALL    CALCATT       ; ADRS. OF ATTRIAUTE FILE
040A'  E0 E0                    LDIA                  ; COPY ATTRIAUTES
040E'  C3 00EF'                 JP      GOODAET       ; RETURN AC=0
                        ;
0408'  A1               COPTSS  LO      A,C           ; # OF LINES SOURCE BLOCK TO A
040C'  YE                       LO      A,E           ; CALCULATE # OF LINES DEST. BLOCK
0430'  0T                       RLCA
040E'  01                       RLCA
040F'  07                       ALCA
04E0'  4F                       LO      C,A
04E3'  EE 0E                    LO      A,E
04E3'  91                       SUE     C             ; A=# LINES DEST.BLOCK
04E4'  4F                       LO      C,A           ; SAVE IN C
04EE'  A6                       CP      E             ; COMPAAE TO LINES SOURCE ALOCK
04E6'  EE 0E                    JE      C,COPTSY      ; DEST.SIZE SMALLER - USE DEST.LINE CNT.
04E0'  C1                       POP     6G            ; 6=TOTAL LINE COUNT/C=SOURCE LINE CNT.
04E9'  TE               COPT3A  LO      A,A
04EA'  91                       SU9     C
04EE'  4T                       LO      6,A           ; E=REMAINING LINECOUNT
04EC'  CE                       PUSH    EC
04E0'  19                       LO      A,C           ; K OF LINES
04EE'  1E EE                    JR      COPY33        ; COPY "A" LINES
04F0'  T9               COPTS7  LO      A,C           ; # LINES DEST.BLK
04F1'  C1                       POP     EC            ; E=TOTAL LINE COUNT
04FI'  4F                       LO      C,A           ; C=DEST. LINE CNT.
04F3'  1A F4                    JR      COPYSA

                                EUATTL  EACHANGE SCAEEN
                        ;
                        ;                             ; EACHANGE LINES WITHIN DA BETWEEN
                        ;                             ; DISPLAT FILES
                        ;
04FE'                   EACHSO                        ; ENTET FAON BASIC - GET PARAMETEES
                        ;                             ; FAOM STRING TO APPROPAIATE AEGS.
                        ;
64F5'  IA SCE0                  LO      HL,(CH_ADD)   ; SAVE CURRENT POINTER
04FE'  EE                       PUSH    HL
04F9'  CC 0AAE'                 CALL    GETSTKG       ; GET STRING OF PAARMETERS
04FC'  IE 0E                    J6      H1,EXCH01     ; FOUND STRING
04FE'  E3                       POP     HL            ; RESTCRE CH_ADD AND AETUAN
04FF'  22 SCE0                  LO      (CH_ADD),HL   ; EARDA (NO STAING OR NULL STRING)
0502'  CE                       RET                   ; (BC CONTAINS STATUS)
                        ;
0501'  22 9C90          EXCH01  LO      (CM_ADD),HL   ; ADDRESS OF STAING
0506'  CO 641E'                 CALL    GETNO         ; GET PAARMETEAS
0E09'  52 0EEE'                 LO      (COPTWK),A    ; SOUACE LINE NO.
050G'  CO 041E'                 GALL    GETNO
050F'  52 03EA'                 LO      (COPYWK+2),A  ; DEST. LINE NO.
0E32'  GC 043E'                 CALL    GETNO         ;
0515'  52 0329'                 LO      (COPYWK+3),A  ; SOUACE OF (3 OA 2)
051E'  G0 041E'                 GALL    GETNO
0E1E'  12 03EE'                 LO      (COPYWR+3),A  ; DEST. OF
011E'  C0 641E'                 CALL    GETNO         ; NO. OF LINES
0521'  E1                       POP     HL            ; EESTORE CH_ADD
0122'  22 SCE0                  LO      (CH_ADD),HL   ;
0529'  00 4E 0EEE'              LO      BG,(COPYWR)   ; LOAD AEGS.
0529'  0C 50 03EA'              LO      DE,(GOPTWK+2)
                        ;                             ;
                        ;
0120'                   EACHSC:                       ; AOUTINE TO EXCHANGE LINES IN THE
                        ;                             ; DISPLAY FILE(S).
                        ;                             ;
                        ;                             ; INPUTS:  B=OF FOA SOUACE 1 (1 OA 2)
                        ;                             ;          G=LINE NO. FOX SOURCE 1
                        ;                             ;          O=OF FOA SOURCE 2 (1 OA 2)
                        ;                             ;          B=LINE NO. FOR SOUKCE 2
                        ;                             ;          A=COUNT (# OF LINES TO BE
                        ;                             ;            EXCHANGED
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
0920'  A7                       AND     A             ; TEST VALS COUNT
012E'  CA 00C4'                 JP      Z,SNVPAR
0131'  FE 39                    CP      Z5
0535'  02 0EC4'                 JP      NC,SNVPAR
05EA'  F5                       PUSH    AF            ; SAVE LINE COUNT
051T'  05                       PUSH    DE            ; SAVE SOUACE 2 INFO.
0130'  50                       LO      0,0           ; DP
0559'  43                       LO      0,G           ; LINE NC
055A'  CO G5AE'                 GALL    GTADAS        ; GET OF ADAS.FOR SOUACE I
0550'  0A 00C4'                 JP      G,SNVPAA      ; ERAOA DETECTED
0540'  01                       POP     DE            ; SOURCE 2 INFO.
0543'  E5                       PUSH    HL            ; SAVE SOURCE I ADAS.
0542'  41                       LO      0,E           ; LINE NO.
0545'  CC 0EAE'                 CALL    GTADES        ; GET OF ADAS. FOA SOLACE 2
0546'  0A 00C4'                 JP      C,SNVPR6
0549'  01                       POP     DE            ; ADDEESSES IN DE AND HL
054A'  F3                       POP     AF            ; LINE COUNT IN A
0546'  05                       PUSH    HL            ; SOURCE 2 STAATING ADAS. FOA EXIT
054G'  05                       PUSH    DE            ; SOUACE I STRATING ACAS. FOR EXIT
0140'  F5                       PUSH    AF            ; OAIG.LINE COUNT FGA EAIT
014E'  F5                       PUSH    AF            ; WOARING LINE CGUNT
014F'  06 00            EACH0   LO      E,0           ; SCAN COUNT
0591'  E5               EACH3   PUSH    HL            ; SAVE ADDRESSES AND SCAN COUNT
0592'  DE                       PUSH    DE
0591'  C5                       PUSH    BG
```

```
0554'  01 0020          1103            LD      BC,32           ; NO.OF BYTES IN SCAN ROW FOR
                        1104    ;                               ; A L2N0 ON TWO SCR00N
0557'  C0 0595'         1105            CALL    0ACMLP          ; 0XCHANG0 BYTES
055A'  C1               1106            POP     BC              ; SCAN COUNT IN B
0558'  01               1107            POP     D0
055C'  81               1108            POP     ML              ; ADDRESSES
055D'  10               1209            INC     D               ; ADJUST TO NEXT SCAN ROW
0550'  24               1110            INC     H
055P'  10 F0            1111            DJN2    0ACM1           ; DO NEXT ROW
0561'  P1               1112            POP     AP              ; L2N0 COUNT IN A
0562'  15               1115            D0C     D               ; ADJUST ADDRESSES BACK 1 SCAN ROW
0565'  25               1114            D0C     H
0564'  50               1115            D0C     A
0565'  20 22            1116            JR      2,0XCHX7        ; DONE
0567'  P5               1117            PUSH    AP              ; REMAINING L1N0 COUNT
0568'  01 0020          1218            LD      BC,32           ; ADJUST ADDRESSES TO START
0568'  05               1119            ADD     ML,BC           ; OF NEXT L1N0
056C'  5E 70            1120            LD      A,75H
0560'  44               1121            AND     H
056P'  67               1122            LD      H,A
0570'  08               1125            EX      D0,ML
0571'  09               1124            ADD     ML,BC
0572'  1E 70            1125            LD      A,70H
0574'  44               1126            AND     H
0575'  67               1127            LD      H,A
0576'  08               1128            OR      D0,ML
0577'  10 06            1129            JB      EACN0           ; DO NEXT L1N0
0579'  P1               1110    0XCHX7  POP     AP              ; A=ORIG. L1N0 COUNT
057A'  6F               1151            LD      L,A             ; L1N0 COUNT
0570'  26 00            1152            LD      H,0
057D'  29               1155            ADD     ML,ML           ; * OF ATTRIBUTE BYTES TO EXCHANGE
057E'  29               1154            ADD     ML,ML
057P'  29               1155            ADD     ML,ML
0580'  29               1156            ADD     ML,ML
0581'  29               1157            ADD     ML,ML           ; (524N0. OF L1N05)
0582'  44               1158            LD      B,H
0585'  40               1159            LD      C,L             ; COUNT TO BC
0584'  E1               1140            POP     ML              ; SOURCE 1 ADRS.
0585'  CD 0692'         1141            CALL    CALCATT         ; GET ADRS. OF ATTRIBUTE FILE
0588'  E8               1142            EX      DE,ML           ; SOURCE 1 ATTRIBUTE ADRS. IN DE
0589'  E1               1145            POP     ML              ; SOURCE 2 ADRS.
058A'  CD 0692'         1144            CALL    CALCATT         ; SOURCE 2 ATTRIBUTE ADRS. IN ML
058D'  CD 0595'         1145            CALL    EXCHLP          ; EXCHANGE ATTRIBUTE BYTES
0590'  C5 00RP'         1146            JP      GOODRET         ; RETURN A=0
                        1147    ;
                        1148    ;
                        1149    ;                               ; 0XCHANG0 LOOP
                        1150    ;
                        1151    ;                               ; EXCHANGES DATA BETWEEN TWO
                        1152    ;                               ; MEMORY AREAS.  ADDRESSES IN
                        1155    ;                               ; DE AND ML, BYTE COUNT IN BC
                        1156    ;
0595'  C5               1155    EXCHLP  PUSH    BC              ; SAVE COUNT
0594'  7E               1156            LD      A,(ML)          ; DATA FROM SOURCE 2
0595'  4P               1157            LD      C,A             ; TO WRG.REG.
0596'  1A               1158            LD      A,(DE)          ; DATA FROM SOURCE 1
0597'  A9               1159            XOR     C               ; EXCHANGE VIA XOR
057R'  P5               1160            PUSH    AP              ; SAVE INTERMEDIATE RESULT
0559'  A9               1161            XOR     C               ; A NOW CONTAINS DATA FOR SOURCE 2
059A'  4F               1162            LD      C,A             ; C=NEW DATA FOR SOURCE 2
0598'  P1               1145            POP     AP              ; INTERMEDIATE RESULT
059C'  A9               1164            XOR     C               ; A=NEW DATA FOR SOURCE 1
0590'  12               1165            LD      (DE),A          ; WRITE NEW DATA
0598'  71               1166            LD      (ML),C
059P'  15               1167            INC     D0              ; ADJUST TO NEXT BYTE
05A0'  25               1168            INC     ML
05A1'  C1               1169            POP     BC              ;
05A2'  0R               1170            DEC     BC              ; DECREMENT COUNT
05A5'  TR               1171            LD      A,R             ; TEST IF DONE
05A4'  B1               1172            OR      C
05A5'  20 EC            1175            JR      N2,ERCHLP
05A7'  C9               1174            RET
                        1175
                        1176    ;
                        1177    ;                               ; SUB-RTN. TO GET
                        1178    ;                               ; OP ADRS. OF LINE IN B (0-25)
                        1179    ;                               ; FOR OP IN 0 (1 OR 2)
                        1180    ;                               ; RETURNS CARRY IF INVALID PARAMETER
05A8'  78               1181    GTADRS  LD      A,B
05A9'  P8 18            1182            CP      24              ; TEST VALID L1N0 NO.
05AB'  50 1E            1185            JR      NC,GADRR
05AD'  0P               1184            RRCA
05AE'  0P               1188            RRCA
05AP'  0P               1186            RRCA
0580'  06 E0            1187            AND     0E0H
0582'  6P               1188            LD      L,A
0585'  7R               1189            LD      A,B
0584'  E6 18            1190            AND     18H
0586'  P6 40            1191            OR      40H
0558'  67               1192            LD      H,A
0589'  7A               1195            LD      A,D
058A'  A7               1194            AND     A
058B'  28 0E            1195            JR      1,GAERR         ; ERROR IF NOT 1 OR 2
058D'  PE 03            1196            CP      3
058P'  50 04            1197            JR      NC,GAERR
05C1'  CB 47            1198            BIT     0,A             ; TEST WHICH OP
05C5'  20 04            1199            JB      N2,GTAD01       ; OP1
05C5'  7C               1200            LD      A,H
05C6'  P6 20            1201            OR      20H             ; OP2
05C8'  67               1202            LD      H,A
05C9'  A7               1205    GTAD01  AND     A               ; RETURN NO CARRY
05CA'  C9               1204            RET
                        1205    ;
05CB'  ST               1206    GAERR   SCP                     ; SET CARRY
05CC'  C9               1207            RET
                        2208    ;
```

```
                      SUETTL  INTERNAL SUB-RTNS.
                 ;
                 ;
                 ;
                                            ; INPUT:  VIDEO MODE IN A
                                            ; OUTPUT: M/W SETTING IN R
                                            ;         VIOMOD SETTING IN C
                                            ; RETURNS NZ STATUS IF MODE INVALID
                                            ;
                                            ;  VALID VALUES:  INPUT     M/W     VIOMOD
                                            ;                 -----     ---     ------
                                            ;   **DISPLAY FILE ACTIVE AT SCREEN
                                            ;
                                            ;    DF1 ONLY        0        0         0
                                            ;   DF1&L DF2       80        0        50
                                            ;   DF1 & DF2*       1        1        01
                                            ;  HIGH RES.GR.      2        2         2
                                            ;  64/80-COLUMN  06 - 3E  06 - 3E   4R - 7E
05C0'  4F         GETVAL  LD    C,A
05CE'  C8 47              BIT   0,A
05D0'  20 16              JR    NZ,TST01
05D2'  FE 80              CP    80H
05D4'  28 16              JR    Z,SET0
05D6'  C8 7F              RIT   7,A
05D8'  C0                 RET   NZ
05D9'  47                 LD    B,A
05DA'  FE 02              CP    2
05DC'  C8                 RET   Z
05D0'  E4 C6              AND   0C8H
05DF'  EE 06              XOR   A
05E1'  C0                 RET   NZ
05E2'  3E 40              LD    A,40H
05E4'  B1                 OR    C
05E6'  4F                 LD    C,A
05E6'  AF                 XOR   A
05E7'  C9                 RET
05E8'  FE 01      TST01   CP    1
05EA'  C0                 RET   NZ
05E8'  06 01              LD    E,1
05E0'  0E 81              LD    C,R1H
05EF'  C9                 RET
05F0'  AF         SET0    ADR   A
05F1'  47                 LD    B,A
05F2'  C9                 RET
                 ;
                 ;
                                            ; ROUTINES TO ENABLE CHUNK 0 IN ROM BAT.
                                            ; FOR ACCESS TO CHNGVID ROUTINE
05F3'  F5         ENBLBAT PUSH  AF
05F4'  C8 FF              IN    A,(MREXPT)
05F6'  F6 80              OR    80H         ; SELECT ROM BAT.
05F8'  D3 FF              OUT   (MREXPT),A
05FA'  3E 05              LD    A,3
05FC'  D3 F4              OUT   (ORWSPT),A
05FE'  F1                 POP   AF
05FF'  C9                 RET
                 ;
0600'  F5         ENBLNOME  PUSH  AF
0601'  AF                   XOR   A
0602'  D3 F4                OUT   (ORWSPT),A
0604'  DB FF                IN    A,(MREXPT)
0606'  E6 3F                AND   03FH
0608'  D3 FF                OUT   (MREXPT),A
060A'  F1                   POP   AF
060B'  C9                   RET
                 ;
                                            ; TEST PARAMETER VALIDITY
                                            ; IF NOT VALID, DISCARDS RETURN
                                            ; AND EXITS VIA INVPAR
060C'  3E 37      TSTPAR  LD    A,23        ; TEST LINE 323
060E'  B8                 CP    B
060F'  30 04              JR    NC,TSTPR1
0611'  F1         PAREAR  POP   AF
0612'  C3 00C4'           JP    INVPAR
0615'  3A 0032'   TSTPR1  LD    A,(LINLEN)
0618'  3D                 DEC   A
0619'  B9                 CP    C           ; TEST COL. 331
061A'  38 FE              JR    C,PARERR
061C'  C9                 RET
                 ;                          ; CONVERT LINE/COL. FROM USER TO
                                            ; INTERNAL FORMAT AND VICE VERSA
                                            ; USER FORMAT:  LINES 0-21
                                            ;               COL.0-31
                                            ; INTERNAL FORMAT: LINES 24-1
                                            ;                  COL.  31-2
                                            ;                  (COL.1 END OF LINE)
061D'  3A 0023'   CONVPM  LD    A,(LINLEN)
0620'  3C                 INC   A
0621'  91                 SUB   C
0622'  4F                 LD    C,A
0623'  3E 1E              LD    A,SCRS2
0625'  90                 SUB   B
0626'  47                 LD    B,A
0627'  C9                 RET
                 ;
                 ;
                 ;
062E'             UPDPOSN                   ; ENTER HERE WITH BC=LINE/COL.IN
                                            ; INTERNAL FORMAT
0628'  CD 0620'           CALL  CALCPOS     ; CALCULATE POSITION
062B'  18 20              JR    STPOSN      ; STORE UPDATED POSITION AND RETURN
                 ;
                 ;
                 ;                          ; ROUTINE TO CALCULATE POSITION IN DF
                                            ; RETURNS DF ADRS. IN HL.  PRESERVES
                                            ; LINE/COLUMN POSITION IN BC
                 ;
062D'  CD 063A'   CALCPOS CALL  LARU        ; GET DISPLAY FILE ADRS.
0630'  3A 0033'           LD    A,(LINLEN)  ; ADD COLUMN OFFSET
0633'  3C                 INC   A
0634'  91                 SUB   C
0635'  5F                 LD    E,A
0636'  16 00              LD    D,0
0638'  19                 ADD   HL,DE
0639'  C9                 RET
```

```
                        1325  ;
                        1326  ;
063A*                   1327  LNBUF                         ; GET DISPLAY FILE ADDRESS
                        1328                                ; FOR START OF LINE IN B
0636* 3E 18             1329          LD   A,SCRS2
063C* 90                1330          SUB  B
063D* 57                1331          LD   D,A
063E* 0F                1332          RRCA
063F* 0F                1333          RRCA
0640* 0F                1334          RRCA
0641* E6 E0             1335          AND  0E0H
0643* 6F                1336          LD   L,A
0644* 7A                1337          LD   A,D
0645* E6 18             1338          AND  18H
0647* F6 40             1339          OR   40H
0649* 67                1340          LD   H,A
064A* 3A SCC2           1341          LD   A,(VIDMOD)
064D* CB 47             1342          BIT  0,A                ; TEST IF OP1
064F* C8                1343          RET  Z
0650* 3E 20             1344          LD   A,20H              ; SET TO OP2
0652* B4                1345          OR   H
0653* 67                1346          LD   H,A
0654* C9                1347          RET
                        1348
0655*                   1349  STPOSN:                        ; STORE CURSOR POSITION
0655* ED 43 0034*       1350          LD   (CURPOS),BC
0659* 22 0036*          1351          LD   (OPADRS),HL
065C* C9                1352          RET
                        1353  ;
065D*                   1354  LDPOSN:                        ; LOAD CURSOR POSITION
065D* ED 4B 0034*       1355          LD   BC,(CURPOS)
0661* 2A 0036*          1356          LD   HL,(OPADRS)
0664* C9                1357          RET
                        1358  ;
                        1359  ;
                        1360  ;
                        1361  ;
0665*                   1362  UPD6TT:                        ; UPD6TE 6TTRIBUTE BYTE FOR CHARACTER
                        1363                                ; JUST PRINTED
                        1364                                ; HL = ANY SCAN OF CHAR,IN OF
                        1365  ;
0665* CD 069I*          1366          CALL   C6LCATT        ; ADRS. OF ATTRIBUTE BYTE TO HL
066B* 3A 0039*          1367          LD     A,(ATTB9T)     ;
066E* 5F                1368          LD     E,A            ; CURRENT ATTB.VALUE TO E
066C* 3A 003F*          1369          LD     A,(ATTMSA)     ;
066F* 57                1370          LD     D,A            ; MASK TO D
0670* 3A 0038*          1371          LD     A,(MASA8)
0673* 47                1372          LD     B,A            ; FLAGS TO B
0674* 7E                1373          LD     A,(HL)         ; BYTE FROM 6TTRIBUTE FILE
0675* A8                1374          XOR    E
0676* A2                1375          AND    D
0677* A8                1376          XOR    E              ; NEW ATTB.FOR OF OLD FOR 1
0678* CB 60             1377          BIT    4,B            ; SET INK COMPLEMENT OF PAPERT
067A* 28 08             1378          JR     I,UPDAT1       ;
067C* E6 F8             1379          AND    0F8H           ; SET INK TO BLACK
067E* CB 6F             1380          BIT    5,A            ; IS PAPER 4-7
0680* 20 01             1381          JR     NZ,UPD6T1      ; YES - USE BLACK INK
0682* F6 07             1382          OR     7              ; NO -  USE WHITE INK
0684* CB 70             1383  UPDAT1   BIT    6,B            ; SET PAPER COMPLEMENT OF INAT
0686* 28 08             1384          JR     Z,UPDAT2       ;
0688* E6 C7             1385          AND    0C7H           ; SET PAPER TO BLACK
068A* CB 57             1386          BIT    I,A            ; IS INK 4-7
068C* 20 02             1387          JR     NE,UPDATI      ; YES - USE BLACK PAPER
068E* F6 38             1388          OR     38H            ; NO -  USE WHITE PAPER
0690* 77                1389  UPDAT2  LD     (HL),A         ; TO ATTRIBUTE FILE
0691* C9                1390          RET
                        1391  ;
                        1392  ;                             ; SUB-RTN. TO CALCULATE ADAS. OF
                        1393  ;                             ; ATTRIBUTE BYTE FROM ADAS. OF
                        1394  ;                             ; ANY SCAN ROW IN OF
                        1395  ;
0691* 7C                1396  CALC6TT LD     A,H            ; UPPER BYTE OF ADDRESS
0693* 0F                1397          RRC6
0694* 0F                1398          RRCA
0695* 0F                1399          RRCA
0696* E6 03             1400          AND    03H
0698* F6 5E             1401          OR     58H
069A* CB 6C             1402          BIT    5,H            ; TEST WHICH OF
069C* 28 01             1403          JR     Z,CALC61
069E* F6 I0             1404          OR     I0H            ; OP1
06A0* 67                1405  CALC61  LD     H,A            ; ADAS. OF ATTRIBUTE IN OP2
06A1* C9                1406          RET
                        1407  ;
06AI*                   1408  LOATTR                        ; LOAD INTERNAL ATTRIBUTE VARIABLES
                        1409                                ; FROM SYSTEM VARIABLES
06AI* F5                1410          PUSH   AF             ; SAVE A
06A3* 3A 3C80           1411          LD     A,(ATTB_P)     ;
06A4* 3I 0039*          1412          LD     (ATTBTT),A     ;
06A9* 3A 5CB8           1413          LD     A,(MASA_P)     ;
06AC* 32 003F*          1414          LD     (ATTMSA),A     ;
06AF* 3A 5C91           1415          LD     A,(P_FLAG)     ;
06BI* 0F                1416          RRCA                  ; SHIFT TWO BITS TO EVEN
0683* 32 0038*          1417          LD     (MASA8),A      ;
0686* F1                1418          POP    AF
0687* C9                1419          RET
                        1420  ;
0688* 3A SCC3           1421  GETSTRG LD    A,(PARAM3)      ; GET STRING ID
0688* E6 1F             1422          AND    1FH            ; MASK OFF UPPER BITS
068D* F6 40             1423          OR     40H            ; STRING VARIABLE IDENTIFIER
068F* 57                1424          LD     D,A            ; SAVE IN D
06C0* 2A 5C4B           1425          LD     HL,(VARS)      ; FIND START
06C3* 7E                1426  GTSTR1  LD     A,(HL)         ;
06C4* E6 7F             1427          AND    07FH           ; TEST IF END OF VARS AREA
06C6* 28 13             1428          JR     I,NOSTRG       ; NO FIND - RETURN BC=2
06C8* BA                1429          CP     D              ; TEST IF MATCH
06C9* 28 0E             1430          JR     I,GTSTRI       ; FOUND STRING
06CB* 05                1431          PUSH   DE             ; SAVE STRING ID
06CC* CC 1720           1432          CALL   RECLEN         ; RETURNS ADRS. OF NEXT VAR. IN DE
06CF* EB                1433          EX     DE,HL          ; ADRS. TO HL
0600* 01                1434          POP    DE             ; RESTORE STRING ID
0601* 18 F0             1435          JR     GTSTR1         ; LOOK AGAIN
0603* 23                1436  GTSTR2  INC    HL             ; GET LENGTH
0604* 4E                1437          LD     C,(HL)
```

```
06D5'  23                   INC   HL
06D6'  46                   LD    B,(HL)
06D7'  23                   INC   HL                ; first text code
06D8'  78                   LD    A,B               ; test if null string
06D9'  B1                   OR    C
06DA'  C9                   RET                     ; return (Z if null string/NZ if not)
                    ;
06DB'  0E 02        NOTFND  LD    C,2
06DD'  06 00                LD    B,0
06DF'  C9                   RET                     ; (Z and BC=2 for no find)
                    ;
                    ;
                            SUBTTL GRAPHICS CHRS.SET
                    ;
                    ;                               ;PIXEL PATTERNS FOR STD.GRAPHICS SET
                    ;                               ; 0=BACKGROUND(PAPER) 1=FOREGROUND(INK)
                    ;
06E0'  00           GRPST   defb 00000000b          ; code 80      graphics space
06E1'  00                   defb 00000000b
06E2'  00                   defb 00000000b
06E3'  00                   defb 00000000b
06E4'  00                   defb 00000000b
06E5'  00                   defb 00000000b
06E6'  00                   defb 00000000b
06E7'  00                   defb 00000000b
                    ;
06E8'  0F                   defb 00001111b          ; code 81      graphics
06E9'  0F                   defb 00001111b
06EA'  0F                   defb 00001111b
06EB'  0F                   defb 00001111b
06EC'  00                   defb 00000000b
06ED'  00                   defb 00000000b
06EE'  00                   defb 00000000b
06EF'  00                   defb 00000000b
                    ;

06F0'  F0                   defb 11110000b          ; code 82      graphics
06F1'  F0                   defb 11110000b
06F2'  F0                   defb 11110000b
06F3'  F0                   defb 11110000b
06F4'  00                   defb 00000000b
06F5'  00                   defb 00000000b
06F6'  00                   defb 00000000b
06F7'  00                   defb 00000000b

06F8'  FF                   defb 11111111b          ; code 83      graphics
06F9'  FF                   defb 11111111b
06FA'  FF                   defb 11111111b
06FB'  FF                   defb 11111111b
06FC'  00                   defb 00000000b
06FD'  00                   defb 00000000b
06FE'  00                   defb 00000000b
06FF'  00                   defb 00000000b

