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

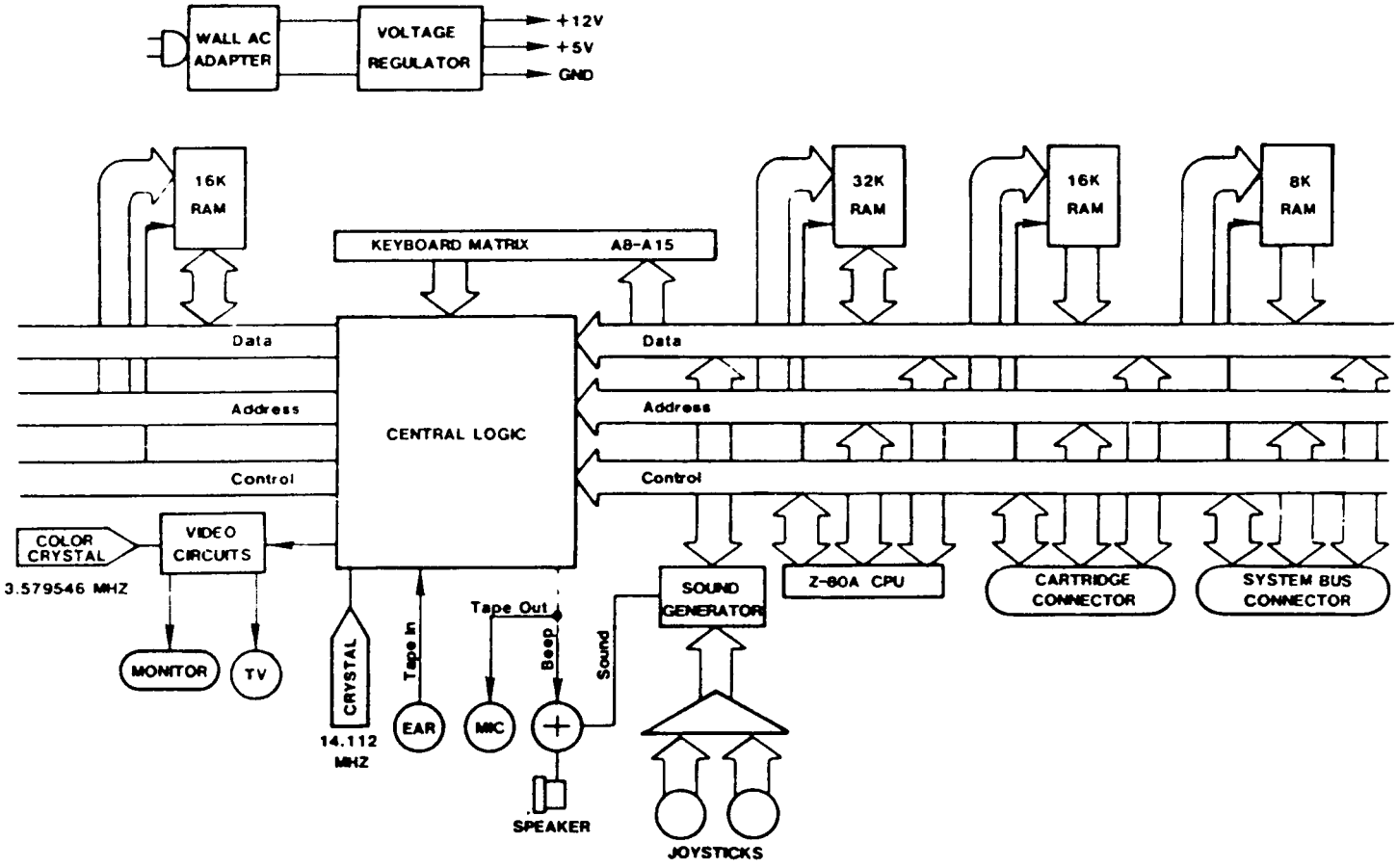
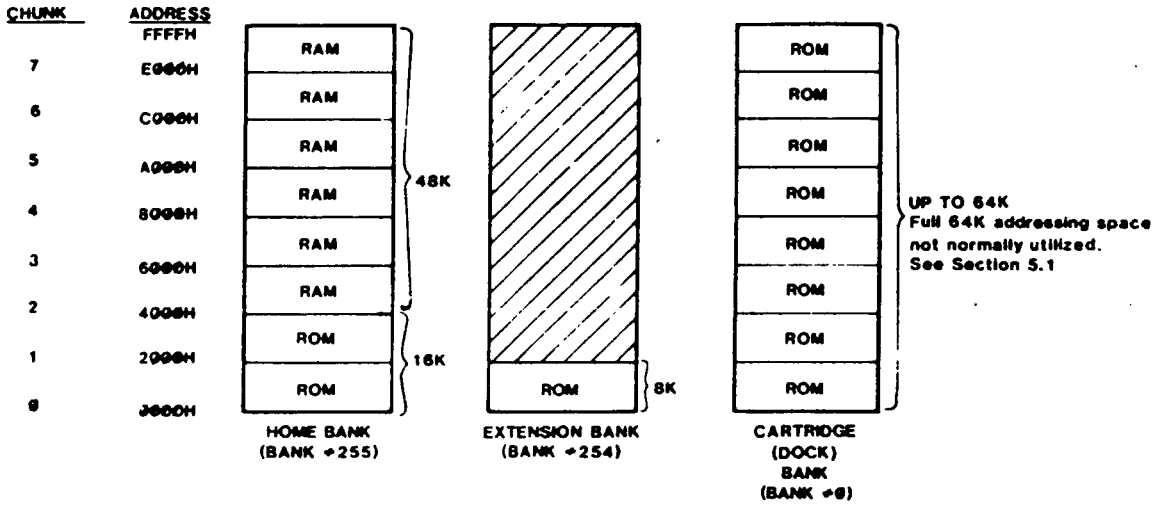


FIGURE 1.1-2

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

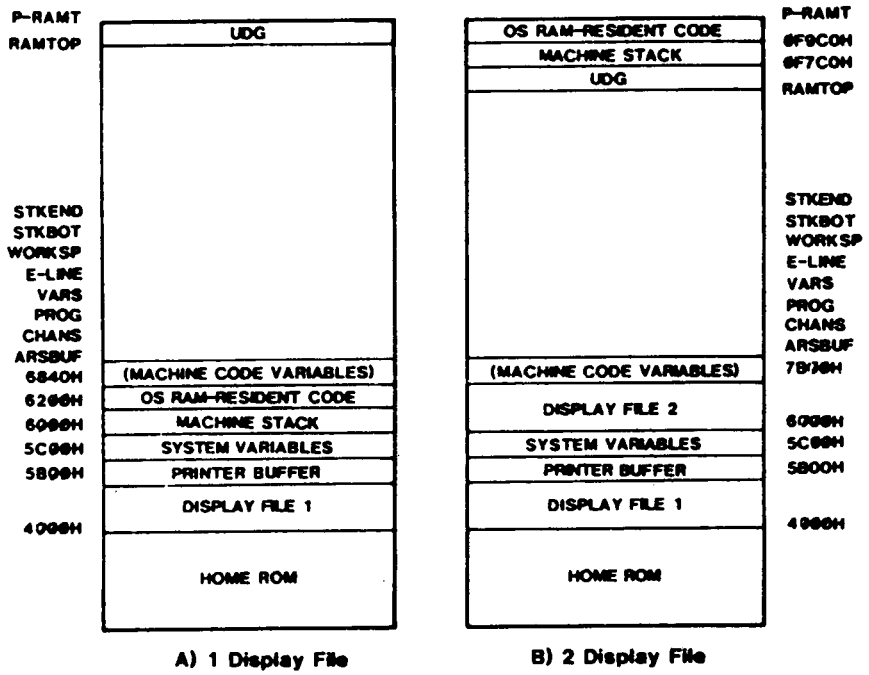
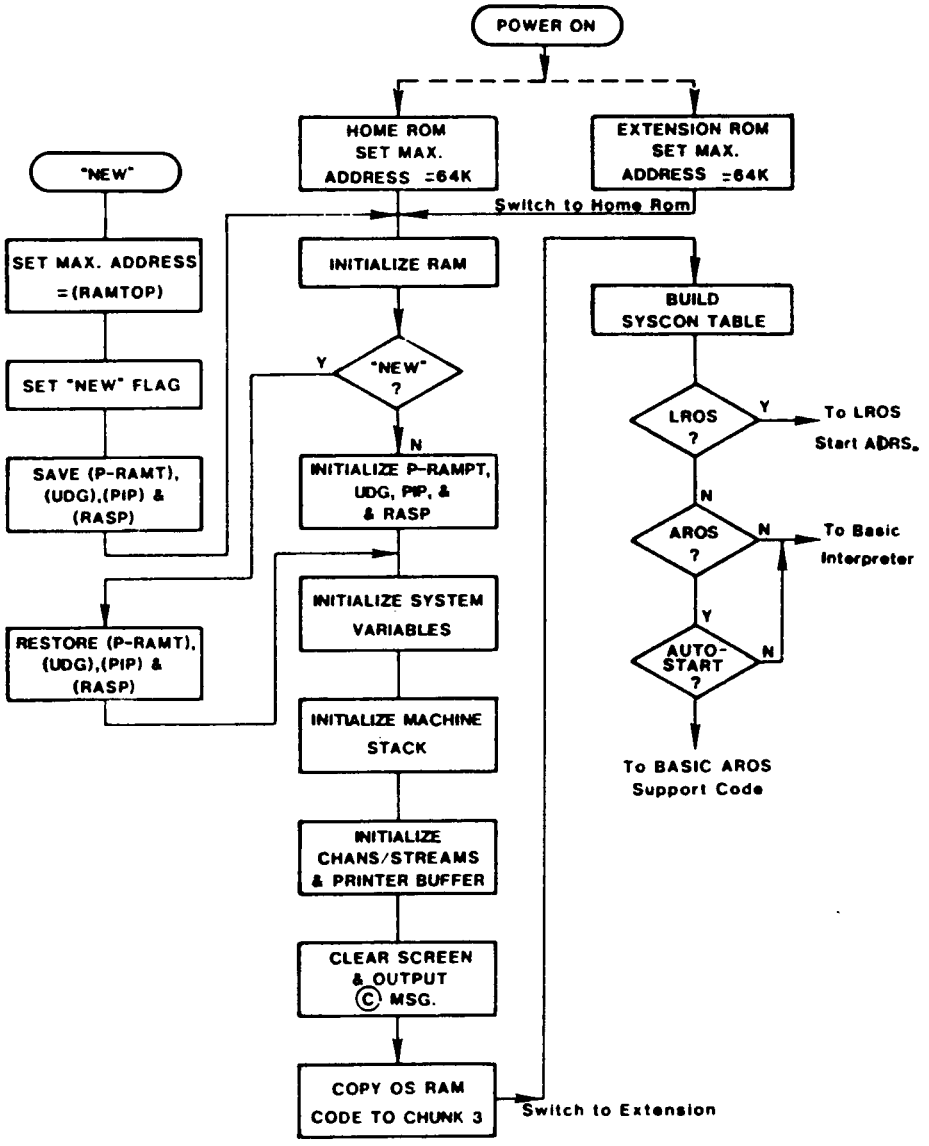