                    ;
0700'  00                   defb 00000000b          ; code 84      graphics
0701'  00                   defb 00000000b
0702'  00                   defb 00000000b
0703'  00                   defb 00000000b
0704'  0F                   defb 00001111b
0705'  0F                   defb 00001111b
0706'  0F                   defb 00001111b
0707'  0F                   defb 00001111b
                    ;
0708'  0F                   defb 00001111b          ; code 85      graphics
0709'  0F                   defb 00001111b
070A'  0F                   defb 00001111b
070B'  0F                   defb 00001111b
070C'  0F                   defb 00001111b
070D'  0F                   defb 00001111b
070E'  0F                   defb 00001111b
070F'  0F                   defb 00001111b


0710'  F0                   defb 11110000b          ; code 86      graphics
0711'  F0                   defb 11110000b
0712'  F0                   defb 11110000b
0713'  F0                   defb 11110000b
0714'  0F                   defb 00001111b
0715'  0F                   defb 00001111b
0716'  0F                   defb 00001111b
0717'  0F                   defb 00001111b
                    ;
0718'  FF                   defb 11111111b          ;code 87       graphics
0719'  FF                   defb 11111111b
071A'  FF                   defb 11111111b
071B'  FF                   defb 11111111b
071C'  0F                   defb 00001111b
071D'  0F                   defb 00001111b
071E'  0F                   defb 00001111b
071F'  0F                   defb 00001111b
                    ;
0720'  00                   defb 00000000b          ; code 88      graphics
0721'  00                   defb 00000000b
0722'  00                   defb 00000000b
0723'  00                   defb 00000000b
0724'  F0                   defb 11110000b
0725'  F0                   defb 11110000b
0726'  F0                   defb 11110000b
0727'  F0                   defb 11110000b
                    ;
0728'  0F                   defb 00001111b          ; code 89      graphics
0729'  0F                   defb 00001111b
072A'  0F                   defb 00001111b
072B'  0F                   defb 00001111b
072C'  F0                   defb 11110000b
072D'  F0                   defb 11110000b
072E'  F0                   defb 11110000b
072F'  F0                   defb 11110000b
```

```
0730'  F0          ;
                        defb 11110000b          ; code 8A       graphics
0731'  F0               defb 11110000b
0732'  F0               defb 11110000b
0733'  F0               defb 11110000b
0734'  F0               defb 11110000b
0735'  F0               defb 11110000b
0736'  F0               defb 11110000b
0737'  F0               defb 11110000b
                    ;
0738'  FF               defb 11111111b          ; code 8B       graphics
0739'  FF               defb 11111111b
073A'  FF               defb 11111111b
073B'  FF               defb 11111111b
073C'  F0               defb 11110000b
073D'  F0               defb 11110000b
073E'  F0               defb 11110000b
073F'  F0               defb 11110000b
                    ;
0740'  00               defb 00000000b          ; code 8C       graphics
0741'  00               defb 00000000b
0742'  00               defb 00000000b
0743'  00               defb 00000000b
0744'  FF               defb 11111111b
0745'  FF               defb 11111111b
0746'  FF               defb 11111111b
0747'  FF               defb 11111111b
                    ;
0748'  0F               defb 00001111b          ; code 8D       graphics
0749'  0F               defb 00001111b
074A'  0F               defb 00001111b
074B'  0F               defb 00001111b
074C'  FF               defb 11111111b
074D'  FF               defb 11111111b
074E'  FF               defb 11111111b
074F'  FF               defb 11111111b
                    ;
0750'  F0               defb 11110000b          ;code 8E        graphics
0751'  F0               defb 11110000b
0752'  F0               defb 11110000b
0753'  F0               defb 11110000b
0754'  FF               defb 11111111b
0755'  FF               defb 11111111b
0756'  FF               defb 11111111b
0757'  FF               defb 11111111b
                    ;
0758'  FF               defb 11111111b          ;code 8F        graphics
0759'  FF               defb 11111111b
075A'  FF               defb 11111111b
075B'  FF               defb 11111111b
075C'  FF               defb 11111111b
075D'  FF               defb 11111111b
075E'  FF               defb 11111111b
075F'  FF               defb 11111111b

                        END
```

```
ADL - B3C 004 DUAL SCREEN MODE SUPPORT  Cd180/11 version 10.58.14       16-Mar-84   12:47:30
GRAPHICS CHAR.SET   1CD3.SRC

B        Reserved    dTTBYT   0059       BTYNSK   003F       dTTRBP   5C80       B        Reserved
BD7LN    002D        C        Reserved   CBLCdT   0B92       CBLCd1   08d0       CdLCPD   0820
CNNGY1   08BE        CHP3ET   5C0D       CHTBL    002F       CN9dD0   5C50       CLINIT   1800
CLRCTL   000C        CLP3B0   0104       CLdSCB   000E IN    CLR3CN   01DE IN    CLdSC0   01EE
CLRSC1   0202        CLRSC2   0200       CLdSC3   0211       CLdSC4   0214       CLR3C5   021F
CLRSC7   0254        CLd3XT   0250       CONVPN   0B10       CDDRDS   5C7D       CDPEdR   045E
CDPEd1   043B        COPY3B   0040 IN    CDPYSC   045D IN    CDP730   050d       CDPYS1   045D
CDP7S2   0477        COPY33   0478       COP7S4   047E       CDPYS5   04DB       CDPYS8   0489
CDP7S7   04F0        COPYNK   05E8       COP7XT   04C2       CDP701   05EC       CDP741   04ED
CDP742   04AF        CUdPOS   0054       D        Reserved   DdTBE    0000       DESTT    F7CE
DPBDRS   005B        DKM1P7   00F4       DR2VES   0E40       DTBDP1   03CE       DTdDP2   05D2
E        Reserved    ENBLEX   05F5       ENBLMD   0B0D       ERRRET   00C1       EXCHLP   0595
EXCHSE   0045 IN     EXCH1C   0520 IN    EXCNS0   04F5       EXCNXT   0579       EXCH0    054F
EXCN01   0505        EXCH1    0551       2XTNRN   0585       F2X      97C0       PIXTBL   1A00
FP2B     5195        FRFV     038d       GdERR    05C8       GE7BE0   020C       GE7d7B   001E 2N
GETBTT   02ED IN     GETCB0   02F1       G2TCTL   0815       G2TCUE   0021 IN    G2YCUd   02F1 2N
GETC1    0505        GE7ND    041E       GE7N1    042E       G2YN2    042F       GETSTR   06E0
GETVBL   05C0        GOOCRE   008F       GRBLd    01d9       GFPNST   0580       GRFST    08E0
GRTEL    0031        GTBDRS   05B8       GTBDR1   05C9       GTCHdR   0280 2N    GTCNB0   025C
GTCHRE   0018 IN     GTCN1    0271       GTCN2    0278       GTCN2S   02EE       GTCN3    E2ED
GTCHS1   0290        GTCH32   02A8       GTCN4    02dF       GTCNS    02CB       G7CNB    0209
G71N0X   005d        G7SVP1   0BC5       GTSTR2   0BD5       H        Reserved   Hd2XPT   00FF
2NINT    30F9        IN1D5L   015F       INSERT   12C0       2NVPBR   03CB       L        Reserved
LDdTTR   08d2        LOFC3H   085D       LINCOL   0097       LINLEN   0055       LNBU     065d
LDOF     0165        LG3P0    0188       LDDP1    01BE       M        Reserved   MdSKE    0058
MBSKBP   5CEE        MCVES1   0E40       NOS7P0   0B0E       PBRBNB   5CC5       PAR2dR   0811
PRBNT    5CE4        PSW      Reserved   PBPLBG   5051       dBHTDP   5C82       X2CL2N   1720
SCINI7   1701        SCPCTL   0028       SCRLB    0E2B IN    SCRLB0   0127       SCRLC7   002E
SCRLXT   01EB        1CRL0    0141       SCROLL   0128 IN    SC#31    0018       SETC80   E117
SE7CUB   0009 IN     3E7CUd   0115 IN    SETMDB   0325 IN    SE7M00   052d       SETN01   0554
SETND2   038d        SE7M03   055F       SE7ND0   0311 IN    SE7M21   E597       SE70     05F0
SP       Reserved    SPddE0   0011       SPddE1   0015       SPdR22   0015       3PdRES   0017
SPdRE4   003D        STBLK    0181       STBLd0   0185       STBLd1   018B       STdBND   5C85
STNDB0   050E        STPO3N   0855       STdGCT   005E       TESTB0   0151       T37PdR   080C
TSTPR1   0815        TST01    05E8       TVPULD   00C8       TVPUL1   00EB       UDG      5C7E
UPDBTE   05B2        UPDdTT   08B5       UPOBT1   0484       UPOdT2   049D       UP3PDS   E82E
VBRS     5C48        V1CHOD   5CC2       VIMOOE   0324       NRCHBR   0049 IN    MdCHdB   0081 IN
NRCHXT   E0BC        NRCH0    004B       NRCH01   004C       MdCN11   00B8       WdCH12   0075
NRCN1S   0077        WdCH14   008D       NRCH15   00EE       NdCH2    0094       NRCH3    089F
NRCNS    00B7        NRCH7    00E5       NRSTLP   00F5       NRSTRB   E904 2N    NRSTdG   00F2 2N
NRSTRB   E0EE        NdSTRS   0EFF       NRSTX1   8108       NRSTX2   8114

    No errors detected
```

```
Cross reference listing (HEBP version 4.71

Symbol     Refs (0 = definition  8 = write  <blank> = read)

6TTEYT     1570    557     1567    14120
4TTMIX     1640    1140    16140
6TT8_P      650    1411
60TLN      1550    260      7320
C6LC61    1401    14050
C6LC6T      425    546      677     1001    1004    1545    5546
                   1566    15960
C6LCP0      569    676     1505    15170
CHNGV1      520    772
CH0S0T      690    147      742
CHT6L      1470    201      550     7450
CH_600      610    855     0400    0460    0570    862     075o
                   0600    1046    10450   10460   10590
CLSNST      460    737
CL8CTL     1040    481      7560
CL8550      107    4800
CL8SC0      461    493     4970
CL8SC1     5100    567
CLR1C2     5190
CL8SC3     5220    576
CL8SC4     5240    543
CL8SC5     5330
CL8SCT      518    5700
CL6SC8       40    1070
CL8SCN       50    4030
CL818T      565    560P
CONVPH      537    500      675     606     12960
C000OS      640    7020
COP8E1     0790
CO0800      067    070      0700
COPT01      636    0460
COPT41      545    5470
COPT42      972    0740
COPTS0      166    0340
COPT11     0060    169
COPTS2     5310
COPT1S     5540    1025
COPTS4     5360    956
COPTS5      516    10060
COPTS6    10200    1025
COPYST      927    1318    10260
COPTS8       65    1660
COPYSC       43    960     8850
COPYUK     0430    0400    0530    0520    0540    958     059
                   10508   10520   10540   10560   1060    1065
COPYXT      062    0900
CU000S     1520    485     13560   1355
06T40       550    174     7400
08577       750     76
004005     1540    13518   1356
0KM18T      500    12610   12600
081V01      720     73      74
0T4001     0160
0T4002      012    9210
6N8L8X      765    12500
8N6LM0      775    12670
E08067     2500    263      206     706
8XCH6     10990    1129
0XCH61     1045    10460
8XCH1     11000    1111
88CHLP     1105    1145    11050   1173
8XCH16      167    10370
8XCH18       44    1670
EXCHSC       44    10640
8XCHXT     1116    11300
8XTNEM      759    763     7050
PSX         760
0S8T6L      700
PP26        550    072
88PY        718    726     8060
G6886      1105    1195    1187    12660
G8T600      115    6710
06T6T        41    1150
G6TATT       39    6740
GETC1       600    695     6070
G8TC86      116    4830
G8TCTL     1120    504      671     7510    7520
G8TCU8       45    1160
88TCU8       39    6040
88TN1      0710
68TN2       065    0740
G8TN0       047    049      551     953     055     0620    076
                   1048    1051    1053    1051    1057
G6TST8      292    037     1042    14210
06TV6L      766    12200
G00008     2500    339      162     494     726     704     015
                   622     1006    1146
G88LX       385    4450
G8PM1T      530    150      744
G8P1T        50    14860
G8T8L      1500    190      653     7450
GT6081     1100    12030
8T6085      002    097     1006    1001    11010
GTCH1      5050    656      661
GTCH2      1960    646
GTCH23      605    6000
GTCH3       601    6120
GTCH31     6140    620
GTCH32      613    635     6370
0TCH4       607    610      619     6410
6TCH5       602    6570
GTCH6       658    6640
GTCH48       38    5060
8TCH80      114    5810
8TCH88       41    1140
GTINOX     1590    5530     601     638     650
```

GTSTR1	1A1A#	1455					
GTSTR2	1430	143A#					
NRBOPT	57#	1259	1261R	1270	1272#		
ININT	54#	871					
INSOOL	5R6#						
INSERT	73#	754					
INVPAR	186	18R	262#	555	358	360	A87
		490	A92	721	809	881	868
		598	895	898	1079	1081	1087
		1092	1203				
LOATTR	177	500	497	1400#			
LOPOSN	181	1554#					
LINCOL	180#	533	97R	749R	750R		
LINLEN	151#	213	66R	362	687	736#	777
		128A	129R	1318			
LNBU	371	505	1317	1327R			
LOOP	309#	A49					
LOOPO	351#	408					
LOOP1	59R#						
MASOR	155#	227	1371	1A17R			
MASO_P	6R#	1415					
MOVRS1	74#	75	755				
NCSTOG	1420	144##					
PAOAMB	71#	1421					
PARERR	1282#	1287					
PRANT	69#						
P_FLAG	R7#	1413					
RAMTOP	R0#	760					
RBCLEN	R3#	1432					
SCINIT	47#	729					
SCOCTL	119#	202	348	730R			
SCRLO	20A	3R1	364#				
SCOLB	40	122#					
SCRLOO	122	34R#					
SCRLCT	140#	272	27R#	261R	73A#		
SCOLOT	A12	415#	470				
SCROLL	58	553#					
SCRS1	4A#	2A5	363	495	700	1300	5329
SETO	1232	1251#					
SETCRO	103	351#					
SETCUR	40	103#					
SETCUR	58	535#					
SETM21	770	772#					
SETMOD	712	714	716	722#			
SETMO1	729R						
SETMO2	72A	764#					
SETMO3	776#	818	824				
SETMO8	42	115#					
SETMOD	42	708#					
SPARRO	108#						
SPARE1	109#						
SPARE2	110#						
SPA003	111#						
SPAREA	163#						
STOL#	377	451#					
STOL#0	434#	A48					
STOL#1	455#						
STOEND	62#	757					
STMORD	118	704#					
STPOSN	257	1310	1549#				
STRGCT	1R1#	322R	740R	7A16			
TESTAD	373#	414	A73				
TST01	1230	124R#					
TSTPAR	31A	587	R7A	1279#			
TSTPR1	12R1	1204#					
TVPUL1	27A	201#					
TVPUL2	222	263#					
UOG	63#	154	A59				
UPORT1	1378	1581	1303#				
UPDAT2	13R4	1387	1389#				
UPDATE	777#						
UPDATT	232	13A2#					
UPOPCS	270	13R	5R9	703	1307R		
VARS	60#	1423					
VIOMOD	70#	722	770R	80R	1341		
VIMODE	117#	706					
WRCMO	97	172#					
WRCMO1	181#	30A					
WRCM11	193	158#					
WRCM12	191	201#					
WRCM13	197	290	202R				
WRCM14	214	215#					
WRCM13	218	220#					
WRCM2	223#						
WRCM3	250	232#					
WRCM5	235#	249					
WRCM7	250#						
WRCM4R	38	176#					
WRCMR8	40	97R					
WRCM#T	257#						
WRSTLP	301#	315					
WRSTRG	98	292#					
WRSTRS	309#	320					
WRSTR8	40	9R#					
WRSTRG	38	300#					
WRST01	307	31R#					
WRST02	323R						

APPENDIX C-5

SPRITE GRAPHICS SUPPORT

Name: Sprite Graphics Support

Description: This component provides support for sprite graphics. A sprite is a graphical object which can be created and manipulated by a set of services to be described below. A sprite can be up to 256 x 256 characters in size, however at most 32 x 24 can be displayed. Each sprite is identified by a sprite-id, a number greater or equal to zero, and has the following properties:

definition - a pointer to a width x height character bitmap, where width and height are user defined. A 1 indicates foreground color, a 0 indicates background color.

position - the row, col location on the screen of the upper left corner of the sprite.

color - the screen attribute (foreground color). A value of 0 indicates black, 1 indicates blue, etc.

size - the width and height of the sprite (in characters).

The sprites graphics package assumes that the system variables area of memory is not used by an application. The sprite support services can be invoked from machine code and BASIC. There is a BASIC interface routine, which takes as its parameter a BASIC string variable. The BASIC interface and sprite services code is loaded at SPRCODE, where SPRCODE = DE00H. The name of this variable is specified by poking in the starting character at address SPRCODE-4. For example, if POKE SPRCODE-4, CODE "C" is executed, the variable C$ will be used to pass commands to the interface routine. This must be done before using any sprite services. The interface routine is invoked by LET <variable> = USR SPRCODE. The status code returned by the last sprite service executed is assigned to <variable>.

The BASIC interface routine causes the report "A: Invalid argument" to be produced whenever an illegal command or invalid argument to a command is found. If too few or too many arguments are given for some command, the report "B: Parameter error" is produced. If the command string variable cannot be found, the report "C: Nonsense in BASIC" is produced. If there is not enough memory for the number of sprites specified, the report "4: Out of memory" is produced (see InitSprites). There exists a variable called COMMAND, at address SPRCODE-9, which contains the number of the last sprite command to be executed. Thus if an error occurs in a sprite command, this variable can be PEEKed to find out which command in the command string caused the error. The first command in the command string is command zero.

The machine code routines all return status codes in the BC register pair. If the value in C is 0, then no error occurred. In this case B contains a value indicating further status information. If an error occurred, C will contain an odd value (thus bit 0 is set).

The BASIC interface string variable contains sprite graphics commands, separated by spaces. The commands have the following syntax:

"<command letter><parameter list>"

where <command letter> is a single upper or lower case letter identifying the command:

I = Init_Sprites
S = InitScreen
C = Create_Sprite
W = Set_Autowrap_Mode
P = Put_Sprite
E = Erase_Sprite
M = Move_Sprite
L = Change_Sprite_Location
A = Change_Sprite_Attribute
O = Overlap?
H = Horizontal_Scroll
V = Vertical_Scroll

and where <parameter list> is a list of numbers separated by commas. The string assigned to C$ can be a string expression, i.e., "<substr>"+STR$(<expr>)+"<substr>". The semantics of these commands will be described in detail below.

Name: Init_Sprites

Machine code interface:

Inputs: A = max-sprites, 0..255

Outputs: BC = 0000h - OK
= 0001h - not enough memory

Entry: SPRSVC = SPRCODE + 219H

BASIC interface:

Inputs: LET C$ = "I<max-sprites>"

Description: This service initializes the data structures used by the sprites services. Memory immediately below the sprites code is used for global variables and memory immediately below this is used to store sprite information. Each sprite takes up 8 bytes of data (not including its definition, which is stored elsewhere). Thus the total amount of memory used by the sprites package is:

8 bytes * max-sprites + 29 byte globals area + 2746 byte code area

If there is not enough memory between RAMTOP and the global variables area for the number of sprites specified, 0001h is returned. Thus, RAMTOP should be set to

SPRCODE - 29 - 8 max-sprites

before calling InitSprites. This service must be called before any of the others.

BASIC example: "I32" initializes memory for 32 sprites.

Name: Init_Screen

Machine code interface:

Inputs: A = screen_height, 0..24
D = background_color, 0..7
E = border_color, 0..7

Outputs: BC = 0000h - OK
= 0007h - illegal screenht
= 0005h - illegal color

Entry: SPRSVC + 200H

BASIC interface:

Inputs: LET C$ =
"S<screen_ht>,<background_color>,<border_color>"

Description: This service clears the top screen_ht lines of the screen and sets their color to the background_color. The bottom 24-screen_ht lines are cleared and set to the border_color. The top screen_ht lines are used for displaying sprites. The remaining lines can be used for text display. The border is set to the border_color.

BASIC example: "S22,5,1" clears the top 22 lines and sets their color to cyan, clears the bottom two lines and sets their color to blue, and sets the border color to blue.

- -

Name: Create_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..max-sprites - 1
B = width, 0..255
C = height, 0..255
D = color, 0..7
HL = definition address, 0..64K-1

Outputs: BC = 0000h - OK (sprite created)
= 0003h - illegal sprite-id

Entry: SPRSVC + 5FH

BASIC interface:

Inputs: LET C$ =
"C(sprite-id),(width),(height),(color),(addr)"

Description: This service creates a sprite with the specified width, height, color, and definition address. Initially, the position of the sprite is (0, 0). If there was already a sprite with the specified id, it is destroyed. In all further operations the new sprite is identified by its sprite-id.

BASIC example: "C0,2,2,2,32768" creates sprite 0 which is 2 x 2 characters in size, is red in color, and whose definition is located at 32768. The initial position of the sprite is (0, 0).

- -

Name: Set_Autowrap_Mode

Machine code interface:

Inputs: A = mode: 0 - off, not 0 - on

Outputs: BC = 0000h - OK

Entry: SPRSVC + 80H

BASIC interface:

Inputs: LET C$ = "W(mode)"

Description: This service turns the Autowrap mode on and off. When the Autowrap mode is on, all changes to a sprite's location are made so that the new location is on the screen (wraparound).

BASIC example: "W1" causes Autowrap mode to be turned on.

- -

Name: Put_Sprite

Machine code interface:

Inputs: A = sprite-id, 0..max-sprites - 1
D = row, 0..255
E = column, 0..255

Outputs: BC = 0000h - OK, nothing overwritten
0100h - OK, something overwritten
0003h - illegal sprite-id
D = row
E = column

Entry: SPRSVC + EEH

BASIC interface:

Inputs: LET C$ = "P(sprite-id),(row),(column)"

Description: This service writes a sprite on the screen. The row and column described above define the location at which the upper left corner of the sprite is written. If the Autowrap mode is set, the row and column values are modified to make sure that the sprite location is a legal screen location. The new row and column values are returned. If the Autowrap mode is not set, the position is not changed. However, only those parts of the sprite corresponding to legal screen positions will be displayed. If writing the sprite causes something on the screen to be overwritten, the appropriate code is returned.

BASIC example: "P0,11,0" puts sprite 0 at row 11, column 0. Anything already at that location is overwritten. If anything is overwritten, the interface routine returns 10h.

- -

- -

Name: Erase_Sprite

Machine code interface:

```
     Inputs:  A = sprite-id, 0..max-sprites - 1

     Outputs: BC = 0000h - OK (sprite erased)
                   0003h = illegal sprite-id

     Entry: SPRSVC + 198H
```

BASIC interface:

```
     Inputs:  LET C$ = "E<sprite-id>"
```

Description: This service erases a sprite. The screen becomes blank where the sprite used to be. The sprite still exists and can be written elsewhere.

BASIC example: "E0" erases sprite 0. The screen locations taken up by the sprite become blank.

- -

Name: Move_Sprite

Machine code interface:

```
     Inputs:  A = sprite-id, 0..max-sprites - 1
              D = relative vertical motion, -128..127
              E = relative horizontal motion, -128..127

     Outputs: BC = 0000h - OK, nothing overwritten
                   0100h - OK, something overwritten
                   0003h - illegal sprite-id
              D = row
              E = column

     Entry: SPRSVC + 20DH
```

BASIC interface:

```
     Inputs:  LET C$ = "M<sprite-id>,<vert>,<hor>"
```

Description: This service moves a sprite. The sprite is erased at its current location and written at a new location defined by:

```
     new row = old row + relative vertical motion
     new col = old col + relative horizontal motion
```

The location of the sprite is updated to reflect the motion. As with the "P" command, if the sprite overwrites anything, 10h is returned. If Autowrap mode is set, then the location is automatically wrapped to fit onto the screen (i.e., (24, 0) becomes (0, 0)). The new row and column values are returned. If Autowrap mode is not set, only those parts of the sprite which correspond to legal screen locations will be displayed.

BASIC example: "M0,2,0" changes the position of the sprite from (11, 0) to (13, 0). This operation is equivalent to "E0 P0,13,0".

- -

Name: Change_Sprite_Location

Machine code interface:

```
     Inputs:  A = sprite-id, 0..max-sprites - 1
              D = new row, 0..255
              E = new column, 0..255

     Outputs: BC = 0000h - OK (location changed)
                   0003h - illegal sprite-id

     Entry: SPRSVC + 237H
```

BASIC interface:

```
     Inputs:  LET C$ = "L<sprite-id>,<new-row>,<newcol>"
```

Description: This service updates the position property of a sprite in the same fashion as for Put_Sprite. The sprite is not updated on the screen until a Move_Sprite, Put_Sprite, or Erase_Sprite is executed.

BASIC example: "L0,13,0" will set the location of sprite 0 to (13, 0).

- -

- -

Name: Change_Sprite_Attribute

Machine code interface:

```
     Inputs:  A = sprite-id, 0..max-sprites - 1
              D = color, 0..7

     Outputs: BC = 0000h - OK (color changed)
                   0003h - illegal sprite-id
                   0005h - illegal color

     Entry: SPRSVC + 240H
```

BASIC interface:

```
     Inputs:  LET C$ = "A<sprite-id>,<color>"
```

Description: This service updates the color property of a sprite. The sprite on the screen is not updated until a Put_Sprite, Move_Sprite, or Erase_Sprite is executed.

BASIC example: "A0,2" causes the color property of sprite 0 to be set to red.

- -

Name: Overlap?

Machine code interface:

```
     Inputs:  D = sprite-id-1, 0..max-sprites - 1
              E = sprite-id-2, 0..max-sprites - 1

     Outputs: BC = 0000h - OK, no overlap
                   0200h - OK, overlap
                   0003h - illegal sprite-id

     Entry: SPRSVC + 26BH
```

BASIC interface:

```
     Inputs:  LET C$ = "D<sprite-id-1>,<sprite-id-2>"
```

Description: This service is used to detect if two sprites overlap.

BASIC example: "D0,1" causes the interface routine to return YES if sprites 0 and 1 overlap on the screen.

- -

Name: Vertical_Scroll

Machine code interface:

```
     Inputs:  A = direction: positive number - down,
                             negative number - up

     Outputs: BC = 0000h - OK

     Entry: SPRSVC + 558H
```

BASIC interface:

```
     Inputs:  LET C$ = "V<direction>"
```

Description: This service is used to scroll the entire screen in the vertical direction by one row. The direction of scroll is up if direction is less than 0, otherwise the scroll is down. The position property of all sprites is updated by one row to reflect the effect of the scroll. If Autowrap mode is not set, then 1 is added to the row value. If Autowrap mode is set, then 1 is added to the row value, and this new value is wrapped to fit on the screen.

BASIC example: "V1" causes the screen to scrolled down one row.

- -

- -

Name: Horizontal_Scroll

Machine code interface:

> Inputs: A = direction: positive number - right,
> negative number - left
>
> Outputs: BC = 0000h - OK
>
> Entry: SPRSVC + 38DH

BASIC interface:

> Inputs: LET C$ = "H<direction>"

Description: This service is uesd to scroll the entire screen in the horizontal direction by one column. The direction of scroll is to the left if direction is less than 0, otherwise the scroll is to the right. The position property of all sprites is updated by one column to reflect the effect of the scroll. If Autowrap mode is not set, then 1 is added to the column value. If Autowrap mode is set, then 1 is added to the column value, and this new value is wrapped to fit on the screen.

BASIC example: "H1" causes the screen to scrolled right one column.

- -


```
        =E000                  SECT CMDINT, ABS, LOC=0E000H
                               NAME CMDINT
                               SUBTTL  Command interpreter module