FIGURE 1.1-4

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

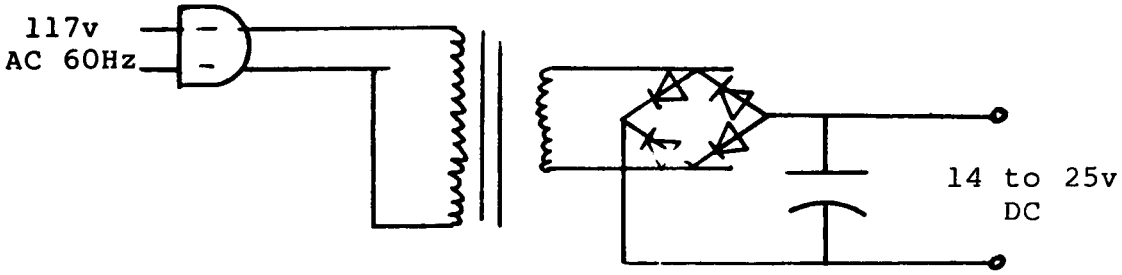
2.1 Description of Major Hardware Functions

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

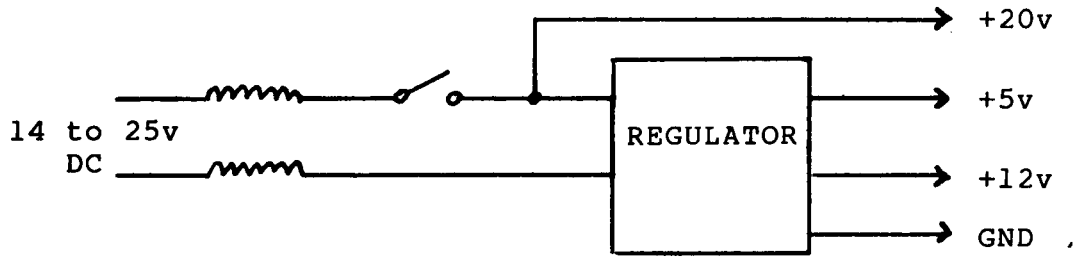
The AC Adapter transforms 117V AC (Nominal) to filtered DC via a step down transformer, full-wave bridge rectifier, and filter capacitor to supply from 14 to 25 volts at 1 amp over the AC voltage variation range of 105 to 130 V AC. Transformer isolation exceeds 1500 volts.



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

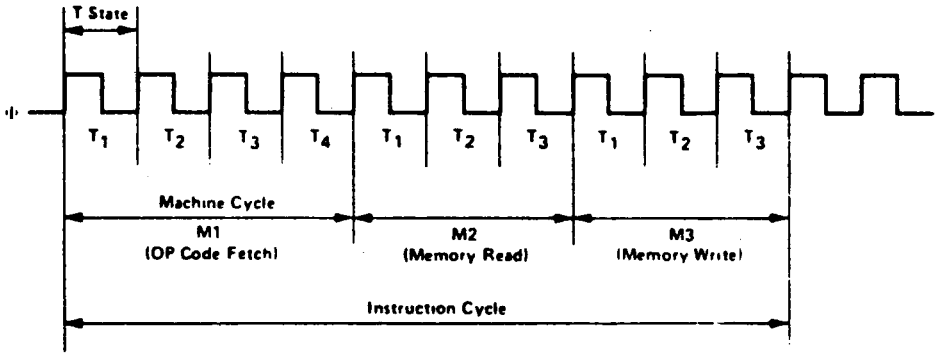
The Z-80 CPU executes instructions by proceeding through a sequence of operations that include:

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

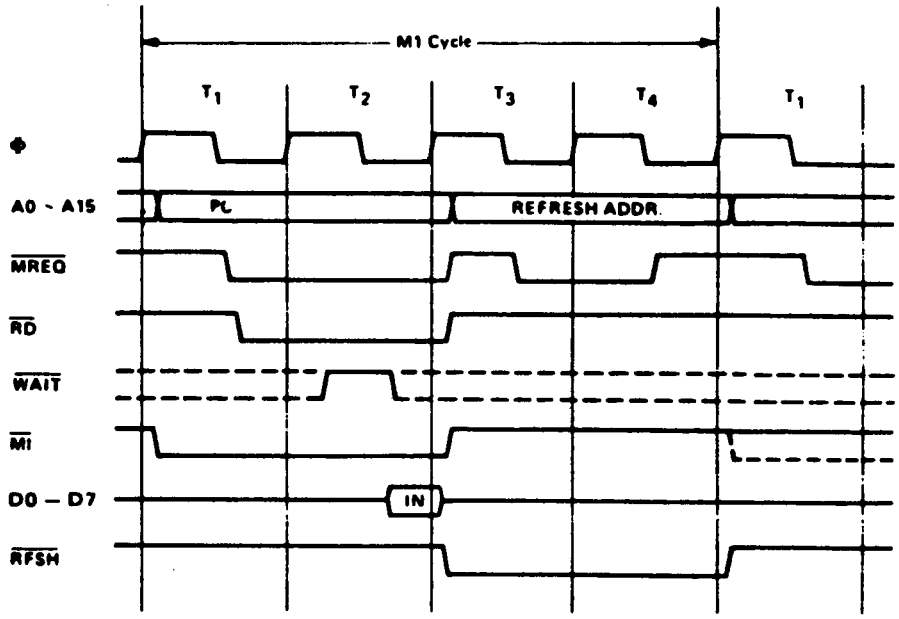
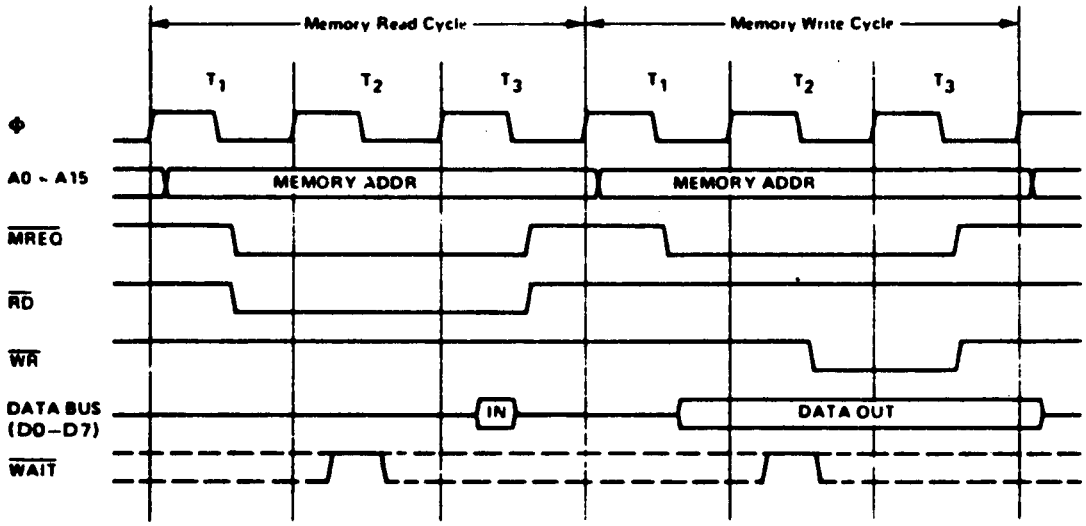