             ;*****************************************************************
             ;
             ; Command interpreter module
             ;
             ; Inputs: none.
             ;
             ; Outputs: a Parameter Block (see below).
             ;
             ; Description: this module parses a sprites command string and, for
             ;              each recognized command, produces a parameter block
             ;              describing the command.  This parameter block is passes
             ;              to the interface handler module.  To parse a command
             ;              string first it must be found.  The global sprites
             ;              variable NAME contains the name of the string variable
             ;              containing the command string.  The RCO routine FIND_N
             ;              is used to find the variable in the VARS area of memory.
             ;              Note: symbols in all upper case are external to this
             ;              module and are defined under Imports below.
             ;
             ;              Following is an architecture overview showing the relation
             ;              between the modules of the sprite services component of
             ;              the application development library:
             ;
             ;
             ;                          command
             ;              -----------  string   ---------------
             ;              |  BASIC  |---------->|   Command   |
             ;              | program |<----------| Interpreter |
             ;              -----------  Status   ---------------
             ;                           Code        |         ^
             ;                                       |         |
             ;                     Parameter Block   |         |  Status Code
             ;                                       v         |
             ;                                    ---------------
             ;                                    |     I/P     |
             ;                                    |   Handler   |
             ;                                    ---------------
             ;                                       |         ^
             ;                     Register Values   |         |  Status Code
             ;                                       |         |
             ;                                       v         |
             ;         ------------ Reg values    ---------------
             ;         | assembly |------------->|   Sprite    |
             ;         | program  |<-------------|  Services   |
             ;         ------------   Status      ---------------
             ;                        Code           |         ^
             ;                                       |         |
```

```
                                  ;                        Register Values |        | Status Code
                                  ;                                        |        |
                                  ;                                        v        |
                                  ;                                    ---------------
                                  ;                                    |  Low Level  |
                                  ;                                    | I/O Services|
                                  ;                                    ---------------
                                  ;
                                  ;
                                  ;            Note: the interface between the I/O handler module and the
                                  ;            Sprite Services module is the same as the interface between
                                  ;            an assembly language program and the Sprite Services module.
                                  ;
                                  ;
                                  ;            Following is the layout in memory of the code and data used
                                  ;            by the sprite services component of the application
                                  ;            development library:
                                  ;
                                  ;
                                  ;                  --------------------
                                  ;                  |                  |
                                  ;                  --------------------<-- (UDG)
                                  ;                  |     sprites      |
                                  ;                  |      code        |
                                  ;                  --------------------<-- SPACODE
                                  ;                  |     sprites      |
                                  ;                  |     global       |
                                  ;                  |     variables    |
                                  ;                  --------------------
                                  ;                  |     sprites      |
                                  ;                  |      data        |
                                  ;                  --------------------<-- (SPRITE1)
                                  ;                  |                  |
                                  ;                  |                  |
                                  ;                  |                  |
                                  ;                  --------------------<-- (RAMTOP)
                                  ;                  |                  |
                                  ;                  |                  |
                                  ;                  |                  |
                                  ;                  |------------------|
                                  ;                  | command string   |
                                  ;                  |------------------|
                                  ;                  --------------------<-- (VARS)
                                  ;
                                  ;*****************************************************************


                                  ; Imports

=5C3A                             FLAG1          equ     5C3AH
=5C0B                             CH_ADD         equ     5C0BH

=0008                             ERROR          equ     8
=30F9                             ININT          equ     30F9H
=3160                             FP2BC          equ     3160H
=2C70                             FIND_N         equ     2C70H
=3008                             DIGIT          equ     3009H

                                                 EXTERNAL  I_F_HANDLER


                                  ; Globals

                                                 INCLUDE  spritedef

                                  ; Global constants

=0018                             max_rows       equ     24             ;number of rows on the screen
=0020                             max_cols       equ     32             ;number of columns on the screen
=E000                             sprcode        equ     0E000H         ;location of sprites code
=0010                             globallen      equ     29             ;length of global vars

                                  ; Global variables

=DFF8                             sprites        equ     sprcode-2      ;location of sprites data
=DFFC                             name           equ     sprites-2      ;name of command string var.
=DFFB                             gflags         equ     name-1         ;global flags
=0000                             neg            equ     0              ;  bit 0 - parm < 0
=0001                             autowrap       equ     1              ;  bit 1 - autowrap mode flag
=DFFA                             max_sprites    equ     gflags-1       ;number of sprites allocated
=DFF9                             screen_ht      equ     max_sprites-1  ;number of lines of screen used by
                                                                        ;  sprites
=DFF8                             backgcolor     equ     screen_ht-1    ;permanent background color
=DFF7                             command        equ     backgcolor-1   ;number of command being executed
=DFEB                             parms          equ     command-12     ;parameter block
=DFE3                             scratchpad     equ     parms-8        ;workspace area

                                  ; Data structures

                                  ; parameter block
=DFEB                             svc_code       equ     parms          ;code identifying service
=DFEC                             num_parms      equ     svc_code+1     ;number of parms found
=DFED                             parm1          equ     num_parms+1    ;first parm
=DFEF                             parm2          equ     parm1+2
=DFF1                             parm3          equ     parm2+2
=DFF3                             parm4          equ     parm3+2
=DFF5                             parm5          equ     parm4+2        ;last possible parm

                                  ; sprite data offsets
=0000                             row            equ     0              ;row position
=0001                             col            equ     row+1          ;column position
=0002                             width          equ     col+1          ;sprite width
=0003                             height         equ     width+1        ;sprite height
=0004                             color          equ     height+1       ;sprite color
```

```
=0E03                          sbitmap      equ     color+1        ;definition address
=0007                          sflags       equ     sbitmap+2      ;local sprite flags
=0000                          alloc        equ     0              ;  bit 0 - allocated?
=0001                          overwrite    equ     1              ;  bit 1 - sprite overwrote something
=0002                          lhs          equ     2              ;  bit 2 - sprite in lower half screen



                               ; Locals

                               ; state names for parser
=0000                          s0           equ     0
=0001                          s1           equ     1
=0002                          s2           equ     2
=0003                          s_end        equ     3
=00FF                          s_error      equ     0FFH


                               ;*****************************************************************
                               ;
                               ; CMDINT
                               ;
                               ; Inputs: none.
                               ;
                               ; Outputs: parameter block.
                               ;
                               ; Globals Read: none.
                               ;
                               ; Globals Written: name
                               ;                  command
                               ;                  gflags
                               ;                  parms
                               ;
                               ; Procedures Called: find_string
                               ;                    u_or_l_alpha
                               ;                    ERROR
                               ;                    letter
                               ;                    minus
                               ;                    digit
                               ;                    number
                               ;                    comma
                               ;                    space
                               ;                    I_F_HANDLER
                               ;
                               ; Procedures Called By: none.
                               ;
                               ; Description: this is the main routine of the command interpreter.
                               ;              The command string is located (using FIND_N) and is
                               ;              recognised by the following automaton:
                               ;
                               ;                                        comma
                               ;                 blank          minus---------
                               ;                 -----          -----|        |
                               ;                 |   |          |    |v       |
                               ;                 --> s0 ----------> s1 ----> s2 ------------> s_end
                               ;                     |  and letter  |  uint  |  space or eos
                               ;                     |              |        |
                               ;                     |              v        |
                               ;                     ----------> s_error <----
                               ;
                               ;              where uint represents an unsigned integer and eos
                               ;              represents the end of the string.
                               ;
                               ;*****************************************************************


=E000                                       ORG     opcode


E000    3E 24                  cmdint       ld      a, "d"         ;put a "d" in the 2'nd byte of name
E002    32 DFFD                             ld      (name+1), a
E005    AF                                  xor     a              ;initialise command variable
E006    32 DFF7                             ld      (command), a
E009    21 DFFB                             ld      hl, gflags
E00C    CB 86                               res     neg, (hl)      ;assume positive number until
                                                                   ;  shown otherwise
E00E    CD E092                             call    find_string    ;hl -> command string record
E011    23                                  inc     hl             ;hl -> string length
E012    5E                                  ld      e, (hl)
E013    23                                  inc     hl
E014    56                                  ld      d, (hl)        ;de = string length
E015    23                                  inc     hl
E016    E5                                  push    hl
E017    19                                  add     hl, de
E018    EB                                  ex      de, hl         ;de -> first byte after string
E019    E1                                  pop     hl             ;hl -> first byte of string
E01A    06 00                               ld      b, s0          ;state <- s0
E01C    0E 00                  while0       ld      c, 0           ;params <- 0
E01E    DD 21 DFED                          ld      ix, parms      ;ix -> current parm
E022    3E 00                               ld      a, s0
E024    B8                                  cp      b
E025    20 66                               jr      nz, endw0      ;state <> s0, so exit outer loop
E027    CB 78                  while1       bit     7, b
E029    28 02                               jr      z, ok          ;state <> s_error, so continue
E02B    CF                                  rst     ERROR          ;otherwise produce report "d"
E02C    E9                                  defb    9
E02D    3E 03                  ok           ld      a, s_end
E02F    B8                                  cp      b
E030    28 3E                               jr      z, endw1       ;state = s_end, so exit inner loop
E032    E5                                  push    hl
E033    A7                                  and     a
E034    ED 52                               sbc     hl, de
E036    E1                                  pop     hl
E037    28 1D                               jr      z, do_cmd      ;at end of command string, so execute
```

```
8059   7E                261                     ld      a, (hl)         ;get next char
803A   CC 8059           262                     call    u_or_l_alpha
803D   30 08             263                     jr      nc, not_alpha   ;jump if not upper or lower case letter
803F   C0 80CE           264                     dall    letter          ;otherwise handle letter
                         265                     jr      while1
8044   0E 10             266     not_alphA       ld      c, '-'
8046   2E 05             267                     jr      nz, not_minuA   ;jump if not minus sign
8048   CC 8180           268                     dall    minus           ;otherwise handle minus
804B   18 06             269                     jr      while1
804D   C0 5009           270     not_minus       call    DIGIT
8050   38 E5             271                     jr      c, not_digit    ;jump if not digit
8052   C0 8121           272                     Acall   number          ;otherwise handle number
8055   18 00             273                     jr      while1
8057   FE 2C             274     not_digit       cp      ','
8059   2E 05             275                     jr      nz, not_Aposs   ;jump if not comma
805B   C0 818E           276                     call    aposs           ;otherwise handle comma
805E   18 C7             277                     jr      while1
8060   0E 20             278     not_Aposs       ld      c, ' '
8062   20 05             279                     jr      nz, bad_char    ;not a space either, so must be illegal
8064   CD 8164           280                     call    Aspace          ;handle space
8067   18 EE             281                     jr      while1
8069   06 FF             282     bad_char        ld      b, s_error      ;state <- s_error
806B   18 E6             283                     jr      while1
                         284     endwl           push    hl              ;here when exited inner loop
                         285                     end     e
806F   BC 01             286                     cbd     hl, de
8071   E1                287                     pop     hl
8072   28 02             288                     jr      z, do_eod       ;at end of string, so execute
8074   0A 00             289                     ld      b, s0           ;else more string left
8076   75                290     do_eod          ld      e, A            ;fill in num_terms slot of term block
8077   32 0FEC           291                     ld      (num_termsl, a
807A   C5                292                     push    bc
807B   C5                293                     push    de
807C   E5                294                     push    hl
807D   CD 8001e          295                     call    I_P_HANDLER     ;invoke interface handler module
8080   C1                296                     push    bA              ;save status code in bc'
8081   09                297                     exx
8082   C1                298                     pop     bA
8083   09                299                     exx
8084   21 0FF7           300                     ld      hl, termcntA    ;increment terminal counter
8087   8A                301                     inc     (hl)
8088   E1                302                     cbd     hl
8089   01                303                     pop     de
808A   C1                304                     pop     bc
808B   1E BF             305                     jr      whileB
808D   09                306     endwB           exx                     ;done with command string, so return
808E   C8                307                     push    bA              ;  last status code in bc
808F   09                308                     exx
8090   C1                309                     pop     bc
8091   C9                310                     ret
                         311
                         312     ;***************************************************************************
                         313     ;
                         314     ; FIND_STRING
                         315     ;
                         316     ; Inputs: name (see global variables).
                         317     ;
                         318     ; Outputs: hl -> record for string variable if found
                         319     ;
                         320     ; Globals Used: CM_ADD
                         321     ;                FLAGB
                         322     ;                name
                         323     ;
                         324     ; Globals Written: CM_ADD
                         325     ;                  FLAGS
                         326     ;
                         327     ; Procedures Called: FIND_N
                         328     ;                    ERROA
                         329     ;
                         330     ; Procedures Called By: cmdint
                         331     ;
                         332     ; Description: this routines attempts to find the record for a string
                         333     ;              variable with name equal to the value of NAME.  If found,
                         334     ;              HL is set to point to the length field of the record.  If
                         335     ;              the variable is not found, report "C" is produced.
                         336     ;
                         337     ;***************************************************************************
                         338
8092   ED 48 8C80        339     find_string     ld      bc, (CM_6001    ;save CM_ADD
8096   C5                340                     push    bc
8097   3A 8C3A           341                     ld      a, (FLAGS1      ;save FLAGS
809A   F5                342                     push    af
809B   01 0FFC           343                     ld      bc, name
809E   ED 43 5C50        344                     ld      (CM_ADD1, bc    ;set CM_ADD to point to name
80A2   3E 80             345                     ld      a, 80H
80A4   32 8C3A           346                     ld      (FLAGB), a      ;set runtime flag, reset others
80A7   CD 2C70           347                     call    FIND_N          ;find variable
80AA   30 02             348                     jr      nc, f_ok        ;jump if found variable
80AC   CF                349                     rst     0AADA           ;else produce report "C"
80AD   09                350                     defb    08H
80AE   C1                351     f_ok            exx     bc
80AF   78                352                     ld      a, b
80B0   32 8C3A           353                     ld      (FLAGB), a      ;restore FLAGB
80B3   C1                354                     exx     bc
80B4   ED 43 5C80        355                     ld      (CM_6003), bc   ;restore CM_ADD
80B8   C9                356                     ret
                         357
```

```
                        ;*********************************************************************
                        ;
                        ; U_OR_L_ALPHA
                        ;
                        ; Inputs: current character in A.
                        ;
                        ; Outputs: sets carry flag is current letter is an upper or lower case
                        ;          letter.  Otherwise, resets the carry flag.
                        ;
                        ; Globals Read: none.
                        ;
                        ; Globals Written: none.
                        ;
                        ; Procedures Called: none.
                        ;
                        ; Procedures Called By: cmdint
                        ;
                        ; Description: this routine checks if the current character is an upper
                        ;              or lower case letter.  If so, the carry flag is set,
                        ;              otherwise the carry flag is reset.  If the character is a
                        ;              lower case letter, it is converted to an upper case letter.
                        ;
                        ;*********************************************************************


E0B9    FE 41           u_or_l_alpha    cp      "A"             ;check if upper case letter
E0BB    3F                              ccf
E0BC    D0                              ret     nc              ;less than "A", so return
E0BD    FE 5B                           cp      "Z"+1
E0BF    D8                              ret     c               ;is upper case letter
E0C0    FE 61                           cp      "a"             ;check if lower case letter
E0C2    3F                              ccf
E0C3    D0                              ret     nc              ;less than "a", so return
E0C4    FE 7B                           cp      "z"+1
E0C6    D0                              ret     nc              ;greater than "z", so return
E0C7    D6 20                           sub     "a"-"A"         ;convert to upper case
E0C9    37                              scf                     ;set carry flag
E0CA    C9                              ret


                        ;*********************************************************************
                        ;
                        ; LETTER
                        ;
                        ; Inputs: current character in A.  The character is an upper case letter.
                        ;         hl = current string pointer.
                        ;         b  = parser state
                        ;
                        ; Outputs: svc_code in the parameter block.
                        ;          b = parser state
                        ;
                        ; Globals Read: none.
                        ;
                        ; Globals Written: parse
                        ;
                        ; Procedures Called: none.
                        ;
                        ; Procedures Called By: cmdint
                        ;
                        ; Description: this routine translates the current character, which is
                        ;              known to be a letter, into a code.  If this code = err,
                        ;              then the letter is not a legal command character.  In this
                        ;              case state is set to s_error.  Otherwise, the code is the
                        ;              service code appropriate to the command letter and is put
                        ;              into svc_code in the parameter block.  state is set to s1.
                        ;
                        ;*********************************************************************


        =00FF           err             equ     0FFH


E0CB    F5              letter          push    af
E0CC    3E 00                           ld      b, s0
E0CE    EE                              cp      b
E0CF    28 E6                           jr      z, l_ok0        ;state = s0, so ok
E0D1    F1                              pop     af
E0D2    E6 FF                           ld      b, s_error      ;otherwise, state is illegal
E0D4    C9                              ret
E0D5    F1              l_ok0           pop     af
E0D6    D5                              push    de
E0D7    E5                              push    hl
E0D8    21 E0F3                         ld      hl, l_table     ;hl -> letter state table
E0DB    D6 41                           sub     "A"
E0DD    16 00                           ld      d, 0
E0DF    5F                              ld      e, a
E0E0    19                              add     hl, de          ;hl -> desired table entry
E0E1    7E                              ld      a, (hl)
E0E2    FE FF                           cp      err
E0E4    20 04                           jr      nz, l_ok1       ;code is ok, so jump
E0E6    06 FF                           ld      b, s_error      ;else code is illegal
E0E8    18 05                           jr      l_exit
E0EA    32 D0E3         l_ok1           ld      (svc_code), a   ;otherwise put code in parm block
E0ED    06 01                           ld      b, s1           ;state <- s1
E0EF    E1              l_exit          pop     hl
E0F0    23                              inc     hl              ;advance string pointer
E0F1    D1                              pop     de
E0F2    C9                              ret


E0F3    07              l_table         defb    7               ;Change_Sprite_Attribute
E0F4    FF                              defb    err
E0F5    01                              defb    1               ;Create_Sprite
E0F6    FF                              defb    err
E0F7    04                              defb    4               ;Erase_Sprite
E0F8    FF                              defb    err
E0F9    FF                              defb    err
E0FA    0A                              defb    10              ;Horizontal_Scroll
```

```
8DF5    0D                          defb    0               ;Init_Sprites
8DFC    0F                          defb    err
8DFD    00                          defb    err
8DFE    D6                          defb    6               ;Change_Sprite_Location
8D00    05                          defb    5               ;Read_Sprite
8100    00                          defb    err
8101    05                          defb    0               ;Overlap?
8102    05                          defb    5               ;Put_Sprite
8103    FF                          defb    err
8104    0F                          defb    err
8105    05                          defb    5               ;Init_Screen
8106    0F                          defb    err
8107    FF                          defb    err
8108    05                          defb    11              ;Vertical_Scroll
8109    02                          defb    2               ;Set_Autowrap_Mode
810A    FF                          defb    err
810B    0F                          defb    err
810C    FF                          defb    err


        ;******************************************************************
        ;
        ; MINUS
        ;
        ; Inputs: hl = string pointer.
        ;         b  = parser state.
        ;
        ; Outputs: b = parser state.
        ;
        ; Globals Read: gflags
        ;
        ; Globals written: gflags
        ;
        ; Procedures Called: none.
        ;
        ; Procedures Called By: gdint
        ;
        ; Description: if the state is not s1, sets state to s_error.  Otherwise,
        ;              toggles bit neg of gflags and stays in state s1.
        ;
        ;******************************************************************


810D    23              minus       inc     hl              ;advance string pointer
810E    1E 01                       ld      e, s1
8110    B5                          cp      b
8111    28 03                       jr      z, m_ok         ;state = s1, so ok
8113    05 FF                       ld      b, s_error      ;otherwise state <- s_error
8115    C5                          ret
8116    E5              m_ok        push    hl
8117    21 D0F5                     ld      hl, gflags
811A    7E                          ld      a, (hl)
811B    1F                          rra
811C    3F                          ccf                     ;toggle bit neg
811D    17                          rla
811E    77                          ld      (hl), a
811F    E1                          pop     hl
8120    C5                          ret


        ;******************************************************************
        ;
        ; NUMBER
        ;
        ; Inputs: current string pointer in HL.
        ;         b = parser state.
        ;
        ; Outputs: a parameter in the parameter block.
        ;          b = parser state.
        ;
        ; Globals Read: CH_ADD
        ;               gflags
        ;
        ; Globals Written: CH_ADD
        ;                  gflags
        ;                  parms
        ;
        ; Procedures Called: ININT
        ;                    FP2BC
        ;
        ; Procedures Called By: gdint
        ;
        ; Description: this routine converts the digit string starting at the
        ;              current value of the string pointer and ending with the
        ;              first non-digit character into an unsigned two byte number.
        ;              If the neg flag is set the 2's complement is taken.  The
        ;              number is put into the current parm slot in the parm block.
        ;              The value of #parms is incremented.  State becomes s2.
        ;
        ;******************************************************************


8121    C5              number      push    bc
8122    D5                          push    de
8123    ED 5B 5C5D                  ld      de, (CH_ADD)    ;save CH_ADD
8127    D5                          push    de
8128    22 5C5D                     ld      (CH_ADD), hl    ;set CH_ADD to point to digit string
812B    CD 50F5                     call    INlNT           ;convert string to number. Number
                                                            ;  ends up on calculator stack.
812E    E5                          push    hl
812F    CD 3165                     call    FP2BC           ;put number in bc
8132    21 D0F5                     ld      hl, gflags
8135    C6 46                       bit     neg, (hl)
8137    28 07                       jr      z, pos
```

```
0139   78                         ld     a, b          ;here to convert to 2's complement
013A   2F                         cpl
013B   47                         ld     b, a
013C   79                         ld     a, c
013D   2F                         cpl
013E   4F                         ld     c, a
013F   03                         inc    bc
0140   CB 86          pos         res    neg, (hl)
0142   DD 71 00                   ld     (ix), c       ;put number into current zero slot
0145   DD 23                      inc    ix
0147   DD 70 00                   ld     (ix), b
014A   DD 23                      inc    ix
014C   E1                         pop    hl
014D   D1                         pop    de
014E   ED 53 5C50                 ld     (CH_600), de  ;restore CH_600
0152   D1                         pop    de
0153   C1                         pop    bc
0154   0C                         inc    c             ;ccount <- ccount + 1
0155   06 02                      ld     b, s2         ;state <- s2
0157   C9                         ret


              ;****************************************************************
              ;
              ; COMMA
              ;
              ; Inputs: current string pointer in HL.
              ;         b = parser state.
              ;
              ; Outputs: b = parser state.
              ;
              ; Globals Read: none.
              ;
              ; Globals Written: none.
              ;
              ; Procedures Called: none.
              ;
              ; Procedures Called By: cmdint
              ;
              ; Description: this routine recognizes a comma.  If state is s2, then
              ;              the comma is skipped and state becomes s1.  Otherwise
              ;              state becomes s_error.
              ;
              ;****************************************************************


0158   3E 02          comma       ld     a, s2
015A   B8                         cp     b
015B   28 03                      jr     z, c_ok       ;if state = s2 then ok
015D   06 FF                      ld     b, s_error    ;else state <- s_error
015F   C9                         ret
0160   23             c_ok        inc    hl            ;advance string pointer
0161   06 01                      ld     b, s1         ;state <- s1
0163   C9                         ret


              ;****************************************************************
              ;
              ; SPACE
              ;
              ; Inputs: current string pointer in HL.
              ;         b = parser state.
              ;
              ; Outputs: b = parser state.
              ;
              ; Globals Read: none.
              ;
              ; Globals Written: none.
              ;
              ; Procedures Called: none.
              ;
              ; Procedures Called By: cmdint
              ;
              ; Description: this routine recognizes a space.  If state is s0, then
              ;              the space is skipped and state remains s0.  Otherwise
              ;              state becomes s_end.
              ;
              ;****************************************************************


0164   3E 00          space       ld     a, s0
0166   B8                         cp     b
0167   28 03                      jr     z, s_ok       ;if state = s0 then ok
0169   06 05                      ld     b, s_end      ;else state <- s_end
016B   C9                         ret
016C   23             s_ok        inc    hl            ;advance string pointer
016D   C9                         ret


                                  end
```

```
CMDINT  CR280/11 version 10.36.14        15-Feb-84  1618162
Command Interpreter module   CMDINT.SRC

A         Reserved     ALLOC    0003       AUTOMB   0001        S        Reserved     BACKGC   80FE
BPBSCH    E065         C        Reserved   CHBPDO   1C50        CMDINT   E030         COL      0301
COLOR     0004         COMMP    E159       COMMPN   0#07        CBOK     E160         D        Reserved
DIGIT     5005         DOBCMD   E076       E        Reserved    EMDWO    EG80         EMDW1    E360
EBB       00PP         EXXOR    000E       FINDSN   0070        FINDBS   E092         FLPGS    5C5P
PP2EC     5160         FBOX     E04E       GPLAGS   0PPE        GLOBAL   8010         H        Reserved
HEIGHT    0003         ININT    50P9       ISPSM6   0001 Ex     L        Reserved     LETTEB   E0C9
LMS       0802         LBEXIT   E0EP       LBOKO    E005        LBOX1    ECE6         LETAEL   E0PS
N         Reserved     MAXBCO   0020       MAXBBO   301E        MAXBSP   G0P6         MINUS    E100
MBOX      E116         NAME     0PPC       NEG      C000        NOTBAL   E044         NOTPCO   E060
NOTBOI    E857         NOTBMI   E040       NUMBEB   F121        NUMBP6   DPEC         OK       E02C
OVERMB    0001         PARMS    0#E8       PAPMI    0#ED        PPRM2    0#EP         PARM3    0PP1
PARM4     0PP3         PARMS    0#P5       PEITMA   C305        POS      E140         PSM      Reserved
BOM       0000         SCPP7C   0PE3       SCBEEN   DPP9        SPLAGS   0007         SP       Reserved
SPACE     E164         SPBCDO   E000       SPBITE   CPFE        SVCBCO   0PE8         SBEND    8005
SEERRO    00PP         SBOK     E16C       S0       0000        S1       8001         S2       8002
UEORBL    E085         WHILE0   E01C       WHILE1   E027        WIDTH    8E02

     No errors detected
```

```
Cross reference listing (MPSP version 4.7)

Symbol     Refs (# = definition  $ = write  <blank> = read)

ALLOC      159P
AUTOMB     131#
BACKGC     135#    156
B40_CH     279     282#
CH_400     103#    341      346$     357$     563      565$     587$
CMDINT     226#
CCL        153#    154
COLOR      156#    157
COMMP      276     619#
COMMPN     136#    137      229$     300
C_OK       621     624#
DIGIT      109#    270
DO_CMD     240     288      290P
EMDWO      248     306#
EMDW1      255     284#
EBB        430R    449      462      466      466      467      470
                   471      474      477      478      480      481
                   484      485      486
ERROR      105R    251      551
FIND_M     108#    549
FIND_S     233     341P
FLAGS      102#    343      348X     555$
FP2BC      107#    569
F_OK       350     353P
GPLAGS     129#    132      250      519      570
GLOBAL     125#
HEIGHT     155#    156
ININT      106#    566
I_#_M6     111#    295
LETTEB     264     433#
LMS        161P
L_EXIT     452     455#
L_OKO      436     440#
L_OK1      450     455#
L_TPEL     443     461P
MAX_CO     121#
MAX_RO     120#
MAX_SP     132#    135
MINUS      268     512P
M_OK       515     518#