FIGURE 2.1.3-3

MEMORY READ OR WRITE CYCLES

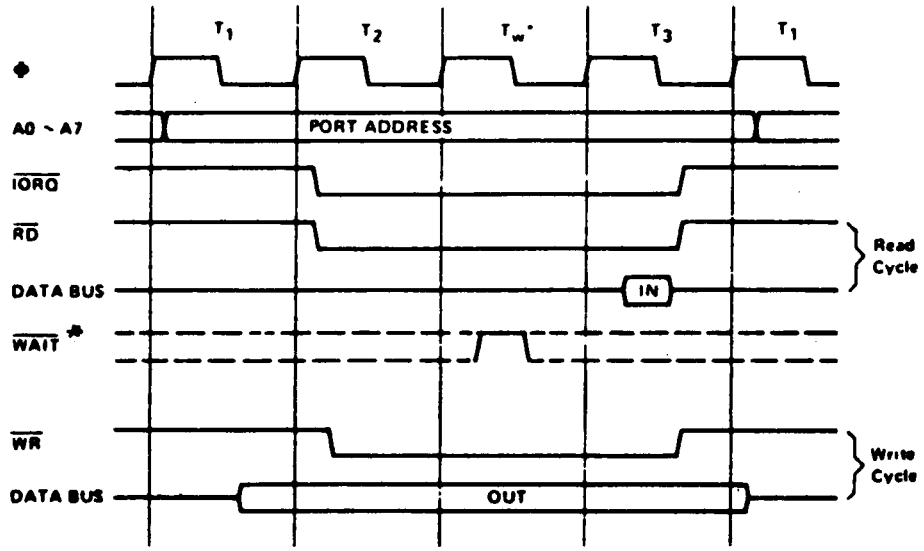


2.1.3.6 I/O READ/WRITE

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



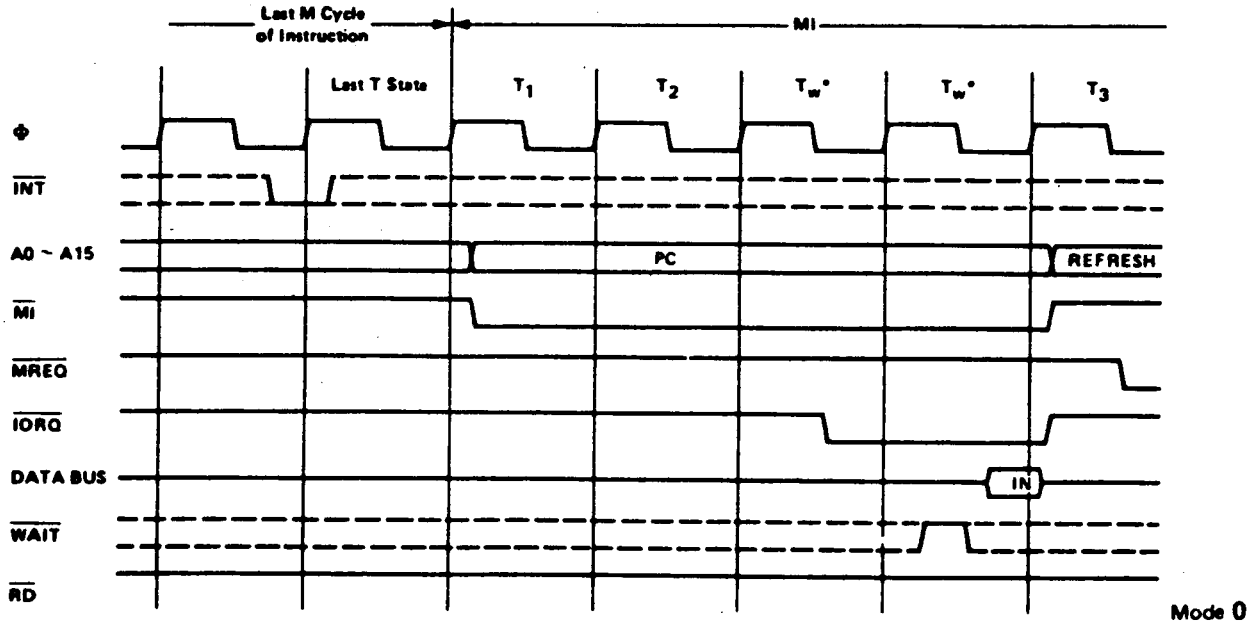
*Inserted by Z80 CPU

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.