NAME       128#    123      227$     345
NEG        130#    251      571      580
NOT_FL     263     266#
NOT_CO     275     278#
NOT_DI     271     274#
NOT_#I     267     270#
NUMBER     272     561#
NUM_P6     144#    145      291$
OK         250     253#
OVERMB     160#
PARM1      145#    146      245
PARM2      146P    147
PARMS      147#    148
PARM6      148#    149
PARMS      149#
PARMS      137#    138      145
PEITM6     157#    158
POS        572 -   580P
BOM        152#    153
S0         168P    243      246      289      454      655
S1         167#    454      515      625
S2         170#    591      615
SCRATC     138#
SCBEEN     133P    155
SFLAGS     158#
SPACE      280     655#
SPRCDO     122#    127      223
SPBITE     127P    128
SVC_CO     143#    144      453$
S_EMD      171#    255      656
S_ERBO     172#    282      438      451      516      622
S_OK       655     658#
U_DB_L     262     386#
WHILE0     244#    305
WHILE1     249P    265      265      273      277      281      285
WIDTH      154#    155
```

```
                              name  IFHANDLER
                              SUS77L interface handler module

            !***********************************************************************
            !
            ! Interface handler module
            !
            ! Inputs: a parameter block.
            !
            ! Outputs: a status code in RC.
            !
            ! Description: this module translates a parameter block into a call to
            !              the appropriate sprite service.  The number of parameters
            !              specified in the parm block is checked against a table
            !              entry for the service containing the correct number of
            !              parameters.  If these values do not match, the report
            !              "C: Parameter error" is produced.  Otherwise the interface
            !              routine appropriate to the service is invoked.  This
            !              interface routine puts the values of the parms in the
            !              parm block into the appropriate registers.  The sprite
            !              service is then invoked.   When the service returns, the
            !              status code is checked and, if an error occurred, an
            !              appropriate report is produced.
            !
            !***********************************************************************


            ! Imports

=0000       ERROR             equ     0
=E219       INIT_SPRITES      equ     0E219H
=E419       INITSCREEN        equ     0E419H
=E278       CREATE_SPRITE     equ     0E278H
=E2C9       SET_AUTOMODE      equ     0E2C9H
=E307       PUT_SPRITE        equ     0E307H
=E381       ERASE_SPRITE      equ     0E381H
=E420       MOVE_SPRITE       equ     0E420H
=E450       CH_LOC            equ     0E450H
=E460       CH_ATTR           equ     0E460H
=E480       OVERLAP           equ     0E480H
=E500       H_SCROLL          equ     0E500H
=E771       V_SCROLL          equ     0E771H



            ! Globals

                              INCLUDE spritedef

            ! Global constants

=0018       max_rows          equ     24              !number of rows on the screen
=0020       max_cols          equ     32              !number of columns on the screen
=E000       sprcode           equ     0E000H          !location of sprites code
=0015       globalslen        equ     29              !length of global vars

            ! Global variables

=DFFE       sprites           equ     sprcode-2       !location of sprites data
=DFFC       name              equ     sprites-2       !name of command string var.
=DFFB       gflags            equ     name-1          !global flags
=0000       neg               equ     0               !  bit 0 - parm < 0
=0001       automode          equ     1               !  bit 1 - automode mode flag
=DFFA       max_sprites       equ     gflags-1        !number of sprites allocated
=DFF9       screen_ht         equ     max_sprites-1   !number of lines of screen used by
                                                      !  sprites
=DFF8       backgcolor        equ     screen_ht-1     !permanent background color
=DFF7       command           equ     backgcolor-1    !number of command being executed
=DFEB       parms             equ     command-12      !parameter block
=DFE3       scratchpad        equ     parms-8         !workspace area

            ! Data structures

            ! parameter block
=DFEB       svc_code          equ     parms           !code identifying service
=DFEC       num_parms         equ     svc_code+1      !number of parms found
=DFED       parm1             equ     num_parms+1     !first parm
=DFEF       parm2             equ     parm1+2
=DFF1       parm3             equ     parm2+2
=DFF3       parm4             equ     parm3+2
=DFF5       parm5             equ     parm4+2         !last possible parm

            ! sprite data offsets
=0000       row               equ     0               !row position
=0001       col               equ     row+1           !column position
=0002       width             equ     col+1           !sprite width
=0003       height            equ     width+1         !sprite height
=0004       color             equ     height+1        !sprite color
=0005       defaddr           equ     color+1         !definition address
=0007       sflags            equ     defaddr+2       !local sprite flags
=0000       alloc             equ     0               !  bit 0 - allocated?
=0001       overwrite         equ     1               !  bit 1 - sprite overwrote something
=0002       lhs               equ     2               !  bit 2 - sprite in lower half screen
```

```
                                ;*****************************************************************
                                ;
                                ; I_F_HANDLER
                                ;
                                ; Inputs: a parameter block.
                                ;
                                ; Outputs: a status code in BC.
                                ;
                                ; Globals Read: parms
                                ;
                                ; Globals Written: none.
                                ;
                                ; Procedures Called: ERROR
                                ;                    jmp_hl
                                ;                    jmp_ix
                                ;
                                ; Procedures Called By: CMDINT
                                ;
                                ; Description: this is the main routine of this module.  It translates
                                ;              the values in the parm block into a call to a sprite
                                ;              service.  The interfaces to the sprite services are
                                ;              described in a table.  For each service the is an entry
                                ;              containing the correct number of parameters, the address
                                ;              of the interface routine appropriate to that number of
                                ;              parameters, and the address of the service.  The number
                                ;              of parms in the table entry is check against the value in
                                ;              the parm block.  If they do not agree, report "Q" is produced.
                                ;              If they do agree, the interface routine indicated in the
                                ;              table entry is called.  The service indicated in the table
                                ;              entry is then called.   If the part of the status code in C
                                ;              is non-zero, then an error occurred in executing the sprite
                                ;              service, and an appropriate report is produced.
                                ;
                                ;*****************************************************************
        =E16E                                   ORG     0E16EH

E16E'   3A 0FE5                 i_f_handler     ld      a, (svc_code)   ;check num_parms against the
                                                                        ;  value in i_f_table
E171'   47                                      ld      b, a
E172'   CB 27                                   sla     a
E174'   CB 27                                   sla     a
E176'   80                                      add     a, b            ;a = 5*svc_code
E177'   21 E1B5'                                ld      hl, i_f_table
E17A'   5F                                      ld      e, a
E17B'   16 00                                   ld      d, 0
E17D'   19                                      add     hl, de          ;hl -> table entry
E17E'   3A 0FEC                                 ld      a, (num_parms)
E181'   BE                                      cp      (hl)
E182'   28 02                                   jr      z, ok           ;if num_parms correct, then jump
E184'   CF                                      rst     ERROR           ;else produce report "Q"
E185'   19                                      defb    25
E186'   23                      ok              inc     hl              ;hl -> i/f routine addr
E187'   5E                                      ld      e, (hl)
E188'   23                                      inc     hl
E189'   56                                      ld      d, (hl)
E18A'   23                                      inc     hl
E18B'   EB                                      ex      de, hl          ;hl = i/f routine addr
E18C'   1A                                      ld      a, (de)         ;de -> service addr
E18D'   4F                                      ld      c, a
E18E'   13                                      inc     de
E18F'   1A                                      ld      a, (de)
E190'   47                                      ld      b, a
E191'   DD 21 0000                              ld      ix, 0
E195'   DD 09                                   add     ix, bc          ;ix = service addr
E197'   DD E5                                   push    ix
E199'   DD 21 0FED                              ld      ix, parms       ;ix -> parameters
E19D'   CD E1B2'                                call    jmp_hl          ;invoke i/f routine
E1A0'   DD E1                                   pop     ix
E1A2'   CD E1B3'                                call    jmp_ix          ;invoke sprite service
E1A5'   CB 41                                   bit     0, c            ;check status code
E1A7'   C8                                      ret     z               ;if c0 = 0, then no error
E1A8'   79                                      ld      a, c
E1A9'   FE 01                                   cp      1
E1AB'   20 02                                   jr      nz, enough_room ;if c <> 1, then not out of room
E1AD'   CF                                      rst     ERROR           ;else produce report "4"
E1AE'   03                                      defb    3
E1AF'   CF                      enough_room     rst     ERROR           ;produce report "6"
E1B0'   09                                      defb    9
E1B1'   C9                                      ret


                                ;*****************************************************************
                                ;
                                ; JMP_HL
                                ;
                                ; Inputs: HL = address to be called.
                                ;
                                ; Outputs: none.
                                ;
                                ; Globals Read: none.
                                ;
                                ; Globals Written: none.
                                ;
                                ; Procedures Called: none.
                                ;
                                ; Procedures Called By: i_f_handler
                                ;
                                ; Description: this routine is called to call the routine whose entry
                                ;              point is in HL.
                                ;
                                ;*****************************************************************


                                jmp_hl          jp      (hl)
```

```
                    ;**********************************************************************
                    ;
                    ; JMP_IX
                    ;
                    ; Inputs: IX = address to be called.
                    ;
                    ; Outputs: none.
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: i_f_handler
                    ;
                    ; Description: this routine is called to call the routine whose entry
                    ;              point is in IX.
                    ;
                    ;**********************************************************************


E5E2'  DD E9        jmp_ix          jp      (ix)


                    ;This table contains the interface definitions for the sprite services.  The
                    ;  first item is the number of parameters to be passed to the service.  The
                    ;  second item is the interface routine for doing this.  The third item is the
                    ;  address of the service.

E2E9'  01           i_f_table       defb    2
01E6'  E1F1'                        defw    i_f1
E1EE'  E21E                         defb    INIT_SPRITES

E1EA'  E5                           defb    9
E1B6'  5206'                        defb    i_f5
E1ED'  E275                         defw    CREATE_SPRITE

E1E0'  01                           defb    1
E1C0'  E1F1'                        defw    i_f1
E1C2'  E2C5                         defw    SET_AUTOMAP

E1C4'  02                           defb    2
E1C5'  E1FC'                        defb    i_f3
E1C7'  E307                         defb    PUT_SPRITE

E1C9'  01                           defb    1
E5C6'  E1F1'                        defb    i_f1
05CC'  E261                         defb    ERASE_SPRITE

E5CE'  02                           defb    3
E1C0'  E1FC'                        defb    i_f3
01D1'  E426                         defb    MOVE_SPRITE

E1D3'  D3                           defb    2
E1D4'  E1FC'                        defb    i_f3
E1D6'  5450                         defb    CH_LOC

E1D8'  02                           defb    2
E1D9'  E1F5'                        defb    i_f2
E1DE'  E466                         defb    CH_ATTR

E1DD'  E2                           defb    2
E1DE'  E1F5'                        defb    i_f2
E1E0'  E4E4                         defb    OVERLAP

E1E2'  02                           defb    3
E1E2'  E1FC'                        defb    i_f2
E1E5'  E4E9                         defb    INITSCREEN

E1E7'  E1                           defb    1
E1EE'  E1F1'                        defb    i_f1
E1EA'  E206                         defb    H_SCROLL

E1EC'  E1                           defb    1
E1ED'  E1F1'                        defb    i_f1
E1E0'  E771                         defb    V_SCROLL


                    ;**********************************************************************
                    ;
                    ; i_f1
                    ;
                    ; Inputs: ix -> first parameter
                    ;
                    ; Outputs: a <- value of first parameter
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: i_f_handler
                    ;
                    ; Description: this is the interface routine for sprite services with
                    ;              one parameter.
                    ;
                    ;**********************************************************************


E1F1'  DD 7E 00     i_f1            ld      a, (ix)
E1F4'  C9                           ret
```

```
                    ;************************************************************************
                    ;
                    ; I_F2
                    ;
                    ; Inputs: ix -> first parameter
                    ;
                    ; Outputs: a <- value of first parameter
                    ;          d <- value of second parameter
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: i_f_handler
                    ;
                    ; Description: this is the interface routine for sprite services with
                    ;              two parameters.
                    ;
                    ;************************************************************************


E1F5'  DD 7E 00     i_f2          ld      a, (ix)
E1F8'  DD 56 02                   ld      d, (ix+2)
E1FB'  C9                         ret


                    ;************************************************************************
                    ;
                    ; I_F3
                    ;
                    ; Inputs: ix -> first parameter
                    ;
                    ; Outputs: a <- value of first parameter
                    ;          d <- value of second parameter
                    ;          e <- value of third parameter
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: i_f_handler
                    ;
                    ; Description: this is the interface routine for sprite services with
                    ;              three parameters.
                    ;
                    ;************************************************************************


E1FC'  DD 7E 00     i_f3          ld      a, (ix)
E1FF'  DD 56 02                   ld      d, (ix+2)
E202'  DD 5E 04                   ld      e, (ix+4)
E205'  C9                         ret


                    ;************************************************************************
                    ;
                    ; I_F5
                    ;
                    ; Inputs: ix -> first parameter
                    ;
                    ; Outputs: a <- value of first parameter
                    ;          b <- value of second parameter
                    ;          c <- value of third parameter
                    ;          d <- value of fourth parameter
                    ;         hl <- value of fifth parameter
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: i_f_handler
                    ;
                    ; Description: this is the interface routine for sprite services with
                    ;              five parameters.
                    ;
                    ;************************************************************************


E206'  DD 7E 00     i_f5          ld      a, (ix)
E209'  DD 46 02                   ld      b, (ix+2)
E20C'  DD 4E 04                   ld      c, (ix+4)
E20F'  DD 56 06                   ld      d, (ix+6)
E212'  DD 6E 08                   ld      l, (ix+8)
E215'  DD 66 09                   ld      h, (ix+9)
E218'  C9                         ret


                                  end
```

```
IPHANDLER       CSI80/11 version 10.38.14        15-May-84  9:52:10
Interface handler module    IPHANDLER.SRC

9        Reserved    ALLOC    0030        AUTOWR   0001        8        Reserved      BACKGC   GPF8
C        Reserved    CHSETT   E448        CHALOC   E450        CCL      0001          COLOR    0004
COMMAN   0CP7        CREATE   E27E        0        Reserved    E        Reserved      ENOUGH   E1AC
ERRSE9   E381        ERRC8    000R        GFLAGS   0FPE        GLOBAL   0010          H        Reserved
HEIGHT   0003        HSSCEG   E509        INITSC   E4E9        INITSS   E219          ISCSHA   E18E
ISCSTR   E185        ISP1     E1P1        ISPI     E1C5        ISC3     E1CC          SSPS     E20E
JOCSHL   E182        JOCSIX   E1E3        L        Reserved    LHS      0002          0        Reserved
MAXSCO   0010        MOXSOC   0018        MAXSSP   0FCA        MCVESS   E42R          NAME     0CPC
NEG      0000        NUMSCO   C0EC        OK       E189        OVERLA   E4E4          OVESHB   G001
CACHS    0PE8        CARNI    0CEC        CARNI    DPE0        PACH3    DPE1          CARN6    0PC5
PRROS    0CP5        PEITHA   0005        CSW      Reserved    PUTSS0   E30T          ROW      E000
SCBATC   0FE3        SCREEN   0PF5        SETSRU   EIC5        SFLAGS   0007          SP       Reserved
SPRCOD   E000        SCRITE   0PCE        SVCSCD   DPE0        VSSCRO   E771          WIDTH    0002

      No errors detected
```

```
SPRSVC  CSI80/11 version 10.54.14        15-May-84  9:51:81
Sprite services module    SPRSVC.SRC

                                         NAME: SPRSVC
                                         SUBTTL  Sprite services module


                  ;*****************************************************************
                  ;
                  ; Sprite services module
                  ;
                  ; Inputs: registers set to parameter values for called service.
                  ;
                  ; Outputs: status code in BC: 0000 - OK
                  ;                             0001 - not enough memory
                  ;                             EE05 - illegal sprite id
                  ;                             0005 - illegal color
                  ;                             0007 - illegal number-lines
                  ;                             0100 - OK, something overwritten
                  ;                             0200 - OK, overlap
                  ;
                  ; Description: This module implements the various sprite graphics services.
                  ;              Upon entry to this module, the parameters to the desired
                  ;              service are put into the appropriate registers.  A status
                  ;              code is returned in BC (see above).
                  ;
                  ;*****************************************************************


                  ; Imports (Note:  all symbols in upper case are imported)

                  ;From The Host

                  ;Variables

=9C5R             V                  equ     9C5RH
=9C82             ROWTOP             equ     9C82H
=9C3E             CHARS              equ     9C3EH
=9C00             ATTR_B             equ     9C80H
=9CEP             ATTR_T             equ     9C8PH
=1C5C             TVFLAG             equ     5C5CH
=5C4E             BORDCR             equ     5C48H
=0EPE             BDBOFT             equ     0PEH

                  ;Routines

=05E2             SET_AT             equ     0582H
=0500             SENDTV             equ     0500H
=5110             SELECT             equ     1150H
=E97P             CLS_0              equ     E57PH


                  ;From module IOL

                  ;Variables

=EE90             CHTBL     equ      CEE90H                      ;  Address of ROM Char.Table

                  ;Services

=EE9D             WRITE_CHRS         equ      0EE90H
=E9DE             SET_POINT_CDS      equ      0E9DEH
=E95F             GET_CHAR           equ      0E51PH


                  ; Globals

                                     INCLUDE spriteref

                  ; Global constants

=0E5E             max_rows           equ      24          ;number of rows on the screen
=9E20             max_cols           equ      32          ;number of columns on the screen
=E000             sprcode            equ      0E000H      ;location of sprites code
=0050             globallen          equ      25          ;length of global vars
```

```
                ; Global variables

=DFF8           sprites         equ     parcode-2       ;location of sprites data
=DFFC           name            equ     sprites-2       ;name of command string var.
=DFF8           gflags          equ     name-1          ;global flags
=0000           neg             equ     0               ;  bit 0 - parm < 0
=0001           autowrap        equ     1               ;  bit 1 - autowrap mode flag
=DFF9           max_sprites     equ     gflags-1        ;number of sprites allocated
=DFF9           screen_ht       equ     max_sprites-1   ;number of lines of screen used by
                                                        ;  sprites
=DFF8           backcolor       equ     screen_ht-1     ;permanent background color
=DFF7           command         equ     backcolor-1     ;number of command being executed
=DFE8           parms           equ     command-12      ;parameter block
=DFE3           scratchpad      equ     parms-5         ;workspace area

                ; Data structures

                ; parameter block
=DFE8           svc_code        equ     parms           ;code identifying service
=DFEC           num_parms       equ     svc_code+1      ;number of parms found
=DFED           parm1           equ     num_parms+1     ;first parm
=DFEF           parm2           equ     parm1+2
=DFF1           parm3           equ     parm2+2
=DFF3           parm4           equ     parm3+2
=DFF5           parm5           equ     parm4+2         ;last possible parm

                ; sprite data offsets
=0000           row             equ     0               ;row position
=0001           col             equ     row+1           ;column position
=0002           width           equ     col+1           ;sprite width
=0003           height          equ     width+1         ;sprite height
=0004           color           equ     height+1        ;sprite color
=0005           pbitmap         equ     color+1         ;definition address
=0007           sflags          equ     pbitmap+2       ;local sprite flags
=0000           alloc           equ     0               ;  bit 0 - allocated?
=0001           overwrite       equ     1               ;  bit 1 - sprite overwrote something
=0002           lhs             equ     2               ;  bit 2 - sprite in lower half screen



                ;****************************************************************
                ;
                ; INIT_SPRITES
                ;
                ; Inputs: A = number-sprites, the number of sprites for which memory is
                ;              to be allocated.
                ;
                ; Outputs: BC = 0000 - OK
                ;               0001 - not enough memory
                ;
                ; Globals Read: RAMTOP
                ;
                ; Globals Written: screen_ht
                ;                  gflags
                ;                  max_sprites
                ;                  sprites
                ;                  sprites data structure
                ;
                ; Procedures Called: none.
                ;
                ; Procedures Called By: I_F_HANDLER
                ;                       user program
                ;
                ; Description: this routines allocates memory for sprite data. (max_sprites)
                ;              is set to number-sprites. One sprite requires 8 bytes of
                ;              data. If there is not enough free memory between (RAMTOP)
                ;              and the sprites global variables, then 01 is returned.
                ;              Otherwise, (sprites) is set to point to the start of the
                ;              sprite data area and the sprite data is cleared.
                ;
                ;****************************************************************

=E219                   ORG     0E219H


E219'  21 DFF9  init_sprites    ld      hl, screen_ht   ;initialize screen_ht to 24
E21C'  36 18                    ld      (hl), max_rows
E21E'  21 DFF6                  ld      hl, gflags
E221'  36 00                    ld      (hl), 0         ;clear global flags
E223'  32 DFF4                  ld      (max_sprites), a
E226'  4F                       ld      c, a            ;calc. where the sprite data
                                                        ;  should go
E227'  06 00                    ld      b, 0
E229'  A7                       and     a
E22A'  CB 11                    rl      c
E22C'  CB 10                    rl      b
E22E'  CB 11                    rl      c
E230'  CB 10                    rl      b
E232'  CB 11                    rl      c
E234'  CB 10                    rl      b
E236'  C5                       push    bc              ;bc = 8*number-sprites
E237'  21 E000                  ld      hl, spbase
E23A'  A7                       and     a
E23B'  ED 42                    sbc     hl, bc
E23D'  01 0010                  ld      bc, globallen
E240'  A7                       and     a
E241'  ED 42                    sbc     hl, bc          ;hl -> start of sprite data
E243'  C1                       pop     bc
E244'  ED 5B 5CB2               ld      de, (RAMTOP)    ;see if there is enough free memory
E248'  E5                       push    hl
E249'  A7                       and     a
E24A'  ED 52                    sbc     hl, de
E24C'  E1                       pop     hl
E24D'  30 04                    jr      nc, enough_room ;if enough room, then jump
E24F'  01 0001                  ld      bc, 0001        ;else return 01
E252'  C9                       ret
E253'  22 DFF8  enough_room     ld      (sprites), hl
E256'  36 00    clear_loop      ld      (hl), 0         ;clear sprite data area
E258'  0B                       dec     bc
E259'  23                       inc     hl
E25A'  78                       ld      a, b
E25B'  B1                       or      c
E25C'  20 F8                    jr      nz, clear_loop
E25E'  C9                       ret                     ;return 00
```

```
                     ;************************************************************
                     ;
                     ; SPR_ADDR
                     ;
                     ; Inputs: A = sprite-id
                     ;
                     ; Outputs: IX <- address of sprite record for sprite sprite-id
                     ;
                     ; Globals Read: sprites
                     ;
                     ; Globals Written: none.
                     ;
                     ; Procedures Called: none.
                     ;
                     ; Procedures Called By: create_sprite
                     ;                       check_id
                     ;                       out_sprite
                     ;                       erase_sprite
                     ;                       ch_loc
                     ;                       ch_attr
                     ;                       overlap
                     ;                       upd_cols
                     ;                       upd_rows
                     ;
                     ; Description: this routine calculates the address of the start of the
                     ;              record for the sprite with id = sprite-id.  This address
                     ;              is returned in ix.  Note that each record is 8 bytes long.
                     ;
                     ;************************************************************


025F'  C5            spr_addr       push    bc
0260'  06 00                        ld      b, 0
0262'  4F                           ld      c, a
0263'  A7                           and     a
0264'  CB 11                        rl      c
0266'  CB 10                        rl      b
0268'  CB 11                        rl      c
026A'  CB 10                        rl      b
026C'  CB 11                        rl      c
026E'  CB 10                        rl      b           ;bc = 8*sprite-id
0270'  DD 2A 0FF8                   ld      ix, (sprites)  ;ix -> start of sprites data
0274'  DD 09                        add     ix, bc         ;ix -> start of record sprite-id
0276'  C1                           pop     bc
0277'  C9                           ret


                     ;************************************************************
                     ;
                     ; CREATE_SPRITE
                     ;
                     ; Inputs: A = sprite-id
                     ;         B = width
                     ;         C = height
                     ;         D = color
                     ;        HL = address of bitmap
                     ;
                     ; Outputs: BC = 0000 - OK
                     ;             = 0003 - illegal sprite-id
                     ;             = 0005 - illegal color
                     ;
                     ; Globals Read: sprites
                     ;
                     ; Globals Written: sprite data structure
                     ;
                     ; Procedures Called: spr_addr
                     ;
                     ; Procedures Called By: I_F_HANDLER
                     ;                       user program
                     ;
                     ; Description: this routine initializes a record for the sprite identified
                     ;              by sprite-id.  If the sprite-id parameter is not less than
                     ;              (max_sprites), then it is illegal and 03 is returned in BC.
                     ;              If the color parameter is greater than 7, then it is
                     ;              illegal and 09 is returned in BC.  If the parameters are
                     ;              legal, then the slots of the record are filled in and the
                     ;              alloc flag of aflags is set to indicate that the record has
                     ;              been filled in.
                     ;
                     ;************************************************************


0278'  E5            create_sprite  push    hl             ;validate sprite-id < (max_sprites)
0279'  21 0FFA                      ld      hl, max_sprites
027C'  BE                           cp      (hl)
027D'  E1                           pop     hl
027E'  38 04                        jr      c, id_ok       ;if sprite-id ok, then jump
0280'  01 0003                      ld      bc, 0003       ;else return 03 in BC
0283'  C9                           ret
0284'  F5            id_ok          push    af             ;validate color < 8
0285'  3E 07                        ld      a, 7
0287'  BA                           cp      d
0288'  30 05                        jr      nc, color_ok   ;if color ok, then jump
028A'  F1                           pop     af
028B'  01 0009                      ld      bc, 0005       ;else return 05 in BC
028E'  C9                           ret
028F'  F1            color_ok       pop     af
0290'  CD 025F'                     call    spr_addr       ;ix -> start of sprite record
0293'  DD 36 00 00                  ld      (ix+row), 0    ;row <- 0
0297'  DD 36 01 00                  ld      (ix+col), 0    ;col <- 0
029B'  DD 70 02                     ld      (ix+width), b  ;width <- b
029E'  DD 71 03                     ld      (ix+height), c ;height <- c
02A1'  DD 72 04                     ld      (ix+color), d  ;color <- d
02A4'  DD 75 05                     ld      (ix+bitmap), l ;bitmap <- hl
02A7'  DD 74 06                     ld      (ix+bitmap+1), h
02AA'  DD CB 07 C6                  set     alloc, (ix+aflags)     ;mark record as initialized
02AE'  01 0000                      ld      bc, 0          ;return 00 in BC
02B1'  C9                           ret
```

```
                    ;*****************************************************************
                    ;
                    ; CHECK_ID
                    ;
                    ; Inputs: A = sprite-id
                    ;
                    ; Outputs: CF is set if sprite-id is legal, reset if it is illegal
                    ;
                    ; Globals Read: sprites
                    ;               sprite data structure
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: spr_addr
                    ;
                    ; Procedures Called By: put_sprite
                    ;                       erase_sprite
                    ;                       ch_loc
                    ;                       ch_attr
                    ;                       overlap
                    ;
                    ; Description: this routine checks (1) that sprite-id is in the range
                    ;              [0..max_sprites-1] and (2) that the record for that
                    ;              sprite-ID has been initialized.
                    ;
                    ;*****************************************************************


03B2'  DD E5        check_id      push   ix
03B4'  E5                         push   hl             ;validate sprite-id < (max_sprites)
03B5'  21 0FFA                    ld     hl, max_sprites
03B8'  BE                         cp     (hl)
03B9'  E1                         pop    hl
03BA'  D0                         ret    nc             ;sprite-ID out of range
03BB'  CD 025F'                   call   spr_addr
03BE'  DD CB 07 0E                rrc    (ix+sflags)
03C2'  DD CB 07 06                rlc    (ix+sflags)    ;alloc flag is now in carry flag
03C6'  DD E1                      pop    ix
03C8'  C9                         ret