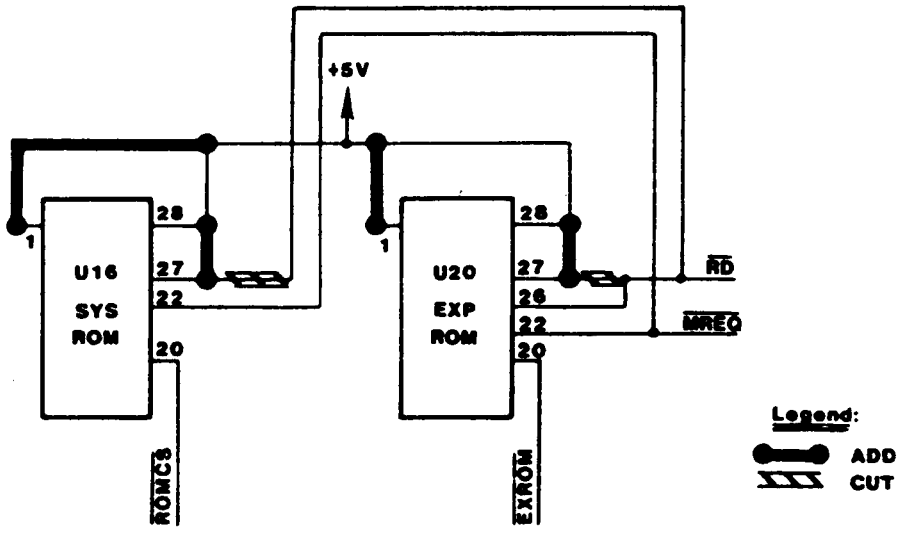
TABLE 2-1

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 $\overline{\text{BUSRQ}}$.

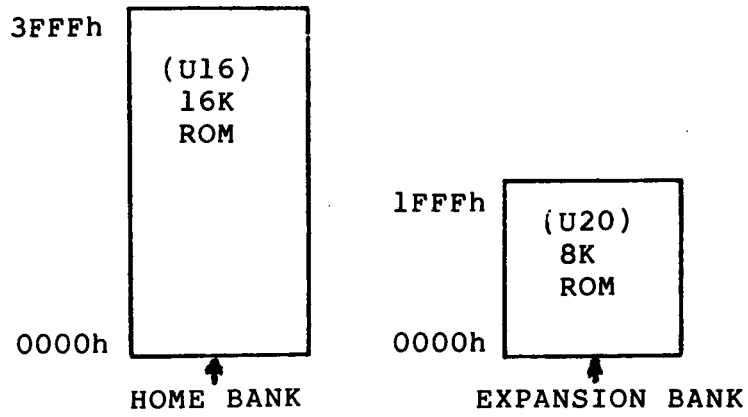
Figure 2.1.4-1

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 OF5H (Address) and OF6H (Data). The basic registers in the PSG which produce the programmed sounds include:

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

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

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

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

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

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

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

2.1.6.1 Tone Generator Control (Registers 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).

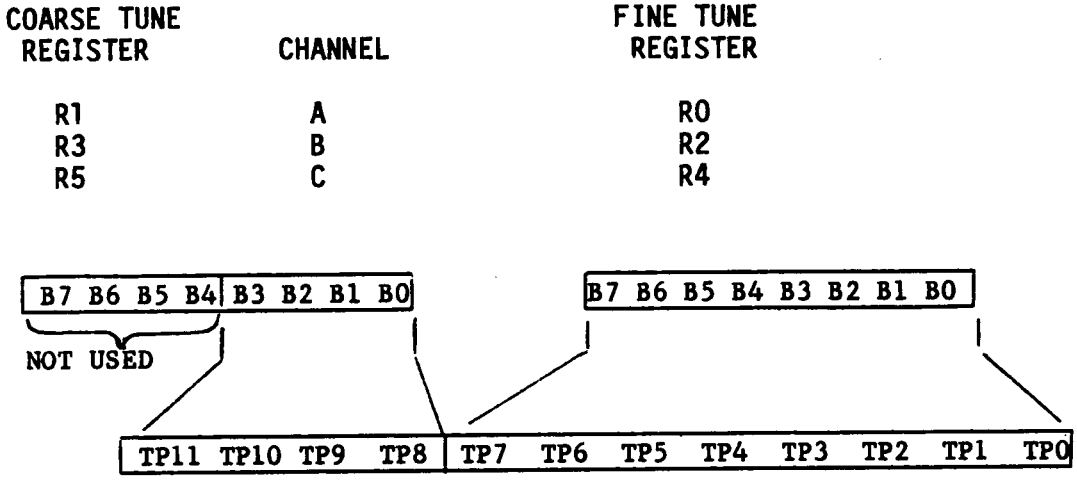
FIGURE 2.1.6-1

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:

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

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:

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

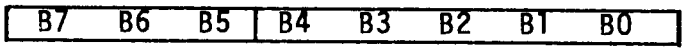
$$FT_{10} = 79_{10} = 01001111_{2}\ (B7\text{-}B0)$$