                    ;*****************************************************************
                    ;
                    ; SET_AUTOWRAP
                    ;
                    ; Inputs: A = 0 - off
                    ;          <> 0 - on
                    ;
                    ; Outputs: BC = 0000 - OK
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: gflags
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: I_F_HANDLER
                    ;                       user program
                    ;
                    ; Description: this routines sets the autowrap flag according to the
                    ;              value of A. If A = 0, the flag is reset, otherwise
                    ;              the flag is set.
                    ;
                    ;*****************************************************************


03C9'  E5           set_autowrap  push   hl
03CA'  21 0FFB                    ld     hl, gflags
03CD'  A7                         and    a
03CE'  28 04                      jr     z, off         ;if A = 0, then jump
03D0'  CB C6                      set    autowrap, (hl) ;else turn flag on
03D2'  18 03                      jr     sa_exit
03D4'  CB 86        off           res    autowrap, (hl)
03D6'  E1           sa_exit       pop    hl
03D7'  01 0000                    ld     bc, 0          ;return 00 in BC
03DA'  C9                         ret


                    ;*****************************************************************
                    ;
                    ; DISPLAY_CHAR
                    ;
                    ; Inputs: B = row
                    ;         C = column
                    ;         DE = location of dot pattern for char to be written
                    ;
                    ; Outputs: DE = location of next char
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: CHTBL
                    ;
                    ; Procedures Called: SET_PRINT_POS
                    ;                    WRITE_CHAR
                    ;
                    ; Procedures Called By: put_sprite
                    ;                       erase_sprite
                    ;
                    ; Description: this routine displays the specified character at the
                    ;              specified screen location.
                    ;
                    ;*****************************************************************


03DB'  E5           display_char  push   hl
03DC'  C5                         push   bc
03DD'  D5                         push   de
03DE'  CD 030E                    call   SET_PRINT_POS  ;set the print position to BC
03E1'  D1                         pop    de
03E2'  3E 20                      ld     a, ' '
03E4'  ED 53 0090                 ld     (CHTBL), de    ;replace (CHTBL) with loc of char
03E8'  D5                         push   de
03E9'  CD 0000                    call   WRITE_CHAR     ;write the character
03EC'  D1                         pop    de
03ED'  21 0008                    ld     hl, 8          ;advance the bitmap pointer
03F0'  19                         add    hl, de
03F1'  EB                         ex     de, hl
```

```
02F2'  21 5C00          417                     ld      hl, 5C00h
02F5'  22 0870          418                     ld      (CH78L), hl     ;restore (CH78L) to original value
02F8'  C1               419     do_exit         pop     bc
02F9'  E1               420                     pop     hl
02FA'  C9               421                     ret
                        422
                        423
                        424     ;***************************************************************************
                        425     ;
                        426     ; V1918L0
                        427     ;
                        428     ; Inputs: B = row
                        429     ;         C = column
                        430     ;
                        431     ; Outputs: CF set if BC is a legal screen location, otherwise CF is reset
                        432     ;
                        433     ; Globals Used: screen_ht
                        434     ;
                        435     ; Globals Written: none.
                        436     ;
                        437     ; Procedures Called: none.
                        438     ;
                        439     ; Procedures Called By: put_sprite
                        440     ;                       erase_sprite
                        441     ;
                        442     ; Description: this routine checks that the specified row and column
                        443     ;              lie within the legal ranges for the screen width and
                        444     ;              height.  If so, the carry flag is set, otherwise it is
                        445     ;              reset.
                        446     ;***************************************************************************
                        447
                        448
02F9'  3A 0FF8          449     visible         ld      a, (screen_ht)  ;0 <= row <= (screen_ht)-1
02FC'  3D               450                     dec     a
02FD'  B8               451                     cp      b
02FE'  3F               452                     ccf
02FF'  D0               453                     ret     nc              ;no, so return
0300'  3E 1F            454                     ld      a, scr_cols-1   ;0 <= col <= max_cols-1
0302'  B8               455                     cp      b
0303'  3F               456                     ccf
0304'  C9               457                     ret                     ;return result of second test
                        458
                        459
                        460     ;***************************************************************************
                        461     ;
                        462     ; PUT_SPRITE
                        463     ;
                        464     ; Inputs: A = sprite-id
                        465     ;         B = row
                        466     ;         C = column
                        467     ;
                        468     ; Outputs: BC = 0000 - OK, nothing overwritten
                        469     ;              = 0100 - OK, something overwritten
                        470     ;              = 0005 - illegal sprite-id
                        471     ;
                        472     ; Globals Used: sprite data structure
                        473     ;               backgcolor
                        474     ;               gflags
                        475     ;               screen_ht
                        476     ;
                        477     ; Globals Written: sprite data structure
                        478     ;                  &778_0
                        479     ;
                        480     ; Procedures Called: check_id
                        481     ;                    spr_addr
                        482     ;                    SBLOC7
                        483     ;                    visible
                        484     ;                    G87_CH48
                        485     ;                    display_char
                        486     ;
                        487     ; Procedures Called By: move_sprite
                        488     ;                       upd_cols
                        489     ;                       upd_rows
                        490     ;                       I_P_HANDL89
                        491     ;                       user program
                        492     ;
                        493     ; Description: this routine writes the specified sprite to the screen.
                        494     ;              if outsprog mode is set, then the specified row and column
                        495     ;              are changed to fit onto the screen size.  If outsprog
                        496     ;              mode is not set, then only the portion of the sprite
                        497     ;              which is visible on the screen will be written.
                        498     ;
                        499     ;***************************************************************************
                        500
                        501
                        502
0307'  F5               503     put_sprite      push    af
0308'  D5               504                     push    de
0309'  E5               505                     push    hl
030A'  F5               506                     di
030B'  CD E252'          507                     call    check_id        ;validate sprite-id
030E'  38 00             508                     jr      c, p_id_ok      ;sprite-id ok, so jump
0310'  01 0005           509                     ld      bc, 0005        ;else return 05 in BC
0313'  C5 0348'          510                     jp      p_exit
0316'  CD 1259'          511     p_id_ok         call    spr_addr        ;get location of sprite record
0319'  36 02             512                     ld      (hl), 2         ;select stream 2
031B'  D5               513                     push    de
031C'  CD 1298           514                     call    SBLOC7
031F'  01               515                     pop     de
0320'  0E 46 05          516                     ld      l, (ix+sbitmap) ;hl -> sprite bitmap
0323'  DD 46 06          517                     ld      h, (ix+sbitmap+1)
0326'  42               518                     ld      b, d            ;b = row
0327'  48               519                     ld      c, e            ;c = col
0328'  DD 70 00          520                     ld      (ix+row), b     ;store sprite position
```

```
0338'  DD 71 01                 ld     (ix+col), c
033B'  3A 0F05                  ld     a, (backgcolor)
0331'  E6 F0                    and    11110000b
0333'  DD B6 06                 or     (ix+color)
0336'  32 9C80                  ld     (ATTR_B), a      ;set attr to screen background color
                                                        ;  and sprite foreground color
0339'  11 0100                  ld     de, 100h
033C'  B7                       and    a
033D'  ED 52                    sbc    hl, de
033F'  EB                       ex     de, hl           ;de -> first scan of sprite bitmap
                                                        ;  minus 100h, acts like (Cn00S)
0340'  2E 00                    ld     l, 0             ;l = row count
0343'  C5                       push   bc
0341'  E3           outer1      ex     (sp), hl
0344'  4D                       ld     c, l             ;c <- leftmost column of sprite
0345'  E3                       ex     (sp), hl
0346'  26 00                    ld     h, 0             ;h = column count
034E'  CD 02FE'     inner1      call   visible          ;is BC a legal screen location?
0348'  38 20                    jr     c, wrtchar       ;yes, so jump
034D'  3A 0FF6                  ld     a, (aflags)      ;is autoerase mode set?
0390'  CB 4F                    bit    autoersg, a
0393'  28 36                    jr     z, nextchar      ;if not autoerase mode then jump
0394'  3E 1F        col_loop    ld     a, max_cols-1    ;map current column onto screen
0396'  B8                       cp     b
0397'  30 0A                    jr     nc, chkrow1
0399'  79                       ld     a, c
0396'  D6 30                    sub    max_cols
039C'  4F                       ld     c, a
0350'  1E F5                    jr     col_loop
039F'  E9           chkrow1     push   hl
0360'  21 0F09                  ld     hl, screen_ht
0363'  7E                       ld     a, (hl)          ;map current row onto screen
0364'  3D                       dec    a
0565'  B8           row_loop    cp     b
0366'  30 07                    jr     nc, cont1
0368'  08                       ex     af, af'
0369'  78                       ld     a, b
0369'  96                       sub    (hl)
0368'  47                       ld     b, a
036C'  08                       ex     af, af'
0360'  18 F6                    jr     row_loop
036F'  E1           cont1       pop    hl
0370'  7C                       ld     a, h
0371'  B5                       or     l
0372'  20 06                    jr     nz, no_store
0374'  DD 70 00                 ld     (ix+row), b      ;first time through loops, store
0377'  DD 71 01                 ld     (ix+col), c      ;  erased screen location
0370'               no_store
0374'  C5           wrtchar     push   bc
0378'  C6 E31F                  call   GET_CH00         ;check if writing the char will
037E'  C1                       pop    bc               ;  overwrite something
0370'  FE 20                    cp     ' '
0381'  38 04                    jr     c, no_overwrite
0383'  DD CB 37 C8              set    overwrite, (ix+sflags)  ;save fact that something
                                                               ;  was overwritten
0387'  CD 0308'     no_overwrite call  display_char
0389'  0C           nextchar    inc    c                ;increment column position
0388'  24                       inc    h                ;increment column count
038C'  DD 7E 03                 ld     a, (ix+width)
038F'  BC                       cp     h
0390'  18 60                    jr     nz, inner1       ;jump if more columns to write
0391'  04                       inc    b                ;else increment row position
0393'  3C                       inc    l                ;increment row count
0394'  DD 7E 03                 ld     a, (ix+height)
0397'  BD                       cp     l
0356'  20 99                    jr     nz, outer1       ;jump if more rows to write
0304'  C1                       pop    bc
0396'  FB           e_exit      ei
039C'  E1                       pop    hl
0390'  D1                       pop    de
0398'  F1                       pop    af
0389'  DD CB 67 4E              bit    overwrite, (ix+sflags)
0303'  DD CB 07 EE              res    overwrite, (ix+sflags)
0107'  20 04                    jr     nz, ovrwrote     ;sprite overwrote something
0109'  01 0000                  ld     bc, 0            ;return 00 in BC, nothing overwritten
016C'  C9                       ret
0100'  01 0100      ovrwrote    ld     bc, 0100H
0180'  C9                       ret


;**************************************************************************
;
; ERASE_SPRITE
;
; Inputs: A = sprite-id
;
; Outputs: BC = 0000 - OK
;             = 0003 - illegal sprite-id
;
; Globals Read: sprite data structure
;               backgcolor
;               aflags
;               screen_ht
;               CH001
;
; Globals Written: sprite data structure
;                  ATTR_T
;                  C0005
;
; Procedures Called: check_id
;                    spr_addr
;                    SELECT
;                    visible
;                    display_char
;
; Procedures Called By: move_sprite
;                       I_P_HANDLER
;                       user program
;
; Description: this routine erases the specified sprite.  The screen
;              locations at which the sprite resides are replaced with
;              blanks.
;
;**************************************************************************
```

```
E3B1'  F5                         erase_sprite   push   af
E3B2'  D5                                        push   de
E3B3'  E5                                        push   hl
E3B4'  F3                                        di
E3B5'  CD E362'                                  call   check_id       ;validate sprite-id
E3B8'  30 06                                     jr     c, s_id_ok     ;sprite-id ok, so jump
E3BA'  01 0003                                   ld     bc, 0003       ;else return 03 in BC
E3BD'  C3 E421'                                  jp     e_exit
E3C0'  CD E35F'                   e_id_ok        call   spr_addr       ;get location of sprite record
E3C3'  3E 03                                     ld     a, 3           ;select stream 3
E3C5'  D5                                        push   de
E3C6'  CD 1335                                   call   SELECT
E3C9'  D1                                        pop    de
E3CA'  DD 46 00                                  ld     b, (ix+row)    ;b = row
E3CD'  DD 4E 01                                  ld     c, (ix+col)    ;c = col
E3D0'  3A DFEE                                   ld     a, (backgcolor)
E3D3'  32 5C8F                                   ld     (ATTR_T), a    ;set attr to screen background color
E3D6'  2E 00                                     ld     l, 0           ;l = row count
E3D8'  C5                                        push   bc
E3D9'  E3                         outer2         ex     (sp), hl
E3DA'  4E                                        ld     c, l           ;c <- leftmost column of sprite
E3DB'  E3                                        ex     (sp), hl
E3DC'  26 00                                     ld     h, 0           ;h = column count
E3DE'  CD E2FE'                   inner2         call   visible        ;is BC a legal screen location?
E3E1'  38 33                                     jr     c, era_char    ;yes, so jump
E3E3'  3A DFFE                                   ld     a, (gflags)    ;is autoerase mode set?
E3E6'  CB 4F                                     bit    autoerase, a
E3E8'  28 23                                     jr     z, nxtchar     ;if not autoerase mode then jump
E3EA'  3E 1F                      c_loop         ld     a, scr_cols-1  ;map current column onto screen
E3EC'  B9                                        cp     c
E3ED'  30 06                                     jr     nc, chrow2
E3EF'  79                                        ld     a, c
E3F0'  D6 20                                     sub    scr_cols
E3F2'  4F                                        ld     c, a
E3F3'  18 F5                                     jr     c_loop
E3F5'  E5                         chrow2         push   hl
E3F6'  21 DFF9                                   ld     hl, screen_ht
E3F9'  7E                                        ld     a, (hl)        ;map current row onto screen
E3FA'  3D                                        dec    a
E3FB'  B8                         r_loop         cp     b
E3FC'  30 07                                     jr     nc, cont2
E3FE'  08                                        ex     af, af'
E3FF'  78                                        ld     a, b
E400'  96                                        sub    (hl)
E401'  47                                        ld     b, a
E402'  08                                        ex     af,af'
E403'  18 F6                                     jr     r_loop
E405'  E1                         cont2          pop    hl
E406'  ED 5B 5C36                 era_char       ld     de, (CHARS)    ;de -> bitmap of blank char
E40A'  CD E2D8'                                  call   display_char
E40D'  0C                         nxtchar        inc    c              ;increment column position
E40E'  24                                        inc    h              ;increment column count
E40F'  DD 7E 02                                  ld     a, (ix+width)
E412'  BC                                        cp     h
E413'  20 C9                                     jr     nz, inner2     ;jump if more columns to write
E415'  04                                        inc    b              ;else increment row position
E416'  2C                                        inc    l              ;increment row count
E417'  DD 7E 03                                  ld     a, (ix+height)
E41A'  BD                                        cp     l
E41B'  20 BC                                     jr     nz, outer2     ;jump if more rows to write
E41D'  C1                                        pop    bc             ;else done
E41E'  01 0000                                   ld     bc, 0          ;return 00 in BC, nothing overwritten
E421'  FB                         e_exit         ei
E422'  E1                                        pop    hl
E423'  D1                                        pop    de
E424'  F1                                        pop    af
E425'  C9                                        ret

;************************************************************************
;
; MOVE_SPRITE
;
; Inputs: A = sprite-id
;         D = relative vertical motion
;         E = relative horizontal motion
;
; Outputs: BC = 0000 - OK, nothing overwritten
;             = 0100 - OK, something overwritten
;             = 0003 - illegal sprite-id
;
; Globals Used: screen_ht
;               gflags
;
; Globals Written: sprite data structure
;
; Procedures Called: erase_sprite
;                    put_sprite
;
; Procedures Called By: I_F_HANDLER
;                       user program
;
; Description: this routine moves the specified sprite by the specified
;              amount.  It is equivalent to Erase_Sprite followed by
;              Put_Sprite.
;
;************************************************************************
```

```
8426'  83                    736   move_sprite     push    hl
8427'  F5                    737                   push    af
8428'  CD 8381'              738                   call    erase_sprite
842B'  CB 41                 739                   bit     0, c            ;if an error occurred, exit
842D'  20 1E                 740                   jr      nz, m_exit
842F'  DD 7E 00              741                   ld      a, (ix+row)     ;update sprite position
8432'  82                    742                   add     a, d
8433'  21 0FF8               743                   ld      hl, sflags
8436'  CB 4E                 744                   bit     outscreen, (hl)
8438'  28 08                 745                   jr      z, row_ok
843A'  CB 7F                 746                   bit     7, a            ;is row negative?
843C'  28 04                 747                   jr      z, row_ok       ;no, so jump
843E'  21 0F89               748                   ld      hl, screen_ht   ;yes, so add it to (screen_ht)
8441'  8E                    749                   add     a, (hl)
8442'  57                    750   row_ok          ld      d, a
8443'  DD 7E 01              751                   ld      a, (ix+col)
8446'  83                    752                   add     a, e
8447'  5F                    753                   ld      e, a
8448'  F1                    754                   pop     af
8449'  F5                    755                   push    af
844A'  CD 8107'              756                   call    put_sprite
844D'  F1                    757   m_exit          pop     af
844E'  E1                    758                   pop     hl
844F'  C9                    759                   ret
                             760
                             761
                             762   ;*****************************************************************************
                             763   ;
                             764   ; CH_LOC
                             765   ;
                             766   ; Inputs: A = sprite-id
                             767   ;         D = new row
                             768   ;         E = new col
                             769   ;
                             770   ; Outputs: BC = 0000 - OK
                             771   ;              = 0003 - illegal sprite-id
                             772   ;
                             773   ; Globals Read: none.
                             774   ;
                             775   ; Globals Written: sprite data structure
                             776   ;
                             777   ; Procedures Called: check_id
                             778   ;                    spr_addr
                             779   ;
                             780   ; Procedures Called By: I_P_HANDLER
                             781   ;                       user_progrm
                             782   ;
                             783   ; Description: this routine updates the row and column in the record for
                             784   ;              the sprite identified by sprite-id.
                             785   ;
                             786   ;*****************************************************************************
                             787
                             788
8450'  CD 8282'              789   ch_loc          call    check_id        ;validate sprite-id
8453'  38 04                 790                   jr      c, ch_l_id_ok   ;if id is ok then jump
8455'  01 0003               791                   ld      bc, 0003        ;else return 03 in BC
8458'  C9                    792                   ret
8459'  CD 829F'              793   ch_l_id_ok      call    spr_addr        ;ix -> loc of sprite record
845C'  DD 72 00              794                   ld      (ix+row), d
845F'  DD 73 01              795                   ld      (ix+col), e
8462'  01 0000               796                   ld      bc, 00          ;return 00 in BC
8465'  C9                    797                   ret
                             798
                             799
                             800   ;*****************************************************************************
                             801   ;
                             802   ; CH_CLR
                             803   ;
                             804   ; Inputs: A = sprite-id
                             805   ;         D = new color
                             806   ;
                             807   ; Outputs: BC = 0000 - OK
                             808   ;              = 0003 - illegal sprite-id
                             809   ;              = 0005 - illegal color
                             810   ;
                             811   ; Globals Read: none.
                             812   ;
                             813   ; Globals Written: sprite data structure
                             814   ;
                             815   ; Procedures Called: check_id
                             816   ;                    spr_addr
                             817   ;
                             818   ; Procedures Called By: I_P_HANDLER
                             819   ;                       user_progrm
                             820   ;
                             821   ; Description: this routine puts a new color into the record for the
                             822   ;              sprite identified by sprite-id.
                             823   ;
                             824   ;*****************************************************************************
                             825
                             826
8466'  CD 8282'              827   ch_clr          call    check_id        ;validate sprite-id
8469'  38 03                 828                   jr      c, ch_c_id_ok   ;if id is ok then jump
846B'  01 0003               829                   ld      bc, 0003        ;else return 03 in BC
846E'  F5                    830   ch_c_id_ok      push    af              ;validate color
846F'  1E 07                 831                   ld      e, 7
8471'  BB                    832                   cp      d
8472'  30 05                 833                   jr      nc, ch_c_color_ok       ;jump if color ok
8474'  F1                    834                   pop     af
8475'  01 0005               835                   ld      bc, 0005        ;else return 05 in BC
8478'  C9                    836                   ret
8479'  F1                    837   ch_c_color_ok   pop     af
847A'  CD 829F'              838                   call    spr_addr
847D'  DD 72 04              839                   ld      (ix+color), d
8480'  01 0000               840                   ld      bc, 0           ;return 00 in BC
8483'  C9                    841                   ret
                             842
```

```
                        ;*****************************************************************
                        ;
                        ; OVERLAY
                        ;
                        ; Inputs: D = sprite-id1
                        ;         E = sprite-id2
                        ;
                        ; Outputs: BC = 0000 - OK, no overlap
                        ;              = 0200 - OK, overlap
                        ;              = 0003 - illegal sprite-id
                        ;
                        ; Globals Read: sprite data structure
                        ;
                        ; Globals Written: none.
                        ;
                        ; Procedures Called: check_id
                        ;                    spr_addr
                        ;
                        ; Procedure Called By: upd_sole
                        ;                      upd_rose
                        ;                      I_P_HANDLER
                        ;                      user program
                        ;
                        ; Description: this routine checks if two sprites overlap on the screen.
                        ;              This is done by comparing the ranges covered by each sprite
                        ;              in the x- and y-directions.  If there is overlap in both
                        ;              directions, then the sprites must overlap.  The following
                        ;              diagram defines terms mentioned below:
                        ;
                        ;      (ulx1, uly1)
                        ;        +-------------------------+
                        ;        |                         |
                        ;        |    (ulx2, uly2)         |
                        ;        |      +------------------|------+
                        ;        |      |                  |      |
                        ;        |      |                  |      |
                        ;        |      |                  |      |
                        ;        |      |                  |      |
                        ;        +------|------------------+      |
                        ;               |                (lrx1, lry1)
                        ;               |                         |
                        ;               |                         |
                        ;               +-------------------------+
                        ;                                   (lrx2, lry2)
                        ;
                        ;*****************************************************************

8484' 00 85             overlap:        push    ix
8486' F5                                push    af
8487' C5                                push    bc
8488' E5                                push    hl
8489' 7A                                ld      a, d                    ;validate sprite-id1
848A' CD 8282'                          call    check_id
848D' 38 04                             jr      c, o_ok1
848F' 01 0003           o_id_ng:        ld      bc, 0003                ;illegal sprite-id, return 3 in BC
8492' C5                                ret
8493' 42                o_ok1:          ld      b, d                    ;save sprite-id1 in B
8494' 48                                ld      c, e
8495' 78                                ld      a, e                    ;validate sprite-id2
8496' CD 8181'                          call    check_id
8499' 30 F4                             jr      nc, o_id_ng
                                                                        ;here if both sprite-ids ok
849B' 78                                ld      a, b                    ;check if there is any overlap in
                                                                        ;  the x-direction
849C' CD 8198'                          call    spr_addr
849F' 00 5A 01                          ld      d, (ix+xcol)
84A1' DD 3E 02                          ld      e, (ix+width)
84A5' 7B                                ld      a, d
84A6' CD 8136'                          call    spr_addr
84A9' DC 69 01                          ld      h, (ix+xcol)
84AC' DC 9E 02                          ld      l, (ix+width)
84AF' 7A                                ld      a, d
84B0' 93                                sub     e
84B1' 3D                                dec     a                       ;a = lrx1
84B2' BC                                cp      h                       ;h = ulx2
84B3' 38 18                             jr      c, no_coll              ;if ulx2 > lrx1, then no overlap
84B5' 7C                                ld      a, h
84B6' 85                                add     a, l
84B7' 3E                                dec     a                       ;a = lrx2
84B8' BA                                cp      d                       ;d = ulx1
84B9' 38 25                             jr      c, no_coll              ;if ulx1 > lrx1, then no overlap
84BB' 78                                ld      a, b                    ;check if there is any overlap in the
                                                                        ;  y-direction
84BC' CD 8198'                          call    spr_addr
84BF' DC 5E 08                          ld      d, (ix+row)
84C2' DC 5E 03                          ld      e, (ix+height)
84C5' 7B                                ld      a, d
84C6' CD 8136'                          call    spr_addr
84C9' DC 6E 08                          ld      h, (ix+row)
84CC' DC 9E 03                          ld      l, (ix+height)
84CF' 7A                                ld      a, d
84D0' 83                                add     a, e
84D1' 3D                                dec     a                       ;a = lry1
84D2' BC                                cp      h                       ;h = uly2
84D3' 38 09                             jr      c, no_coll              ;if lry1 < uly2, then no overlap
84D5' 7C                                ld      a, h
84D6' 83                                add     a, l
84D7' 3D                                dec     a                       ;a = lry2
84D8' BB                                cp      e                       ;e = uly1
84D9' 3A 05                             jr      c, no_coll              ;if lry2 < uly1, then no overlap
84DB' 01 0200           yes_coll:       ld      bc, 0100H               ;return BC, overlap
84DE' 18 03                             jr      o_exit
84E0' 01 0000           no_coll:        ld      bc, 0                   ;return BC, no overlap
84E3' E1                o_exit:         pop     hl
84E4' C1                                pop     bc
84E5' F1                                pop     af
84E6' DD E1                             pop     ix
84E8' C9                                ret
```

```
                 ;**************************************************************
                 ;
                 ; INITSCREEN
                 ;
                 ; Inputs: A = screen-height
                 ;         D = background-color
                 ;         E = border-color
                 ;
                 ; Outputs: BC = 0000 - OK
                 ;             = 0005 - illegal color
                 ;             = 0007 - illegal screen-height
                 ;
                 ; Globals Read: ATTR_P
                 ;               TVFLAG
                 ;               screen_ht
                 ;
                 ; Globals Written: screen_ht
                 ;                  ATTR_P
                 ;                  backgcolor
                 ;                  BORDCR
                 ;                  TVFLAG
                 ;
                 ; Procedures Called: CLS_B
                 ;
                 ; Procedures Called By: I_P_HANDLER
                 ;                       user programs
                 ;
                 ; Description: this routine sets the screen_ht variable to the specified
                 ;              value.  If screen-height is not in the range 0 - 24, 07
                 ;              is returned in BC.  If background-color or border-color is
                 ;              not in the range 0 - 7, 05 is returned in BC.  The top
                 ;              screen-height lines of the screen are cleared and set to
                 ;              the specified screen background color.  The remaining
                 ;              lines of the screen are cleared and set to the specified
                 ;              border color.
                 ;
                 ;**************************************************************


E4E9'  FE 19     initscreen    cp      max_rows+1       ;validate screen-height
E4EB'  30 04                   jr      c, sh_ok         ;if screen-height ok, then jump
E4ED'  01 0007                 ld      bc, 0007         ;else return 07 in BC
E4F0'  C9                      ret
E4F1'  32 0FF9'  sh_ok         ld      (screen_ht), a   ;store screen-height
E4F4'  F5                      push    af
E4F5'  3E 07                   ld      a, 7             ;validate background-color
E4F7'  BA                      cp      d
E4F8'  30 05                   jr      nc, bc_ok        ;if background-color ok, then jump
E4FA'  01 0005                 ld      bc, 0005         ;else return 05 in BC
E4FD'  F1                      pop     af
E4FE'  C9                      ret
E4FF'  BB        bc_ok         cp      e                ;validate border-color
E500'  30 05                   jr      nc, bdr_ok       ;if border-color ok, then jump
E502'  01 0005                 ld      bc, 0005         ;else return 05 in BC
E505'  F1                      pop     af
E506'  C9                      ret
E507'  3A 5C8D   bdr_ok        ld      a, (ATTR_P)
E50A'  E6 C7                   and     11000111B        ;clear current background color
E50C'  CB 22                   sla     d
E50E'  CB 22                   sla     d
E510'  CB 22                   sla     d
E512'  B2                      or      d                ;set new background color
E513'  32 5C8D                 ld      (ATTR_P), a
E516'  32 0FF8'                ld      (backgcolor), a
E519'  7B                      ld      a, e             ;set new border color
E51A'  D3 FE                   out     (BORDPT), a      ;change the border
E51C'  CB 27                   sla     a
E51E'  CB 27                   sla     a
E520'  CB 27                   sla     a
E522'  CB 6F                   bit     5, a             ;set the appropriate ink color
E524'  20 02                   jr      nz, black_ink    ;if light border, then black ink
E526'  EE 07                   xor     7                ;else white ink
E528'  32 5C48   black_ink     ld      (BORDCR), a
E52B'  D5                      push    de
E52C'  E5                      push    hl
E52D'  21 5C3C                 ld      hl, TVFLAG
E530'  7E                      ld      a, (hl)          ;save (TVFLAG)
E531'  F5                      push    af
E532'  CB 86                   res     0, (hl) ;clear top part of screen
E534'  06 18                   ld      b, max_rows
E536'  E5                      push    hl
E537'  CD 0D7F                 call    CLS_B
E53A'  E1                      pop     hl
E53B'  3A 0FF9'                ld      a, (screen_ht)   ;clear bottom part of screen
E53E'  4F                      ld      c, a
E53F'  78                      ld      a, b
E540'  91                      sub     c
E541'  47                      ld      b, a             ;b <- max_rows - (screen_ht)
E542'  28 03                   jr      z, no_clear
E544'  CB C6                   set     0, (hl)
E546'  CD 0D7F                 call    CLS_B
E549'  F1        no_clear      pop     af
E54A'  32 5C3C                 ld      (TVFLAG), a      ;restore (TVFLAG)
E54D'  E1                      pop     hl
E54E'  D1                      pop     de
E54F'  F1                      pop     af
E550'  01 0000                 ld      bc, 0            ;return 0 in BC
E553'  C9                      ret
```

```
                      ;************************************************************************
                      ;
                      ; UPD_COLS
                      ;
                      ; Inputs: a = column adjustment (1 for right scroll, -1 for left scroll)
                      ;
                      ; Outputs: none.
                      ;
                      ; Globals Read: max_sprites
                      ;               screen_ht
                      ;               gflags
                      ;
                      ; Globals Written: max_sprites
                      ;                  sprite data structure
                      ;                  scratchpad
                      ;
                      ; Procedures Called: spr_addr
                      ;                    overlap
                      ;                    out_sprite
                      ;
                      ; Procedures Called By: h_scroll
                      ;
                      ; Description: this routine updates the column position of all sprites.
                      ;              It also checks if, as a result of the horizontal scroll,
                      ;              each sprite should be written to the screen.  This is done
                      ;              by creating two dummy sprites (sprite-id is max_sprites)
                      ;              whose size and location correspond to the right and left
                      ;              columns of the screen respectively.  If a given sprite
                      ;              overlaps either of these dummy sprites, it is written
                      ;              to the screen.
                      ;
                      ;************************************************************************


E554'  DD E5          upd_cols   push  ix
E556'  C5                        push  bc
E557'  D5                        push  de
E558'  E5                        push  hl
E559'  67                        ld    h, a            ;save column adjustment in h
E55A'  3A 0FFA                   ld    a, (max_sprites)
E55D'  3C                        inc   a               ;temporarily increment max_sprites
E55E'  32 0FFA                   ld    (max_sprites), a ; to allow for the dummy sprite
E561'  3D                        dec   a
E562'  5F                        ld    e, a            ;e = sprite-id of the dummy sprite
E563'  DD 21 0FE3                ld    ix, scratchpad  ;initialize the dummy sprite
E567'  DD 36 00 00               ld    (ix+row), 0
E56B'  DD 36 07 01               ld    (ix+sflags), 1
E56F'  DD 36 02 01               ld    (ix+width), 1
E573'  F5                        push  af
E574'  3A 0FF9                   ld    a, (screen_ht)
E577'  3D                        dec   a
E578'  DD 77 03                  ld    (ix+height), a
E57B'  F1                        pop   af
E57C'  DD 36 01 1F               ld    (ix+col), max_cols-1
E580'  3D             updc_loop  dec   a               ;a = id of current sprite
E581'  CD E25F'                  call  spr_addr        ;update column position
E584'  6F                        ld    l, a
E585'  DD 7E 01                  ld    a, (ix+col)
E588'  84                        add   a, h
E589'  E5                        push  hl
E58A'  21 0FFE                   ld    hl, gflags
E58D'  CB 4E                     bit   automap, (hl)
E58F'  E1                        pop   hl
E590'  28 0D                     jr    z, uc_cont      ;if not automap mode, then jump
E592'  FE 20                     cp    max_cols
E594'  20 03                     jr    nz, uc1         ;if column <> max_cols then jump
E596'  AF                        xor   a               ;else column <- 0
E597'  18 06                     jr    uc_cont         ;continue
E599'  FE FF          uc1        cp    -1
E59B'  20 02                     jr    nz, uc_cont     ;if column <> -1, then jump
E59D'  3E 1F                     ld    a, max_cols-1   ;else column <- max_cols-1
E59F'  DD 77 01       uc_cont    ld    (ix+col), a     ;don't need to erase position as this
E5A2'  55                        ld    d, l            ; is done by out_sprite
E5A3'  CD E686'                  call  overlap
E5A6'  CB 48                     bit   1, b
E5A8'  7D                        ld    a, l
E5A9'  20 13                     jr    nz, wrt_spr
E5AB'  DD E5                     push  ix
E5AD'  DD 21 0FE3                ld    ix, scratchpad
E5B1'  DD 36 01 00               ld    (ix+col), 0
E5B5'  DD E1                     pop   ix
E5B7'  CD E686'                  call  overlap
E5BA'  CB 48                     bit   1, b
E5BC'  28 0B                     jr    z, next_spr     ;if no overlap, go on to the next one
E5BE'  6F             wrt_spr    ld    l, a
E5BF'  DD 5E 00                  ld    d, (ix+row)
E5C2'  DD 5E 01                  ld    a, (ix+col)
E5C5'  CD E307'                  call  out_sprite      ;overlaps, so write sprite to screen
E5C8'  7D                        ld    a, l
E5C9'  A7             next_spr   and   a
E5CA'  20 B4                     jr    nz, updc_loop
E5CC'  21 0FFA                   ld    hl, max_sprites
E5CF'  35                        dec   (hl)            ;restore max_sprites to its original
                                                       ; value
E5D0'  E1                        pop   hl
E5D1'  D1                        pop   de
E5D2'  C1                        pop   bc
E5D3'  DD E1                     pop   ix
E5D5'  C9                        ret
```

```
                ;****************************************************************
                ;
                ; H_SCROLL
                ;
                ; Inputs: A < 0 - left scroll
                ;         A >= 0 - right scroll
                ;
                ; Outputs: BC = 0000 - OK
                ;
                ; Globals Read: screen_ht
                ;               backgcolor
                ;
                ; Globals Written: none.
                ;
                ; Procedures Called: upd_cols
                ;
                ; Procedures Called By: I_P_HANDLER
                ;                       user program
                ;
                ; Description: this routine causes the contents of the top (screen_ht)
                ;              lines of the screen to be scrolled left or right one
                ;              column, depending on the value of A.  All sprites have
                ;              their column position updated by 1 or -1, depending on
                ;              the direction of scroll.
                ;
                ;****************************************************************


E5D6'  F5       h_scroll     push   af
E5D7'  D5                    push   de
E5D8'  E5                    push   hl
E5D9'  D9                    exx
E5DA'  C5                    push   bc
E5DB'  D5                    push   de
E5DC'  E5                    push   hl
E5DD'  D9                    exx
E5DE'  F5                    di
E5DF'  47                    ld     b, a             ;save direction
E5E0'  3A 0FF9               ld     a, (screen_ht)   ;calc. the number of scan lines
E5E3'  CB 27                 sla    a                ;  to be updated
E5E5'  CB 27                 sla    a
E5E7'  CB 27                 sla    a
E5E9'  CB 78                 bit    7, b
E5EB'  28 45                 jr     z, right_scroll  ;if A >=0, then jump
E5ED'  D9       left_scroll  exx                     ;use alt. regs. for attr pointers
E5EE'  21 5801               ld     hl, 5801H
E5F1'  11 5800               ld     de, 5800H
E5F4'  D9                    exx
E5F5'  3A 0FF9               ld     a, (screen_ht)
E5F8'  47                    ld     b, a             ;b = line counter
E5F9'  21 E6E6'              ld     hl, left_table
E5FC'  E5                    push   hl
E5FD'  E1       left_outer   pop    hl
E5FE'  5E                    ld     e, (hl)
E5FF'  23                    inc    hl
E600'  56                    ld     d, (hl)          ;de -> dest for next line
E601'  23                    inc    hl
E602'  E5                    push   hl
E603'  62                    ld     h, d
E604'  6B                    ld     l, e
E605'  23                    inc    hl               ;hl -> source for next line
E606'  3E 08                 ld     a, 8             ;a = scan counter
E608'  C5       left_inner   push   bc
E609'  01 001F               ld     bc, max_cols-1
E60C'  ED B0                 ldir                    ;move scan left
E60E'  EB                    ex     de, hl
E60F'  36 00                 ld     (hl), 0          ;clear last column
E611'  01 00E2               ld     bc, 256-(max_cols-2)
E614'  09                    add    hl, bc           ;hl -> start of next scan + one byte
E615'  54                    ld     d, h
E616'  5D                    ld     e, l
E617'  1B                    dec    de               ;de -> start of next scan
E618'  3D                    dec    a
E619'  20 EE                 jr     nz, left_inner   ;more scans to do, so jump
E61B'  D9                    exx
E61C'  01 001F               ld     bc, max_cols-1
E61F'  ED B0                 ldir                    ;move attributes
E621'  3A 0FFE               ld     a, (backgcolor)  ;clear rightmost column to permanent
E624'  12                    ld     (de), a          ;  attribute
E625'  23                    inc    hl               ;hl -> second byte of next line
E626'  13                    inc    de               ;de -> first byte of next line
E627'  D9                    exx
E628'  C1                    pop    bc
E629'  10 D2                 djnz   left_outer       ;if more lines to do, then jump
E62B'  E1                    pop    hl
E62C'  1E FF                 ld     e, -1            ;set parameter for upd_cols
E62E'  18 46                 jr     hs_cont
E630'  D9       right_scroll exx                     ;use alt. regs. for attr pointers
E631'  21 581E               ld     hl, 581EH
E634'  11 581F               ld     de, 581FH
E637'  D9                    exx
E638'  3A 0FF9               ld     a, (screen_ht)
E63B'  47                    ld     b, a             ;b = line counter
E63C'  21 E6B6'              ld     hl, rt_table
E63F'  E5                    push   hl
E640'  E1       rt_outer     pop    hl
E641'  5E                    ld     e, (hl)
E642'  23                    inc    hl
E643'  56                    ld     d, (hl)          ;de -> dest for next line
E644'  23                    inc    hl
E645'  E5                    push   hl
E646'  62                    ld     h, d
E647'  6B                    ld     l, e
E648'  2B                    dec    hl               ;hl -> source for next line
E649'  3E 08                 ld     a, 8             ;a = scan counter
E64B'  C5                    push   bc
E64C'  01 001F  rt_inner     ld     bc, max_cols-1
E64F'  ED B8                 lddr                    ;move scan right
```

```
E651'  EB                              ex      de, hl
E652'  36 00                           ld      (hl), 0         ;clear first column
E654'  01 011E                         ld      bc, 256+(num_cols-2)
E657'  09                              add     hl, bc          ;hl -> end of next scan - 1 byte
E658'  54                              ld      d, h
E659'  5D                              ld      e, l
E65A'  13                              inc     de              ;de -> end of next scan
E65B'  3D                              dec     a
E65C'  20 EE                           jr      nz, rt_inner    ;more scans to do, so jump
E65E'  D9                              exx
E65F'  01 001F                         ld      bc, 31
E662'  ED B8                           lddr                    ;move attributes
E664'  3A DFF6                         ld      a, (backcolor)  ;clear leftmost column to background
E667'  12                              ld      (de), a         ;  attribute
E668'  01 005F                         ld      bc, 3*num_cols-1
E66B'  09                              add     hl, bc          ;hl -> second to last byte of next line
E66C'  54                              ld      d, h
E66D'  5D                              ld      e, l
E66E'  13                              inc     de              ;de -> last byte of next line
E66F'  D9                              exx
E670'  C1                              pop     bc
E671'  10 C0                           djnz    rt_outer        ;more lines to do, so jump
E673'  E1                              pop     hl
E674'  1E 01                           ld      e, 1            ;set parameter for upd_cols
E676'  CD E554'        no_cont         call    upd_cols
E679'  FB                              ei
E67A'  08                              ex      af, af'
E67B'  E1                              pop     hl
E67C'  D1                              pop     de
E67D'  C1                              pop     bc
E67E'  D9                              exx
E67F'  E1                              pop     hl
E680'  D1                              pop     de
E681'  F1                              pop     af
E682'  11 0000                         ld      de, 0
E685'  C9                              ret


                       ;This table contains the destination address for each row for scrolling left

E686'  4000            left_table      defw    4000H
E688'  4020                            defw    4020H
E68A'  4040                            defw    4040H
E68C'  4060                            defw    4060H
E68E'  4080                            defw    4080H
E690'  40A0                            defw    40A0H
E692'  40C0                            defw    40C0H
E694'  40E0                            defw    40E0H
E696'  4800                            defw    4800H
E698'  4820                            defw    4820H
E69A'  4840                            defw    4840H
E69C'  4860                            defw    4860H
E69E'  4880                            defw    4880H
E6A0'  48A0                            defw    48A0H
E6A2'  48C0                            defw    48C0H
E6A4'  48E0                            defw    48E0H
E6A6'  5000                            defw    5000H
E6A8'  5020                            defw    5020H
E6AA'  5040                            defw    5040H
E6AC'  5060                            defw    5060H
E6AE'  5080                            defw    5080H
E6B0'  50A0                            defw    50A0H
E6B2'  50C0                            defw    50C0H
E6B4'  50E0                            defw    50E0H


                       ;This table contains the destination address for each row for scrolling right

E6B6'  401F            rt_table        defw    401FH
E6B8'  403F                            defw    403FH
E6BA'  405F                            defw    405FH
E6BC'  407F                            defw    407FH
E6BE'  409F                            defw    409FH
E6C0'  40BF                            defw    40BFH
E6C2'  40DF                            defw    40DFH
E6C4'  40FF                            defw    40FFH
E6C6'  481F                            defw    481FH
E6C8'  483F                            defw    483FH
E6CA'  485F                            defw    485FH
E6CC'  487F                            defw    487FH
E6CE'  489F                            defw    489FH
E6D0'  48BF                            defw    48BFH
E6D2'  48DF                            defw    48DFH
E6D4'  48FF                            defw    48FFH
E6D6'  501F                            defw    501FH
E6D8'  503F                            defw    503FH
E6DA'  505F                            defw    505FH
E6DC'  507F                            defw    507FH
E6DE'  509F                            defw    509FH
E6E0'  50BF                            defw    50BFH
E6E2'  50DF                            defw    50DFH
E6E4'  50FF                            defw    50FFH
```

```
                    ;*****************************************************************
                    ;
                    ; UPD_ROWS
                    ;
                    ; Inputs: a = row adjustment (1 for scroll down, -1 for scroll up)
                    ;
                    ; Outputs: none.
                    ;
                    ; Globals Read: num_sprites
                    ;               screen_ht
                    ;               gflags
                    ;
                    ; Globals written: num_sprites
                    ;                  sprite data structure
                    ;                  spritehead
                    ;
                    ; Procedures Called: spr_addr
                    ;                    overlap
                    ;                    put_sprite
                    ;
                    ; Procedures Called By: v_scroll
                    ;
                    ; Description: this routine updates the row position of all sprites.
                    ;              It also checks if, as a result of the vertical scroll,
                    ;              each sprite should be written to the screen.  This is done
                    ;              by creating two dummy sprites (sprite-id is num_sprites)
                    ;              whose size and location correspond to the top and bottom
                    ;              rows of the screen respectively.  If a given sprite
                    ;              overlaps either of these dummy sprites, it is written
                    ;              to the screen.
                    ;
                    ;*****************************************************************

E6E6'  DD E5        upd_rows    push   ix
E6E8'  C5                       push   bc
E6E9'  D5                       push   de
E6EA'  E5                       push   hl
E6EB'  67                       ld     h, a              ;save row adjustment in h
E6EC'  3A 0FF6                  ld     a, (num_sprites)
E6EF'  3C                       inc    a                 ;temporarily increment num_sprites
E6F0'  32 0FF6                  ld     (num_sprites), a  ; to allow for the dummy sprite
E6F3'  3D                       dec    a
E6F4'  5F                       ld     e, a              ;e = sprite-id of the dummy sprite
E6F5'  DD 21 0FE3               ld     ix, spritehead    ;initialize the dummy sprite
E6F9'  DD 36 01 00              ld     (ix+col), 0
E6FD'  DD 36 07 01              ld     (ix+flags), 1
E701'  DD 36 02 20              ld     (ix+width), max_cols
E705'  DD 36 03 01              ld     (ix+height), 1
E709'  F5                       push   af
E70A'  3A 0FF9                  ld     a, (screen_ht)
E70D'  3D                       dec    a
E70E'  DD 77 00                 ld     (ix+row), a
E711'  F1                       pop    af
E712'  3C           updr_loop   dec    a                 ;a = id of current sprite
E713'  CD E25F'                 call   spr_addr          ;update column position
E716'  6F                       ld     l, a
E717'  DD 7E 00                 ld     a, (ix+row)
E71A'  84                       add    a, h
E71B'  E5                       push   hl
E71C'  21 0FFE                  ld     hl, gflags
E71F'  CB 4E                    bit    autowrap, (hl)
E721'  E1                       pop    hl
E722'  28 16                    jr     z, ur_cont        ;if not autowrap mode, then jump
E724'  E5                       push   hl
E725'  21 0FF9                  ld     hl, screen_ht
E728'  BE                       cp     (hl)
E729'  E1                       pop    hl
E72A'  20 03                    jr     nz, ur1           ;if row <> screen_ht then jump
E72C'  AF                       xor    a                 ;else row <- 0
E72D'  18 08                    jr     ur_cont           ;continue
E72F'  FE FF        ur1         cp     -1
E731'  20 07                    jr     nz, ur_cont       ;if row <> -1, then jump
E733'  E5                       push   hl
E734'  21 0FF9                  ld     hl, screen_ht
E737'  7E                       ld     a, (hl)           ;else row <- screen_ht-1
E738'  3D                       dec    a
E739'  E1                       pop    hl
E73A'  DD 77 00     ur_cont     ld     (ix+row), a
E73D'  55                       ld     d, 1
E73E'  CD E6B6'                 call   overlap
E741'  CB 48                    bit    1, b
E743'  7D                       ld     a, 1
E744'  20 13                    jr     nz, wt_spr
E746'  DD E5                    push   ix
E748'  DD 21 0FE3               ld     ix, spritehead
E74C'  DD 36 00 00              ld     (ix+row), 0
E750'  DD E1                    pop    ix
E752'  CD E6B6'                 call   overlap
E755'  CB 48                    bit    1, b
E757'  28 08                    jr     z, no_spr         ;if no overlap, go on to the next one
E759'  6F           wt_spr      ld     l, a
E75A'  DD 56 00                 ld     d, (ix+row)
E75D'  DD 5E 01                 ld     e, (ix+col)
E760'  CD 0307'                 call   put_sprite        ;overlaps, so write sprite to screen
E763'  7D                       ld     a, l
E764'  A7           no_spr      and    a
E765'  20 AE                    jr     nz, updr_loop
E767'  21 0FFA                  ld     hl, num_sprites
E76A'  35                       dec    (hl)              ;restore num_sprites to its original
                                                         ; value

E76B'  E1                       pop    hl
E76C'  D1                       pop    de
E76D'  C1                       pop    bc
E76E'  DD E1                    pop    ix
E770'  C9                       ret
```

```
                  ;****************************************************************
                  ;
                  ; V_SCROLL
                  ;
                  ; Inputs: A < 0 - left scroll
                  ;         A >= 0 - right scroll
                  ;
                  ; Outputs: BC = 0000 - OK
                  ;
                  ; Globals Read: screen_ht
                  ;               backcolor
                  ;
                  ; Globals Written: ATTR_T
                  ;
                  ; Procedures Called: SET_PRINT_POS
                  ;                    WRITE_CHAR
                  ;
                  ; Procedures Called By: I_O_HANDLER
                  ;                       user program
                  ;
                  ; Description: this routine causes the contents of the top (screen_ht)
                  ;              lines of the screen to be scrolled left or right one
                  ;              column, depending on the value of A.  All sprites have
                  ;              their column position updated by 1 or -1, depending on
                  ;              the direction of scroll.
                  ;
                  ;****************************************************************


8771'  F5         v_scroll     push  af
8771'  D5                      push  de
8775'  E5                      push  hl
8774'  D9                      exx
8775'  C5                      push  bc
8776'  D5                      push  de
8777'  E5                      push  hl
8778'  D9                      exx
8779'  F5                      ei
877A'  C6 7F                   bit   7, a
E77C'  1E 59                   jr    z, down_scroll   ;if A >=0, then jump
877E'  D9         up_scroll    exx                    ;use alt. regs. for attr pointers
877F'  21 5810                 ld    hl, 5810H
E7A2'  11 5600                 ld    de, 5800H
E785'  D9                      exx
87AA'  5A DFF9                 ld    a, (screen_ht)
E789'  5C                      dec   a
E7A0'  47                      ld    b, a             ;b = line counter
8788'  21 E560'                ld    hl, vs_table
8788'  E5                      push  hl
E78F'  E1         up_outer     pop   hl
E790'  58                      ld    e, (hl)
8791'  25                      inc   hl
E792'  5A                      ld    d, (hl)          ;de -> dest for next line
E795'  23                      inc   hl
8794'  D5                      push  de               ;put dest pointer on stack
E795'  5E                      ld    e, (hl)
E79b'  25                      inc   hl
E797'  56                      ld    d, (hl)          ;de -> source for next line
E798'  2A                      dec   hl
8799'  F5                      ex    (sp), hl         ;exchange dest pointer and table
                                                      ;  pointer
879A'  E9                      ex    de, hl           ;put pointers in proper regs
879B'  5E 08                   ld    e, 8             ;e = scan counter
E79D'  C5                      push  bc
879E'  01 0020    up_inner     ld    bc, num_cols
8761'  EC 60                   ldir                   ;move scan up
E7A3'  01 00E0                 ld    bc, 256-num_cols
E7AA'  09                      add   hl, bc           ;hl -> start of next scan
87A7'  E8                      ex    de, hl
E7A8'  09                      add   hl, bc
E7A9'  E6                      ex    de, hl           ;de -> start of next scan dest
E7A0'  5C                      dec   e
87A8'  2C F1                   jr    nz, up_inner     ;more scans to do, so jump
E7AD'  D9                      exx
E7AE'  01 0020                 ld    bc, num_cols
8781'  EC 80                   ldir                   ;move attributes
E783'  D9                      exx
8784'  C1                      pop   bc
8785'  10 D8                   djnz  up_outer         ;if more lines to do, then jump
E787'  E1                      pop   hl
8788'  5A DFF9                 ld    a, (screen_ht)   ;clear the bottom line
E789'  5C                      dec   a
878C'  47                      ld    b, a
878D'  0E 00                   ld    c, 0
878F'  CD 890E                 call  SET_PRINT_POS
E7C2'  0A 20                   ld    b, num_cols
87C4'  5A DFF6                 ld    a, (backcolor)   ;set background color to permanent
                                                      ;  value
87C7'  52 5C6F                 ld    (ATTR_T), a
87CA'  C5         cl_bot_line  push  bc
87CB'  5E 20                   ld    a, ' '
E7CD'  CC E890                 call  WRITE_CHAR
87D0'  C1                      pop   bc
87D1'  10 F7                   djnz  cl_bot_line
87D3'  3E FF                   ld    a, -1            ;set parameter for upd_cols
E7D5'  18 79                   jr    vs_cont
87D7'  5A DFF9    down_scroll  ld    a, (screen_ht)   ;set up table pointer
E7DA'  5C                      dec   a
E7DB'  F5                      push  af
87DC'  CB 17                   rla   a
87DE'  4F                      ld    c, a
87DF'  06 00                   ld    b, 0
87E1'  11 EE60'                ld    hl, vs_table
87E4'  09                      add   hl, bc           ;hl -> entry for last line
87E9'  E5                      push  hl
87EA'  E5                      push  hl
E7E7'  D9                      exx                    ;use alt. regs. for attr pointers
```