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

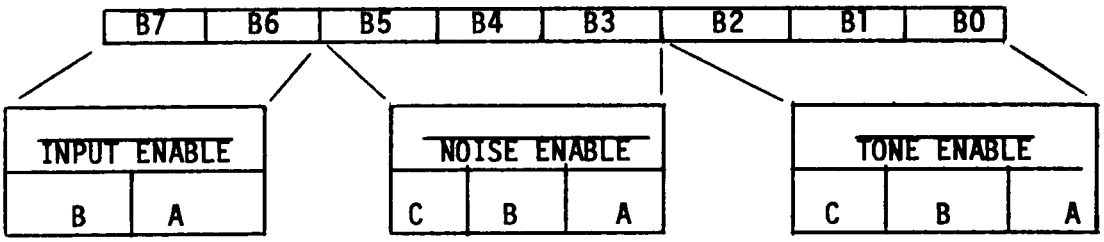


TABLE 2.1.6-1

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

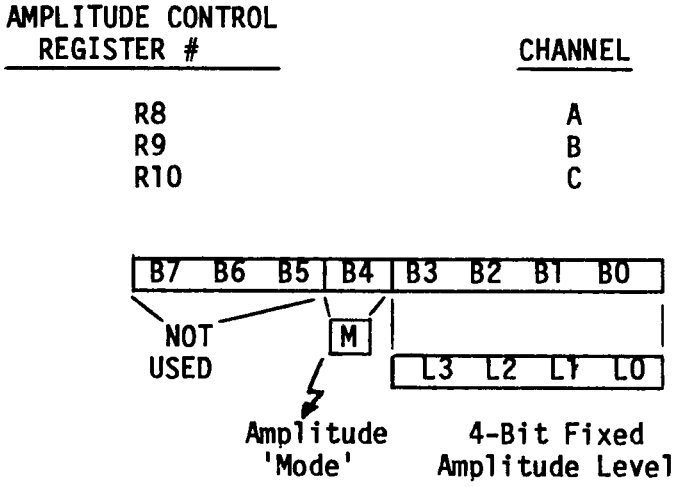
Disabling noise and tone does not turn off a channel. Turning a channel off can only be accomplished by writing all zeroes into the corresponding Amplitude Control register, R8, R9 or R10 (refer to Paragraph 2.1.6.4).

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)								

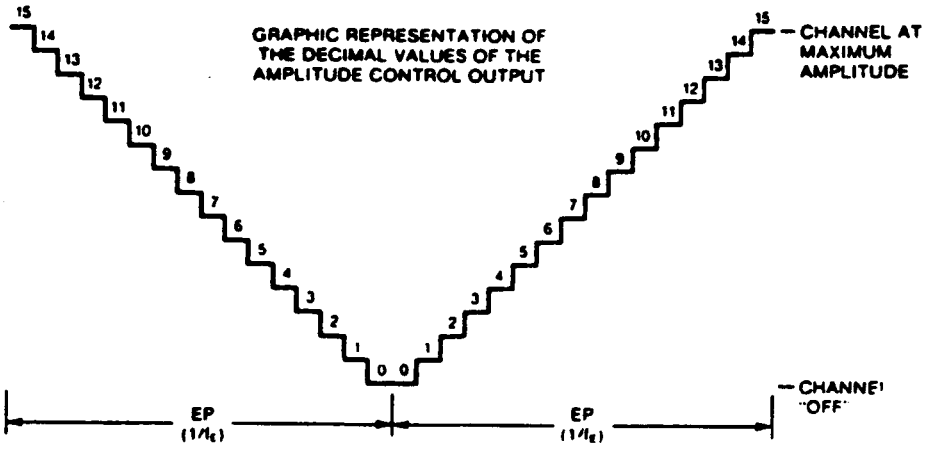
*The all zeros code is used to turn a channel "off".

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)

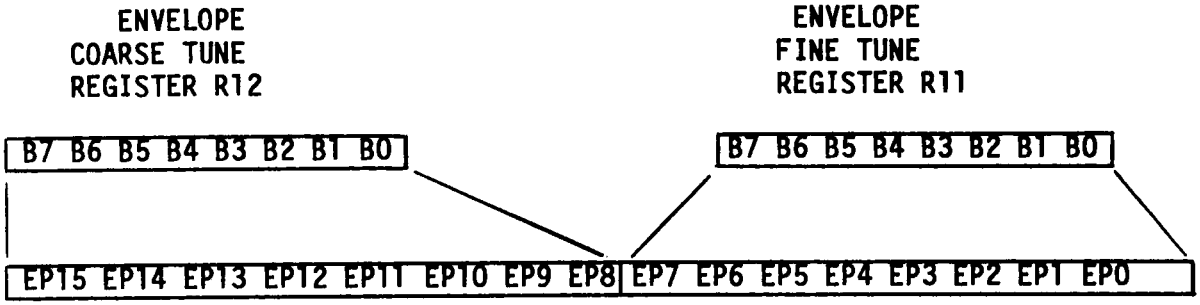
To accomplish the generation of fairly complex envelope patterns, two independent methods of control are provided in the PSG; first, it is possible to vary the frequency of the envelope using registers R11 and R12; and second, the relative shape and cycle pattern of the envelope can be varied using register 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_2 (divide by 1); the highest period value is 1111111111111111_2 (divide by $65{,}535_{10}$).