```
8788'  81                1574                    pop     hl
8789'  5E                1575                    ld      e, (hl)
878A'  23                1576                    inc     hl
878B'  56                1577                    ld      d, (hl)
878C'  E5                1578                    ex      de, hl          ;hl -> lst scan of last line
878D'  7C                1579                    ld      a, h            ;calc. address of attr for last line
878E'  09                1580                    rrca                    ;  as done in AT78Y7 in Home dCA
878F'  0F                1581                    rrca
8790'  0F                1582                    rrca
8791'  E6 03             1583                    and     000000110
8793'  F6 58             1584                    or      010110000
8795'  67                1585                    ld      h, a
8796'  54                1586                    ld      d, h            ;de -> lst attr byte of last line
8797'  5D                1587                    ld      e, l
8798'  01 0020           1588                    ld      bc, max_cols
879B'  A7                1589                    and     a
879C'  ED 42             1590                    sbc     hl, bc          ;hl -> lst attr byte of next to last
879E'  09                1591                    add     hl, bc          ;  line
879F'  E1                1592                    pop     hl
87A0'  F1                1593                    pop     af
87A1'  23                1594                    inc     hl
87A2'  E5                1595                    push    hl
87A3'  47                1596                    ld      b, a            ;b = line counter
87A4'  E1                1597    down_outer      pop     hl
87A5'  5E                1598                    ld      e, (hl)
87A6'  23                1599                    inc     hl
87A7'  56                1600                    ld      d, (hl)         ;de -> dest for next line
87A8'  23                1601                    inc     hl
87A9'  E5                1602                    push    de              ;put dest pointer on stack
87AA'  56                1603                    ld      d, (hl)
87AB'  23                1604                    inc     hl
87AC'  5E                1605                    ld      e, (hl)         ;de -> source for next line
87AD'  23                1606                    inc     hl
87AE'  E3                1607                    ex      (sp), hl        ;exchange dest pointer and table
1608                                             ;  pointer
87AF'  EB                1609                    ex      de, hl          ;put pointers in proper regs
87B0'  38 08             1610                    ld      a, 8            ;a = scan counter
87B2'  C5                1611                    push    bc
87B3'  01 0020           1612    down_inner      ld      bc, max_cols
87B6'  ED B0             1613                    ldir                    ;move scan down
87B8'  01 0080           1614                    ld      bc, 256-max_cols
87BB'  09                1615                    add     hl, bc          ;hl -> start of next scan
87BC'  EB                1616                    ex      de, hl
87BD'  09                1617                    add     hl, bc
87BE'  EB                1618                    ex      de, hl          ;de -> start of next scan dest
87BF'  30                1619                    inc     a
87C0'  20 41             1620                    jr      nz, down_inner  ;more scans to do, so jump
87C2'  09                1621                    exx
87C3'  01 0020           1622                    ld      bc, max_cols
87C6'  ED B0             1623                    ldir                    ;move attributes
87C8'  01 0040           1624                    ld      bc, 2*max_cols
87CB'  A7                1625                    and     a
87CC'  ED 42             1626                    sbc     hl, bc
87CE'  EB                1627                    ex      de, hl
87CF'  ED 42             1628                    sbc     hl, bc
87D1'  EB                1629                    ex      de, hl
87D2'  D9                1630                    exx
87D3'  C1                1631                    pop     bc
87D4'  10 CE             1632                    djnz    down_outer      ;if more lines to do, then jump
87D6'  E1                1633                    pop     hl
87D7'  01 0000           1634                    ld      bc, 0           ;clear top line
87DA'  CD 8908           1635                    call    SBT_SPIN7_805
87DD'  06 20             1636                    ld      b, max_cols
87DF'  3E 04F8           1637                    ld      a, (backgcolor) ;set background color to permanent
1638                                             ;  value
87E1'  32 5C8F           1639                    ld      (d776_7), a
87E4'  C5                1640    cl_top_line     push    bc
87E5'  3E 20             1641                    ld      a, ' '
87E7'  CD 8895           1642                    call    W61TE_CHAR
87EA'  C1                1643                    pop     bc
87EB'  10 F7             1644                    djnz    cl_top_line
87ED'  3E 01             1645                    ld      a, 1            ;set parameter for mod_cols
87EF'  CD 8686'          1646    vs_cont         call    mod_rows
87F2'  F8                1647                    ei
87F3'  D5                1648                    exx
87F4'  E1                1649                    pop     hl
87F5'  D1                1650                    pop     de
87F6'  C1                1651                    pop     bc
87F7'  D9                1652                    exx
87F8'  E1                1653                    pop     hl
87F9'  D1                1654                    pop     de
87FA'  F1                1655                    pop     af
87FB'  01 8000           1656                    ld      bc, 8
87FE'  C9                1657                    ret
                         1658
                         1659
                         1660    ;This table contains the starting address of each row for use in vertical
                         1661    ;  scrolling
                         1662
8800'  4000              1663    vs_table        defw    4000H
8802'  4020              1664                    defw    4020H
8804'  4040              1665                    defw    4040H
8806'  4060              1666                    defw    4060H
8808'  4080              1667                    defw    4080H
880A'  40A0              1668                    defw    40A0H
880C'  40C0              1669                    defw    40C0H
880E'  40E0              1670                    defw    40E0H
8810'  4800              1671                    defw    4800H
8812'  4820              1672                    defw    4820H
8814'  4840              1673                    defw    4840H
8816'  4860              1674                    defw    4860H
8818'  4880              1675                    defw    4880H
881A'  48A0              1676                    defw    48A0H
881C'  48C0              1677                    defw    48C0H
881E'  48E0              1678                    defw    48E0H
8820'  5000              1679                    defw    5000H
8822'  5020              1680                    defw    5020H
8824'  5040              1681                    defw    5040H
8826'  5060              1682                    defw    5060H
8828'  5080              1683                    defw    5080H
882A'  50A0              1684                    defw    50A0H
882C'  50C0              1685                    defw    50C0H
882E'  50E0              1686                    defw    50E0H
                         1687                    end
                         1688
                         1689
```

```
SPRSVC  CAZE0/11 version 10.36.14        15-May-E4  9:51:11
Sprite services module    S00SVC.SAC

A        Reserved     ALLOC    0000        ATTA0P   SCED        ATTR0T   ECEF        AUTOWR   0001
B        Reserved     BACKGC   30F9        0C0DK    E4F0        00C0DK   E50T        BLAC00   E52E
BDROCR   SC4E         SCRD0T   00FE        C        Reserved    CHARS    SC36        CHEC09   E2E2
CNT6L    E090         CH0ATT   E465        CN0A9C   E4T9        CN0A01   E46E        C00LCC   E450
CN0L01   E459         CKADW1   E35F        CK00W2   E3FS        CLEAA0   E250        CLS06    097F
CL06DT   ETCA         CL0TD0   EE45        CCL      0001        CCLDA    0004        COLC00   E2EF
COL0LO   E354         COMMAN   0F0T        CONT1    E360        CONT2    E405        CAEATE   E27E
C0LOOF   E3EA         D        Reserved    CC0E01   E2FE        DISPLA   E20A        DOW09I   E013
DOWN0D   E004         DOWN0S   ETDT        E        Reserved    ENOUGH   E253        ERASE0   E361
ERA0CH   E404         E0EXIT   E421        E0TD0D   E3C0        GET0CH   E010        GFLAGS   DFFE
GLOBAL   0010         H        Reserved    HEIGHT   0003        HS0CON   E6T4        H0SC00   E5D6
1D0DK    E2E4         INITSC   E4E9        INIT0S   E219        INNEA1   E346        INNE02   E3DE
L        Reserved     LEFT0I   E609        LEFT0D   E5F0        LEFT0S   E5E0        LEFT0T   E600
LNS      0002         M        Reserved    MAX0CD   0020        MAS0AD   0016        MA00SF   DFFA
MOVE0S   E426         M0E0IT   E440        NAME     30FE        NEG      0000        NE0TCH   E30A
NE0T0S   E5C9         NO0CLE   E549        NO0CCL   E4EE        NO0CVE   E30T        NO0STD   E37A
NUM0FA   D0EC         NXTCHA   E40C        NX0SPR   ET64        DF0      E204        OUTE01   E343
OUTER2   E309         OVEAL0   E464        CVEAWA   0001        OVRW0D   E3A0        D0E0I7   E4E3
D01D0N   E40F         0ODK1    E493        FA0HS    0FE0        0A0H1    0FED        0AA02    00EF
0AAN3    0F01         FA0H4    D003        0AAHS    30FS        001THA   0005        FSW      Reserved
0UT0SF   E30T         00EXIT   E399        F0ID0G   E310        0AHTOF   SC02        AIG0T0   E430
ADW      0000         00W0LD   E365        00W00K   E442        0T0INN   E64C        AT0OUT   E640
RT0TAB   E456         00LOOP   E3FS        SA0EKI   E206        SCAATC   00E3        SCREEN   00F9
SELECT   1230         SENDTV   0500        SET0AT   0502        SET0AU   E2C9        SET0FR   E90E
SFLAGS   000T         SH0C0    E4F1        SF       Reserved    SF0C00   E000        SFAITE   DFFE
SFR0AD   E25F         SVC0CO   00EE        TVFLAG   SC3C        UC0CON   E590        UC1      E599
UFDC0L   E5E0         UFOA0L   ET12        U00SCO   E954        UFD0AD   E6E6        UF01WN   E79E
UF0OUT   ETEF         UF0SCA   ETTE        UA0CON   E73A        URI      ET20        VISIEL   E2FE
VS0CON   E050         VS0T00   E940        V0SC0C   ETT1        WIDTH    0002        WRITE0   E090
WRTCHA   E3TA         W0T0SF   E50E        WT0S0A   ETS9        T        SC3A        YES0CO   E400

    No errors detected


                                NAME  IOLOW
                                SUETTL Low level I/O module



          ;************************************************************
          ;
          ; Low Level I/O Module
          ;
          ; Inputs: parameters to desired service in appropriate registers.
          ;
          ; Outputs: none.
          ;
          ; Description: this module provides the following low level i/o services:
          ;
          ;       Set_Print_Pos - sets the print position
          ;       Write_Char - writes a character to the current print position
          ;       Get_Char - returns the character code of the character at the
          ;                   desired position
          ;
          ;             These services are based on the routines SETCUR, WPCHAR,
          ;             and GTCHAR of the dual-column support component of the
          ;             Applications Development Library.
          ;
          ;************************************************************


          ; Imports (Note: all symbols in upper case are imported)

          ;From the Name ABM

          ;Variables

      =SCTE          UDG              EQU      SCTEH
      =SCE0          ATTA_P           EQU      SCE0H
      =SCEE          MASK_P           EQU      SCEEH
      =EC91          P_FLAG           EQU      SC91H
      =SCC2          VIDMOD           EQU      SIC2H


          ; Exports

          ;Variables

          ;                  chtA1

          ;Services

          ;                  write_Char, Set_Print_Pos, Get_Char


          ; Locals

          ;Constants

      =001E          nrows            equ      24        ; 24 lines
      =3C00          chrset           equ      3e00h     ; rom char.table-100h
      =ES3A          grphst           equ      ((grest1-100h) ;std.graphics chars.


          ;Variables
      =E89E                   ORG      0EE90H

E650  3C00           chtbl            defw     chrset    ; address of character table  ST391
                                                         ; (defaults to ts2068 system
                                                         ;  rom)
EE92  ES3A           grtbl            defw     grphst    ; address of std.graphics tbl.ST393
0E94  20             linlen           defb     20h       ; line length (32)            ST395
EE95  1E21           curpos           defw     1E21h     ; current position (internal  ST396
                                                         ;  format)
EE9T  0E00           dfadrs           defb     4800h     ; current display file addrs. ST395
EE95  00             maskA6           defb     0         ; mask to As position to      ST400
                                                         ; display characters
EE9A  00             attbyt           defb     0h        ; current attributes          ST401
0E9E  E0             gtindx           defA     E0h       ; index for adjusting         ST402
                                                         ; ch.ords in gtchar
EE8C  00             attmsk           defb     E         ; attribute mask              ST407
```

```
                ;**********************************************************************
                ;
                ; WRITE_CHAR
                ;
                ; Inputs: A = character code.
                ;
                ; Outputs: AC = 0000 - OK
                ;             = 0CXX - error, XA <> 00
                ;
                ; Globals Read: UDG
                ;               grtbl
                ;               chtbl
                ;               linlen
                ;               maskd
                ;
                ; Globals Written: none.
                ;
                ; Procedures Called: ldattr
                ;                    ldscan
                ;                    updatt
                ;                    stpsan
                ;
                ; Procedures Called By: DISPLAY_CHAR
                ;                       V_SCROLL
                ;
                ; Description: this service writes the specified character to the current
                ;              print position.
                ;
                ;**********************************************************************



889D  F5                write_char     push   af
889E  D5                               push   de
889F  E5                               push   hl
88A0  CD E624                          call   ldattr        ; entry with code in a
                                                             ; get attribute and mask controls

                                                             ; entry to use attribute values in
                                                             ; internal variables without reloading
88A3  CD 890F           wrch01         call   ldscan        ; load registers for current position
                                                             ; bc=cursor position from 'cursor'
                                                             ; hl=display file address from 'dfcore'
                                                             ; a=code for char. to be displayed

88A6  C5                               push   bc            ; save cursor position
88A7  FE 80                            cp     80h           ; test if ascii printable (20h-7fh)
88A9  38 15                            jr     c, wrch12     ; yes
88AB  FE 90                            cp     90h           ; test if std. graphics (80h-8fh)
88AD  38 05                            jr     c, wrch11
88AF  ED 4B 8C78                       ld     bc, (UDG)     ; user-defined graphics(90h-a4h)
88B3  05                               dec    b             ; adjust address-100h
88B4  D6 70                            sub    70h           ; adjust for index into table
88B6  18 0C                            jr     wrch13
88B8  ED 4B 8892        wrch11         ld     bc, (grtbl)   ; address of graphics table
88BC  D6 60                            sub    60h           ; adjust for index into table
88BE  18 04                            jr     wrch13
88C0  ED 4B 8890        wrch12         ld     bc, (chtbl)   ; character table
88C4  EB                wrch13         ex     de, hl        ; de=scan.in display file
88C5  26 00                            ld     h, 0
88C7  6F                               ld     l, a          ; code is index into table
88C8  29                               add    hl, hl
88C9  29                               add    hl, hl
88CA  29                               add    hl, hl
88CB  09                               add    hl, bc        ; hl=pixel pattern in table
88CC  C1                               pop    bc
88CD  EB                               ex     de, hl
88CE  79                               ld     a, c          ; test if end of line
88CF  3C                               inc    a
88D0  3A 8894                          ld     a, (linlen)
88D3  20 05                            jr     nz, wrch14
88D5  3C                               inc    a             ; start of new line
88D6  05                               dec    b             ; bump line no.
88D7  4F                               ld     c, a          ; linlen+1=start of line
88D8  18 01                            jr     wrch15
88DA  3C                wrch14         inc    a
88DB                    wrch15                              ; here to put character in display file
88DB  C5                wrch2          push   bc            ; cursor position
88DC  E5                               push   hl            ; display file adrs.
88DD  3A E899                          ld     a, (maskb)
88E0  06 FF                            ld     b, -1
88E2  1F                               rra
88E3  38 01                            jr     c, wrch3      ; if xoring new into old
88E5  04                               inc    b             ; b=-1 if xoring =0 if not
88E6  1F                wrch3          rra
88E7  1F                               rra
88E8  9F                               sbc    a, a
88E9  4F                               ld     c, a          ; c=-1 for inverse =0 if not
88EA  3E 08                            ld     a, 8          ; scan count
88EC  47                               and    a
88ED  EB                               ex     de, hl
88EE  08                wrch1          ex     af, af'
88EF  1A                               ld     a, (de)       ; old bit row
88F0  A0                               and    b
88F1  AE                               xor    (hl)
88F2  A9                               xor    c             ; invert if c=-1
88F3  12                               ld     (de), a       ; update display file
88F4  08                               ex     af, af'
88F5  14                               inc    d             ; next scan
88F6  23                               inc    hl            ; next byte in char.file
88F7  3D                               dec    a             ; test if more scans
88F8  20 F4                            jr     nz, wrch1
88FA  15                wrch7          dec    d                 ; adjust df adrs.
88FB  EB                               ex     de, hl        ; df adrs. to hl
88FC  CD 89E7                          call   updatt        ; update attribute byte
```

```
EEPP   E1                          pop     hl            ! starting address
E900   C1                          poo     bd            ! cursor position
5901   00                          oec     c             ! adjust cursor position
E903   23                          inc     hl            ! adjust df address
E903   CC 5907     wrchat          call    stoeen        ! store new position
5906   CE 00       goodret         ld      e, 0          ! return be=0
5908   06 00       errret          ld      b, 0          ! enter here with e non-zero
E9EA   E1                          pop     hl
E90E   D1                          pop     de
E90C   P1                          pop     af
E9ED   C9                          ret


           ;****************************************************************
           !
           1 SET_PRINT_POS
           !
           1 Inputs! S = row
           !         C = coluen
           !
           ! Outputs! SC = 0300 - OK
           !
           ! Gloeels Read: none.
           !
           ! Globals written: none.
           !
           ! Procedures Called: convfm
           !                    uedseen
           !
           ! Procedures Called Sy! DISPLAY_CHAR
           !                       V_SCROLL
           !
           ! Description! this service sets the print position to the specified roe
           !              ane coluen.  The row position must be in the range 0..23
           !              ane the column position must be in the range 0..31.
           !
           ;****************************************************************


E90E   P5          set_print_pos   push    af
E90P   D5                          push    de
E910   E5                          push    hl
E511   CD E99D                     call    convfe        ! entry with line/col. in bc reg.
                                                         ! convert to internal format
E914   CC 5915                     call    updseen       ! calc. and save the print posn.
E917   0E 00                       ld      c, 0          ! return bc=0
5919   06 C0                       ld      b, 0          ! enter here with c non-zero
E915   E1                          pop     hl
E91C   D1                          pop     de
E91D   P1                          pop     af
E91E   C9                          ret


           ;****************************************************************
           !
           ! GET_CHAS
           !
           ! Inputs! S = row
           !         C = coluen
           !
           ! Outputs! A = character code, if a chars is found, 0 otherwise
           !          SC = 00ce - OK, match found, XX = A
           !             = 0000 - OK, no match found
           !
           ! Globals Read: chtbl
           !               gtindx
           !               prtal
           !               UOG
           !
           ! Globals Written: gtindx
           !
           ! Procedures Called! convfe
           !                    calcpos
           !
           ! Procedures Called Sy! PUT_SPRITE
           !
           ! Description! this service returns the character code of the character
           !              at the specified screen location.  The row and column must
           !              be in the range specified above for Set_Print_Pos.
           !
           ;****************************************************************


E91P   D5          get_char        push    de
E930   E5                          push    hl
E931   CD E95D                     call    convfe        ! entry with position in bc
                                                         ! convert to internal format
E924   CD E94P                     call    calcpos       ! get display file adrs.
E927   5D 55 E590                  ld      de, (chtbl)   ! char.table
E92E   E6 60                       ld      b, 96         ! no. of printable characters
E92D   3E E0                       ld      a, E0h        ! set adjustment index
E92P   32 E898     gtch1           ld      (gtindx), a
E932   1s                          inc     d             ! adjust to start of table
E933   EE                          ex      de, hl        ! df adrs.in hl, char.set in hl
E934   C5          gtch2           push    bc            ! save count
E935   D5                          push    de            ! save adrs. in df
E936   E5                          push    hl            ! save adrs. in char.table
E937   1A                          ld      a, (de)       ! scan row from df
E93E   65                          xor     (hl)          ! test against char. set
E939   25 10                       jr      n, gtch3      ! match
E93E   P5                          push    af
E93C   34 E59E                     ld      a, (gtindx)
E93P   FE 90                       cp      90h           ! test if std.graphics
E941   2E D3                       jr      nz, gtch23
E943   P1                          pop     af
E944   16 35                       jr      gtch6         ! do not test for inverse
E946   P1          gtch23          pop     af
5947   3C                          inc     a             ! test inverse
```

```
E948   20 24                       jr     nc, gtch4
E94A   3D                          dec    a             ; a=-1 for inverse
E94B   4F               gtch3      ld     c, a          ; c=0 for match  -1 for inverse
E94C   06 07                       ld     b, 7
E94E   1C               gtch31     inc    e             ; next scan in df
E94F   23                          inc    hl            ; next byte in char.set
E950   1A                          ld     a, (de)
E951   AE                          xor    (hl)
E952   A9                          xor    c
E953   20 19                       jr     nz, gtch4     ; no match
E955   10 F7                       djnz   gtch31        ; for next scan

                                   ; here if match with (sp)=char in char set
                                   ; (sp+2)=char.in display file (sp+4)=char.
                                   ; count

E957   79                          ld     a, c          ; a=-2 if match on inverse
E958   C1                          pop    bc
E959   C1                          pop    bc
E95A   C1                          pop    bc
E95B   4F                          ld     c, a          ; b=count c=-1 if match on inverse
E95C   3A E99D                     ld     a, (gtinev)   ; adjust char.code
E95F   90                          sub    b             ; a=char.code
E960   FE 20                       cp     20h           ; test if space
E962   20 02                       jr     nz, gtch22
E964   0C                          inc    c             ; test if match on inverse space
E965   20 02                       jr     nz, gtch22
E967   3E 8F                       ld     a, 8fh        ; set code for graphics block
E969   4F               gtch32     ld     c, a
E96A   06 00                       ld     b, 0
E96C   18 2C                       jr     gc_exit       ; char.code in a and in bc

E96E   E1               gtch4      pop    hl            ; here when no match
E96F   11 0008                     ld     de, 8         ; look at next char. in set
E972   19                          add    hl, de        ; char. set
E973   D1                          pop    de            ; char. in df
E974   C1                          pop    bc            ; char.count
E975   10 E0                       djnz   gtch2         ; look again

                                   ; here to test if done

E977   EB                          ex     de, hl        ; df adrs.to hl
E978   3A E99D                     ld     a, (gtinev)
E97B   FE 80                       cp     80h           ; test if std.chr.set
E97D   20 CA                       jr     nz, gtch2     ; try graphics
E97F   ED 5B E892                  ld     de, (grtbl)   ; std.graphics table
E983   06 10                       ld     b, 16         ; no. of entries
E985   3E 90                       ld     a, 90h        ; index
E987   18 A6                       jr     gtch1
E989   FE 90            gtch5      cp     90h           ; test if std.graphics
E98B   20 0E                       jr     nz, gtch6     ; done if not
E98D   ED 5B 5C7B                  ld     de, (UDG)     ; try user-defined graphics area
E991   15                          dec    d             ; adjust address-100h
E992   06 15                       ld     b, 21         ; no of entries
E994   3E A5                       ld     a, 0a5h       ; index
E996   18 97                       jr     gtch1
E998   48               gtch6      ld     c, b          ; here when no match anywhere
E999   AF                          xor    a             ; set bc and a=zero
E99A   E1               gc_exit    pop    hl
E99B   D1                          pop    de
E99C   C9                          ret


                        ;*****************************************************************
                        ;
                        ; CONVRC
                        ;
                        ; Inputs: B = row
                        ;         C = column
                        ;
                        ; Outputs: B = row in internal format
                        ;          C = column in internal format
                        ;
                        ; Globals Read: linlen
                        ;
                        ; Globals Written: none.
                        ;
                        ; Procedures Called: none.
                        ;
                        ; Procedures Called By: set_print_pos
                        ;                       get_char
                        ;
                        ; Description: this routine converts the row and column as used by
                        ;              Set_Print_Pos and Get_Char into an intenal format used
                        ;              by the other routines in this module.
                        ;
                        ;*****************************************************************


E99D   3A 8E94          convrc     ld     a, (linlen)
E9A0   3C                          inc    a
E9A1   91                          sub    c
E9A2   4F                          ld     c, a
E9A3   3E 18                       ld     a, scrsz
E9A5   90                          sub    b
E9A6   47                          ld     b, a
E9A7   C9                          ret
```

```
                        331   ;*****************************************************************
                        332   ;
                        333   ; UPDPOSN
                        334   ;
                        335   ; Inputs: B = row in internal format
                        336   ;         C = column in internal format
                        337   ;
                        338   ; Outputs: none
                        339   ;
                        340   ; Globals Read: none.
                        341   ;
                        342   ; Globals Written: none
                        343   ;
                        344   ; Procedures Called: calcpos
                        345   ;                    sipcon
                        346   ;
                        347   ; Procedures Called By: put_print_pos
                        348   ;
                        349   ; Description: this routine put the print position to the specified value.
                        350   ;
                        351   ;*****************************************************************
                        352
                        353
                        354   updposn                           ; enter here with bc=line/col.in
                        355                                     ; internal format
                        356
8948   CD 894F          357                   call    calcpos         ; calculate position
8948   CD 8987          358                   call    sipcon          ; store updated position and return
8948   C9               359                   ret
                        360
                        361
                        362   ;*****************************************************************
                        363   ;
                        364   ; CALCPOS
                        365   ;
                        366   ; Inputs: B = row in internal format
                        367   ;         C = column in internal format
                        368   ;
                        369   ; Outputs: HL = address in display file
                        370   ;
                        371   ; Globals Read: linlen
                        372   ;
                        373   ; Globals Written: none.
                        374   ;
                        375   ; Procedures Called: lnbu
                        376   ;
                        377   ; Procedures Called By: put_char
                        378   ;                       updposn
                        379   ;
                        380   ; Description: this routine calculates the position in the display file
                        381   ;              corresponding to the specified row and column position.
                        382   ;
                        383   ;*****************************************************************
                        384
                        385
8949   CD 898C          386   calcpos           call    lnbu            ; get display file addr.
894C   3A 8894          387                     ld      a, (linlen)     ; add column offset
894F   3C               388                     inc     a
8950   91               389                     sub     c
8951   9F               390                     ld      e, a
8952   16 00            391                     ld      d, 0
8954   19               392                     add     hl, de
8955   C9               393                     ret
                        394
                        395
                        396   ;*****************************************************************
                        397   ;
                        398   ; LNBU
                        399   ;
                        400   ; Inputs: B = row in internal format
                        401   ;
                        402   ; Outputs: HL = address in display file of start of line B.
                        403   ;
                        404   ; Globals Read: VIDM00
                        405   ;
                        406   ; Globals Written: none.
                        407   ;
                        408   ; Procedures Called: none.
                        409   ;
                        410   ; Procedures Called By: calcpos
                        411   ;
                        412   ; Description: this routine calculates the address of the start of row B
                        413   ;              in the display file.  This is the first byte of the first
                        414   ;              scan line.
                        415   ;
                        416   ;*****************************************************************
                        417
                        418
898C                    419   lnbu                              ; get display file address
                        420                                     ; for start of line in b
898C   3E 18            421                     ld      a, 24
898E   90               422                     sub     b
898F   57               423                     ld      d, a
89C0   8F               424                     rrca
89C1   8F               425                     rrca
89C2   8F               426                     rrca
89C3   E6 E0            427                     and     0e0h
89C5   6F               428                     ld      l, a
89C6   7A               429                     ld      a, d
89C7   E6 18            430                     and     18h
89C9   F6 40            431                     or      40h
89CB   67               432                     ld      h, a
89CC   3A 9CC2          433                     ld      a, (VIDM00)
89CF   CB 47            434                     bit     0, a            ; test if df1
8901   C8               435                     ret     z
8902   3E 20            436                     ld      a, 20h          ; set to df2
8904   B4               437                     or      h
8905   67               438                     ld      h, a
8906   C9               439                     ret
                        440
                        441
```

```
                        ;*******************************************************************
                        ;
                        ; STPOSN
                        ;
                        ; Inputs: BC = current print position
                        ;         HL = display file address
                        ;
                        ; Outputs: none
                        ;
                        ; Globals Read: none.
                        ;
                        ; Globals Written: curpos
                        ;                  dfadrs
                        ;
                        ; Procedures Called: none.
                        ;
                        ; Procedures Called By: write_char
                        ;                       updscan
                        ;
                        ; Description: this routine saves the inputs in the appropriate variables.
                        ;
                        ;*******************************************************************


E9D7                    stposn                          ; store cursor position
E9D7    ED 43 EE95                      ld      (curpos), bc
E9DB    22 EE97                         ld      (dfadrs), hl
E9DE    C9                              ret


                        ;*******************************************************************
                        ;
                        ; LDPOSN
                        ;
                        ; Inputs: none.
                        ;
                        ; Outputs: BC = current print position
                        ;          HL = current display file address
                        ;
                        ; Globals Read: curpos
                        ;               dfadrs
                        ;
                        ; Globals Written: none.
                        ;
                        ; Procedures Called: none.
                        ;
                        ; Procedures Called By: write_char
                        ;
                        ; Description: this routine retrieves the current print position and
                        ;              display file address from the appropriate variables.
                        ;
                        ;*******************************************************************


E9DF                    ldposn                          ; load cursor position
E9DF    ED 4B EE95                      ld      bc, (curpos)
E9E3    2A EE97                         ld      hl, (dfadrs)
E9E6    C9                              ret


                        ;*******************************************************************
                        ;
                        ; UPDATT
                        ;
                        ; Inputs: HL = address of any scan line of a character position
                        ;
                        ; Outputs: none.
                        ;
                        ; Globals Read: attbyt
                        ;               attmsk
                        ;               maskb
                        ;
                        ; Globals Written: none.
                        ;
                        ; Procedures Called: calcatt
                        ;
                        ; Procedures Called By: write_char
                        ;
                        ; Description: this routine updates the attribute byte for the character
                        ;              position identified by the address in HL.  The attribute
                        ;              value written is taken from the attbyt variable.
                        ;
                        ;*******************************************************************


E9E7                    updatt                          ; update attribute byte for character
                                                        ; just printed
                                                        ; hl = any scan of char.in df

E9E7    CD EA14                         call    calcatt         ; adrs. of attribute byte to hl
E9EA    3A EE96                         ld      a, (attbyt)
E9ED    5F                              ld      e, a            ; current attr.value to e
E9EE    3A EE9C                         ld      a, (attmsk)
E9F1    57                              ld      d, a            ; mask to d
E9F2    3A EE99                         ld      a, (maskb)
E9F5    47                              ld      b, a            ; flags to b
E9F6    7E                              ld      a, (hl)         ; byte from attribute file
E9F7    AB                              xor     e
E9F8    A2                              and     d
E9F9    AB                              xor     e               ; new attr.for 0; old for 1
E9FA    CB 60                           bit     4, b            ; set ink complement of paper?
E9FC    28 08                           jr      z, updat1
E9FE    E6 F8                           and     0F8h            ; set ink to black
EA00    CB 6F                           bit     5, a            ; is paper 4-7
EA02    20 02                           jr      nz, updat1      ; yes - use black ink
EA04    F6 07                           or      7               ; no -  use white ink
EA06    CB 70                   updat1  bit     6, b            ; set paper complement of ink?
EA08    28 08                           jr      z, updat2       ;
EA0A    E6 C7                           and     0C7h            ; set paper to black
EA0C    CB 57                           bit     2, a            ; is ink 4-7
EA0E    20 02                           jr      nz, updat2      ; yes - use black paper
EA10    F6 38                           or      38h             ; no -  use white paper
EA12    77                      updat2  ld      (hl), a         ; to attribute file
EA13    C9                              ret
```

```
                    ;*****************************************************************
                    ;
                    ; CALCATT
                    ;
                    ; Inputs: HL = address of any scan of a character position
                    ;
                    ; Outputs: HL = address of attribute byte
                    ;
                    ; Globals Read: none.
                    ;
                    ; Globals Written: none.
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: setatt
                    ;
                    ; Description: this routine calculates the address of the attribute byte
                    ;              corresponding to the character position identified with
                    ;              HL.
                    ;
                    ;*****************************************************************


EA14   7C           calcatt     ld      a, h            ; upper byte of address
EA15   0F                       rrca
EA16   0F                       rrca
EA17   0F                       rrca
EA18   E6 03                    and     03h
EA1A   F6 58                    or      58h
EA1C   CB 6C                    bit     5, h            ; test which df
EA1E   28 02                    jr      z, calca1
EA20   F6 20                    or      30h             ; df3
EA22   67           calca1      ld      h, a            ; addr. of attribute in df3
EA23   C9                       ret


                    ;*****************************************************************
                    ;
                    ; LDATTR
                    ;
                    ; Inputs: none.
                    ;
                    ; Outputs: none.
                    ;
                    ; Globals Read: ATTR_P
                    ;               MASK_P
                    ;               P_FLAG
                    ;
                    ; Globals Written: attbyt
                    ;                  attmsk
                    ;                  maskb
                    ;
                    ; Procedures Called: none.
                    ;
                    ; Procedures Called By: write_char
                    ;
                    ; Description: this routine sets the values of the variables used by this
                    ;              module for manipulating attributes with values from the
                    ;              variables used by the Home ROM.
                    ;
                    ;*****************************************************************


EA24                ldattr                              ; load internal attribute variables
                                                        ; from system variables

EA24   F5                       push    af              ; save a
EA25   3A 8D5C                  ld      a, (ATTR_P)
EA28   32 E89A                  ld      (attbyt), a
EA2B   3A 8E5C                  ld      a, (MASK_P)
EA2E   32 E89C                  ld      (attmsk), a
EA31   3A 915C                  ld      a, (P_FLAG)
EA34   0F                       rrca                    ; shift odd bits to even
EA35   32 E89E                  ld      (maskb), a
EA38   F1                       pop     af
EA39   C9                       ret


                    ; Standard graphics characters

EA3A   00           grset   defb 00000000b              ; code 80      graphics space
EA3B   00                   defb 00000000b
EA3C   00                   defb 00000000b
EA3D   00                   defb 00000000b
EA3E   00                   defb 00000000b
EA3F   00                   defb 00000000b
EA40   00                   defb 00000000b
EA41   00                   defb 00000000b

EA42   0F                   defb 00001111b              ; code 81      graphics
EA43   0F                   defb 00001111b
EA44   0F                   defb 00001111b
EA45   0F                   defb 00001111b
EA46   00                   defb 00000000b
EA47   00                   defb 00000000b
EA48   00                   defb 00000000b
EA49   00                   defb 00000000b

EA4A   F0                   defb 11110000b              ; code 82      graphics
EA4B   F0                   defb 11110000b
EA4C   F0                   defb 11110000b
EA4D   F0                   defb 11110000b
EA4E   00                   defb 00000000b
EA4F   00                   defb 00000000b
EA50   00                   defb 00000000b
EA51   00                   defb 00000000b

EA52   FF                   defb 11111111b              ; code 83      graphics
EA53   FF                   defb 11111111b
EA54   FF                   defb 11111111b
EA55   FF                   defb 11111111b
```

```
0b56   00
0b57   00
0b58   00
0A39   00

0b3b   00
0A98   00
0A8C   00
0A3D   00
0A8E   0F
0A3F   0F
0b60   0F
0A61   0F

0A62   0F
0A63   0F
0b64   0F
0A63   0F
0b66   0F
0b67   0F
0b6E   0F
0A69   0F

0b6A   F0
0A68   F0
0A6C   F0
0b6D   F0
0b6E   0F
0b6F   0F
0A70   0F
0A71   0F

0b72   FF
0b73   FF
0b74   FF
0b73   0F
0A7e   0F
0b77   0F
0A78   0F
0A79   0F

0A7b   00
0b7B   00
0b7C   00
0A7D   00
0b7E   F0
0b7F   F0
0A80   F0
0A81   F0

0A82   0F
0A83   0F
 0A84   0F
 0A83   0F
 0b86   F0
 0b87   F0
 0A88   F0
 0A83   F0

 0b8b   F0
 0A8B   F0
 0A8C   F0
 0A8D   F0
 0b8E   F0
 0b8F   F0
 0A90   F0
 0A91   F0

 0A92   FF
 0A93   FF
 0b94   FF
 0b95   FF
 0A86   F0
 0A97   F0
 0b98   F0
 0A99   F0

 0A9A   00
 0b9B   00
 0A8C   00
 0b9D   00
 0A8E   FF
 0A8F   FF
 0A40   FF
 0bb1   FF

 0bb2   0F
 0bb3   0F
 0bb4   0F
 0AB8   0F
 0AB6   FF
 0bA7   FF
 0bb8   FF
 0bb8   FF

 0ABA   F0
 0AB8   F0
 0ABC   F0
 0AB0   F0
 0AB8   FF
 0ABF   FF
 0b50   FF
 0b81   FF

 0b82   FF
 0AB3   00
 0AB4   FF
 0b85   FF
 0AB6   FF
 0AB7   FF
 0AB8   FF
 0b89   FF
```

```
729          defb 00000000b
730          defb 00000000e
731          defb 00002000b
732          defb 00C00000b
933
734          defb 00000000b          1 code 84       graphics
733          defb 00000000e
736          defb 00000000b
737          defb 000CE000b
738          defb 000C1111b
739          defb 00001111b
940          defb 00001111b
761          defb 00001111b
742
743          defb 00001111b          1 code 88       graphics
746          defb 000C1111b
748          defb 00001111b
746          defb 00001111b
747          defb 00001111b
748          defb 00001111e
749          defb 000C1111b
730          defb 00001111b
731
732          defb 11110000b          1 code 84       graphics
733          defb 1111C000e
734          defb 11113000b
733          defb 1111C000b
736          defb 000011110
737          defb 00001111b
738          defb 00001111e
789          defb 00001111e
760
761          defb 11111111b          1code 87        graphics
762          defb 11111111b
783          defb 11111111b
964          defb 11111111b
783          defb 000C1111b
766          defb 00001111b
787          defb 00031111e
788          defb 00001111b
789
770          defe 00000300b          1 code 8E       graphics
771          defb 00000000e
772          defb 00003000e
773          defe 000C8000e
774          defb 11113000e
775          defe 1111C000e
776          defb 1111@000b
777          defb 11110000e
778
779          defb 0000111le          1 code 89       graphics
780          defb 00001111b
781          defe 000C1111e
782          defb 000C1111b
783          defb 11113000b
784          defb 1111C000b
788          defb 11113000b
786          defb 11113000b
787
788          defb 1111C000b          1 code 8b       graphics
788          defb 1111C000b
780          defb 1111C000b
791          defb 1111C000b
792          defb 1111C000e
793          defb 1111C000b
794          defb 1111C000b
798          defb 1111C000b
786
787          defb 1110l111b          1 code 86       graphics
738          defb 1110l111b
789          defb 1110l111b
800          defb 1110l111b
801          defb 1110C000b
802          defb 1110C000b
803          defb 1110C000b
806          defb 1110C000b
803
806          defb 000C0000b          1 code 8C       graphics
807          defb 000C0C000b
808          defb 000C0C000e
809          defb 000C0C000b
810          defb 11111111b
811          defe 11111111b
812          defb 11111111b
813          defb 11111111b
814
815          defb 00011111b          1 code 80       graphics
816          defb 00011111b
817          defb 00011111b
818          defb 00011111b
819          defb 11111111b
820          defb 11111111b
821          defb 11111111b
822          defb 11111111b
823
824          defb 11113000b          1code 8E        graphics
823          defb 1111C000b
826          defe 11116000b
827          defb 1111C000b
828          defb 11111111b
823          defb 11111111b
830          defe 11111111e
831          befe 11111111e
832
833          defb 11111111b          1code 8F        graphics
834          defb 11111111e
838          defb 11111111b
838          defb 11111111e
837          befb 11111111b
838          befb 11111111e
839          defb 11111111b
840          defb 11111111b
841
842
843          end
```

```
IOLOW   CRZ80/11 version 10.3A.14        7-May-84  15:19:10
Low level I/O module   IOLOW.SRC

A       Reserved    ATTBYT  E89A      ATTMSK  E89C      ATTRRP  5C8D      S       Reserved
C       Reserved    CALCAT  EA16      CALCA1  E822      CBLCPO  E9AF      CHRSET  3C00
CNTBL   E898        CONVPN  E99D      CURPCS  E893      D       Reserved  DPADRS  EE9T
E       Reserved    ERRRET  EA0E      GCHEBI  E99A      GETRCH  E71F      GODDRE  E906
GRPHST  E93A        GBPST   EA3A      GRTBC   EA92      GTCH1   EA2F      GTCH2   EA34
GTCH23  E94A        GTCH3   E94E      GTCH31  E94E      GTCH32  E969      GTCH4   EA8E
GTCH5   E9E9        GTCH6   EA9E      GTINDB  E898      H       Reserved  L       Reserved
LDBTTR  EA26        LOPDSN  E9DF      LINLEN  E896      LNEU    E98C      M       Reserved
MBSKB   E899        MASKRP  5CEE      PSW     Reserved  PRFLAG  5C91      SCRS2   031E
SETRPS  E90E        SP      Reserved  STPGSW  EA07      UDG     5C7B      UPDATT  EAE7
UPDAT1  EA06        UPDAT2  E812      UPDPC3  E9AE      VIDMOD  5CC2      WACHKT  EA03
WRCHD1  E8A3        WRCH11  EEEE      WRCH12  EEC0      WBCH13  E8C4      WRCH14  EED4
WBCH15  EAD8        WRCH2   E808      WRCH3   EEF6      WRCH5   EEE8      WRCHT   EAFA
WRITEB  EE90

     No errors detected
```

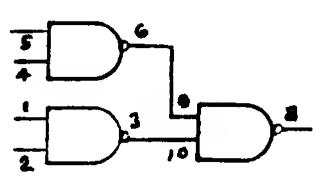
APPENDIX D

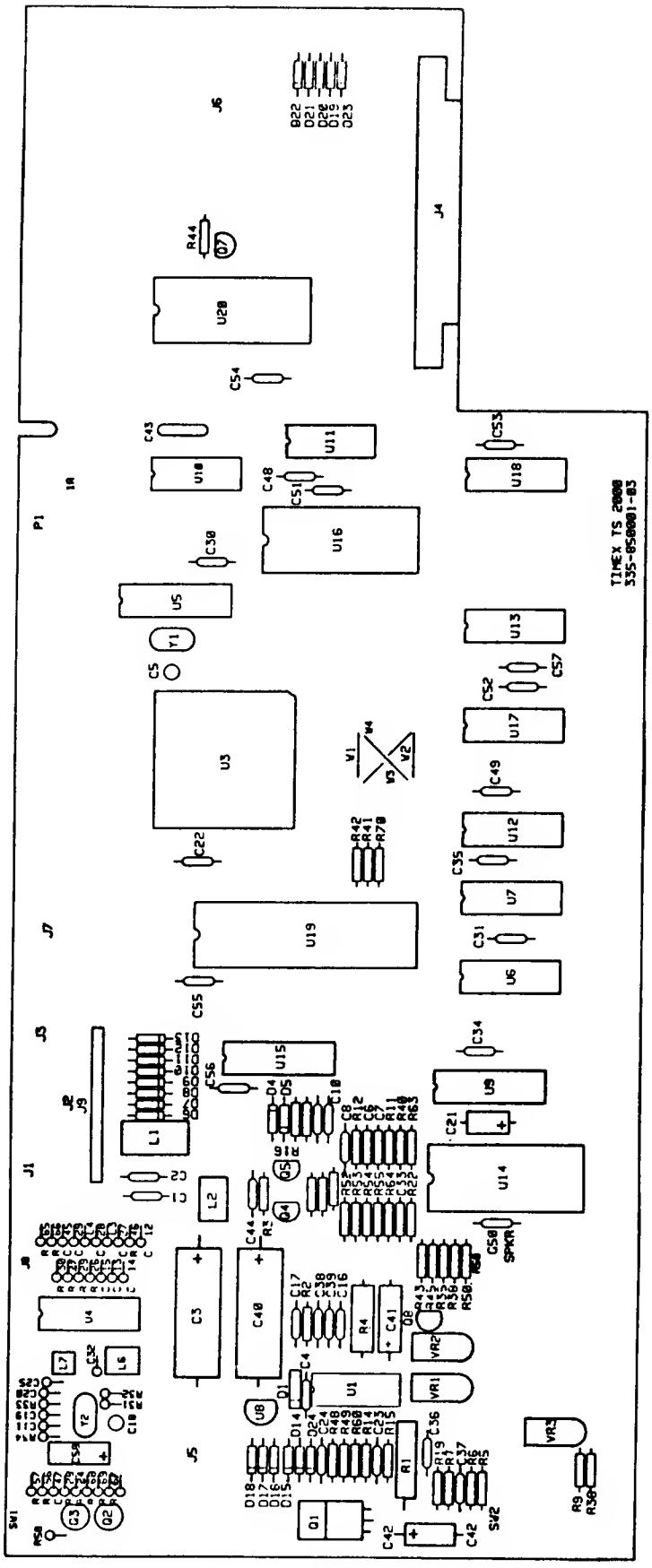
TS2068 PCB Assembly and Schematic Diagram

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.
VR1: U3-33 goes to VR1/Q5
Q4: Connect base to R55/R54
Solder dots on horizontal lines below keyboard:
U12-4 to U3-65 ($\overline{WR}$)
U12-5 to U3-66 ($\overline{MREQ}$)
U5: U5-2 to U3-38 (A7R not A7RB)
P1: P1-4B +15V (not -15V)
U21:





TS2068 PC BOARD COMPONENT LAYOUT

APPENDIX D

TS2068 PARTS LIST

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	COMMENTS
P.C.B. (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	COMMENTS
CAP. 68 pf Ceramic Axial	C70	1	-20 +80% or GMV TEMP Z5U
CAP. 24 pf Ceramic Axial	C29,27	2	-20 +80% 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.100K 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.1.10K OHM 1/4W, +/-1%, MF	R6	1	
RES.3.32K OHM 1/4W, +/-1%, 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	R1	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	CR4,5,6,7,8,9,10 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	1	
IC, UA 78S40 NPC, Switching Regulator	U1	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	1	
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, 74LS00	U21	1	
TRAN. PNP D43C1	Q1	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	L1	1	
Inductor 230 uh	L2	1	
Inductor .33uh Axial	L3,4	2	
Inductor .12uh	L6,7	2	
Crystal Oscillator 14.112 MHz	Y1	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 1/2 MM Pin	J1	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	HS1	1	
Heat Sink Insulation Pad			

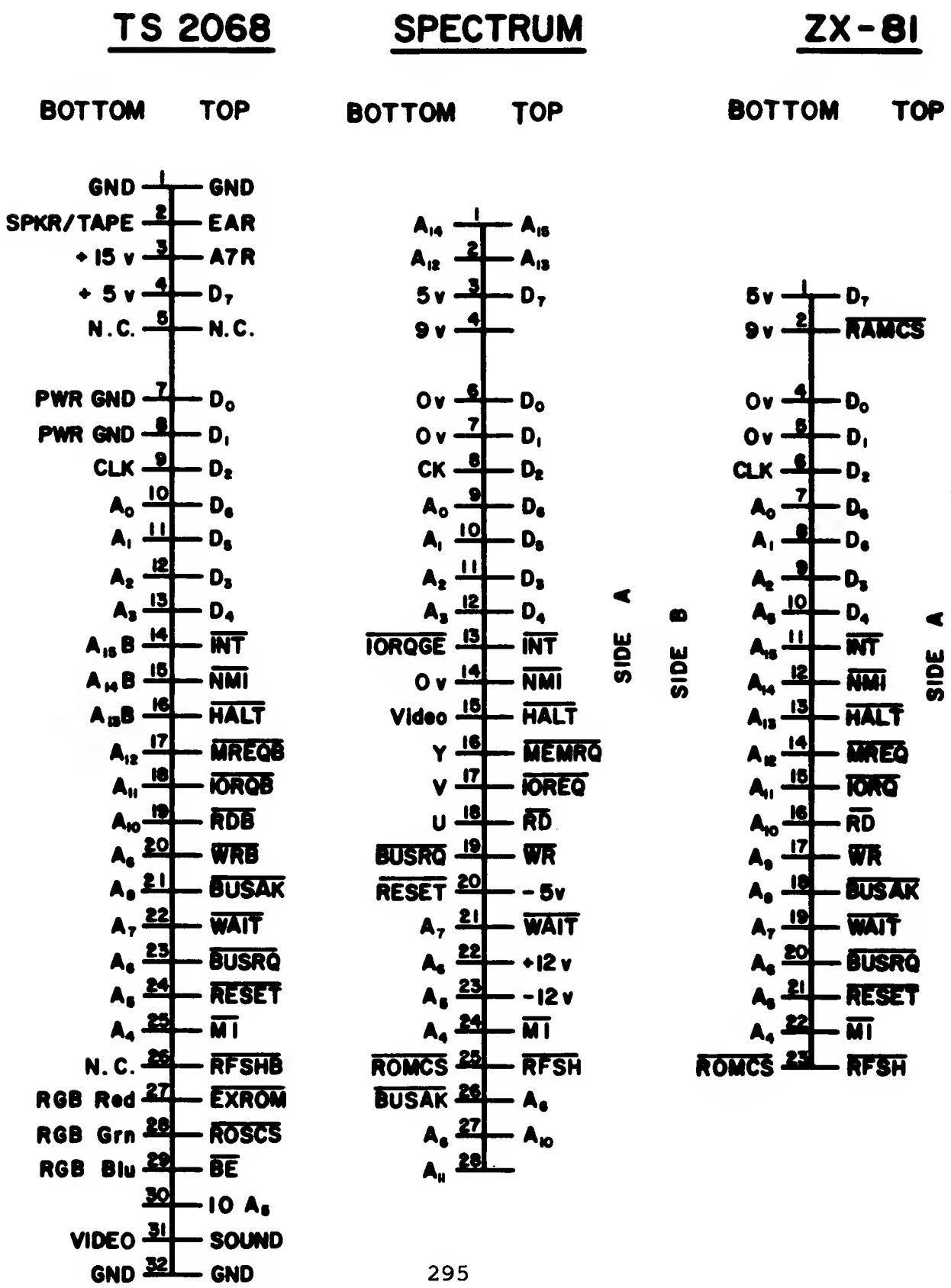
APPENDIX D

TS2068 PARTS LIST (continued)

DESCRIPTION	COMPONENT DESIGNATION	QTY PER ASSY	COMMENTS
Socket, IC, 28 Pin		2	
Socket, IC, 40 Pin		1	
Speaker, 45 OHM, Mylar Cone		1	
Jumper Wire	W1, 2, 50	3	
Ferrite Bead	L5,8	2	
PC Board Assembly,Daughter		1	

APPENDIX E

Expansion Buss Comparison of
TS2068, Sinclair Spectrum and ZX81

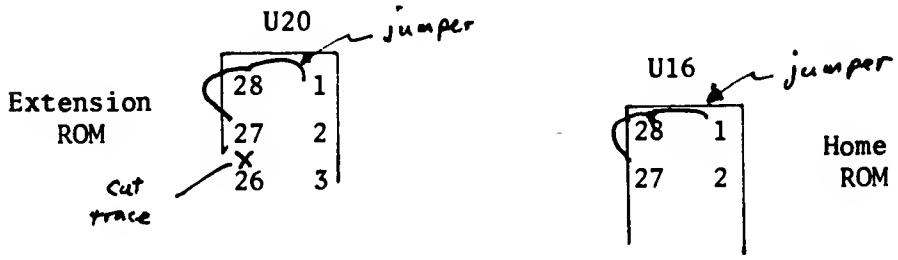


APPENDIX F

August 1985
Bob Orrfelt

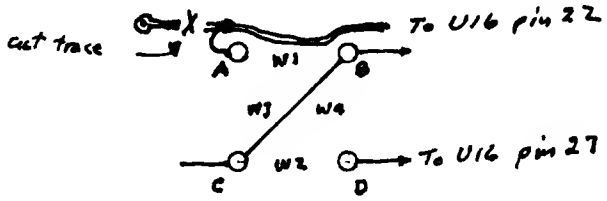
TS2068 MODIFICATIONS FOR EPROMS

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-component side of the pcb,

0. Remove ROMs.
1. Cut the trace between U20-26 and U20-27
2. Jumper pins 1 to 28 to 27 on each socket.



Component side of pcb.

3. Remove the two zero ohm resistors W1 and W2.
4. Cut the trace just above and to the left of hole A.
5. Add a jumper from hole A to the trace. This connects $\overline{\text{MREQ}}$ 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

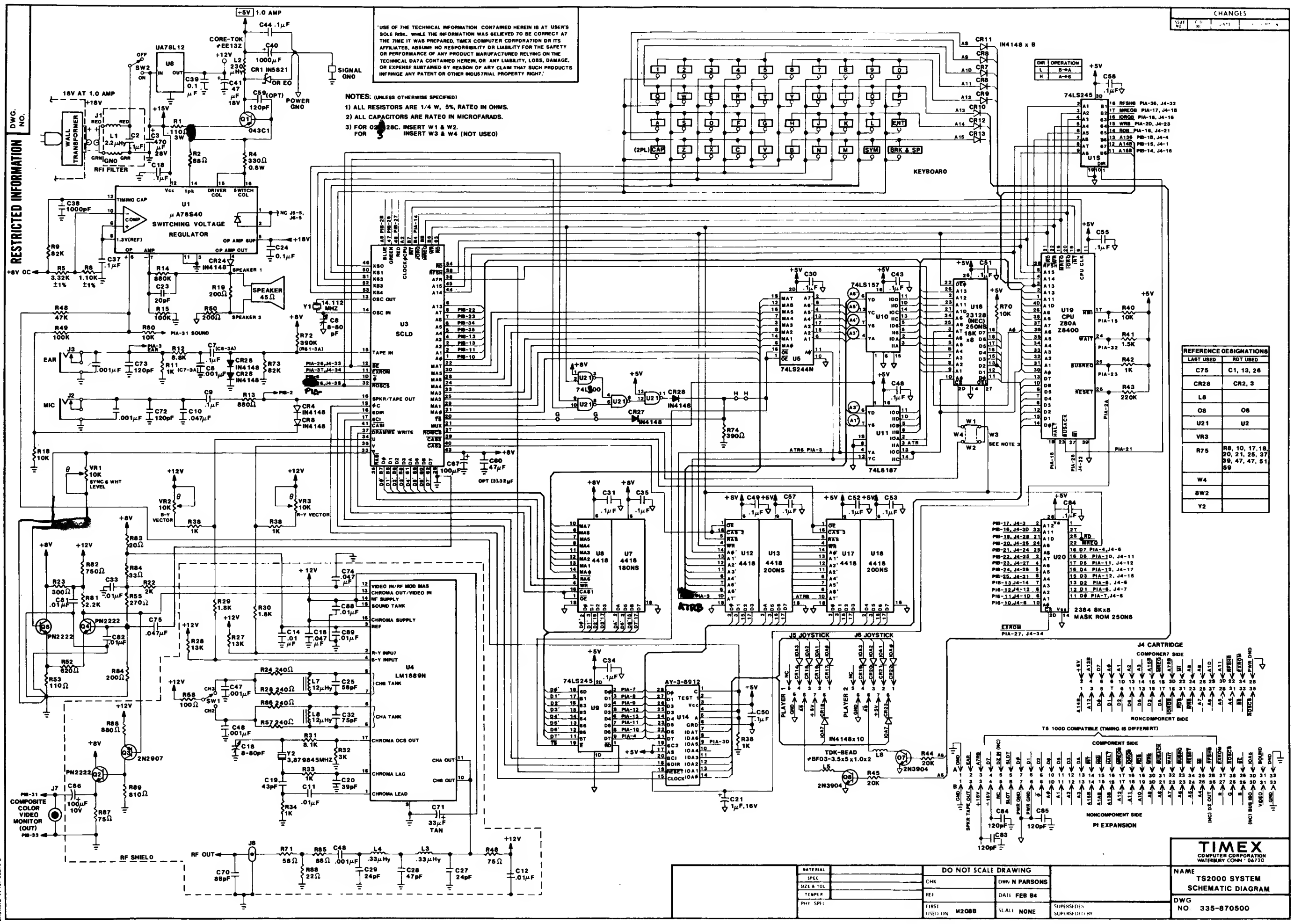
Proposed TS2068 Home ROM Corrections and Improvements

```
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 on message.
(Last character add 80H)
1118 Property of Bob
1128 Orrfelt.........
1138 ................
1148 ..

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   HL
35E7 73        LD    (HL),E
35E8 23        INC   HL
35E9 72        LD    (HL),D
35EA 2B        DEC   HL
35EB 2B        DEC   HL
35EC 2B        DEC   HL
35ED D1        POP   DE
35EE C9        RET

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

35F9 FFFFFFFF blanks
35FD FF

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

More for EPROM programmer.
37B8 D9        EXX
37B9 212B00    LD    HL,002BH
37BC 85        ADD   A,L
37BD 6F        LD    L,A
37BE 6E        LD    L,(HL)
37BF 2636      LD    H,36H

37C1 D9        EXX
37C2 AF        XOR   A
37C3 C9        RET
37C4 00        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 02128C. INSERT W1 & W2.
FOR INSERT W3 & W4 (NOT USED)

REFERENCE DESIGNATIONS	
LAST USED	NOT USED
C75	C1, 13, 26
CR28	CR2, 3
L8	
Q8	Q8
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 M2088	SCALE NONE
PHY SPEC		SUPERSEDES	SUPERSEDED BY

TIMEX
COMPUTER CORPORATION
WATERBURY CONN 06720

NAME TS2000 SYSTEM SCHEMATIC DIAGRAM

DWG NO 335-870500