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:

$$(a)\quad EP_{10} = \frac{fCLOCK}{256fE} \qquad (b)\quad CT_{10} + \frac{FT_{10}}{256} = \frac{EP_{10}}{256}$$

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):

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

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

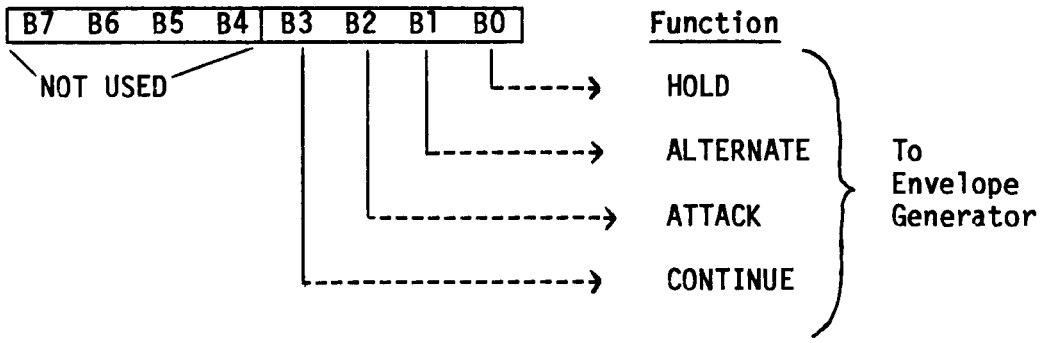
$$FT_{10} = 219 = 11011011_{2}\quad (B7\text{-}B0)$$

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

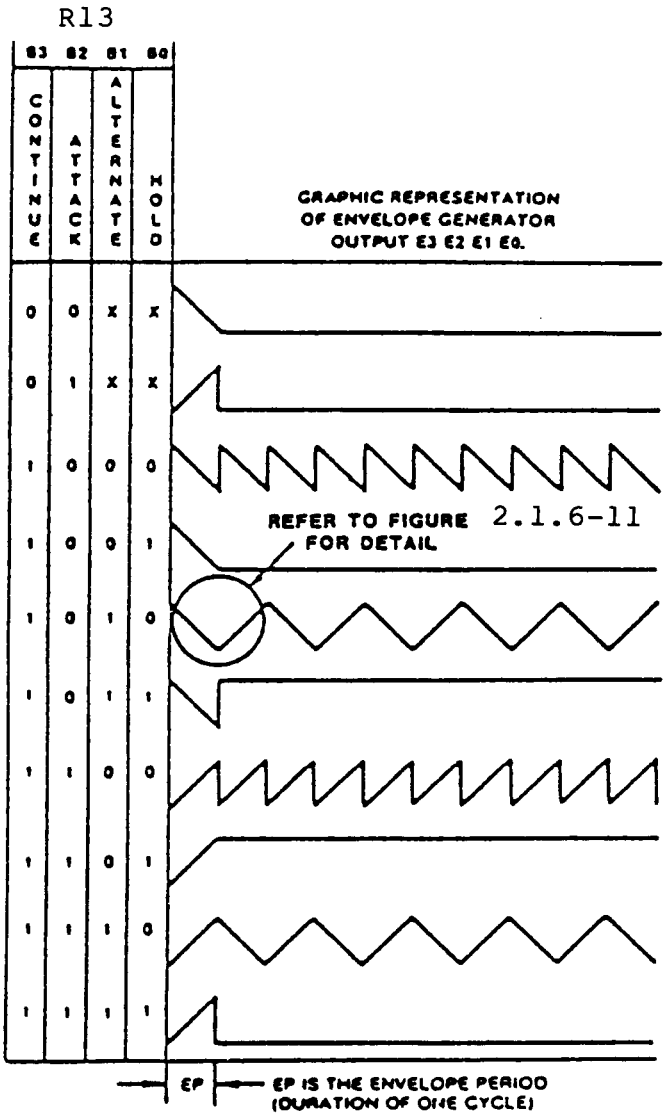
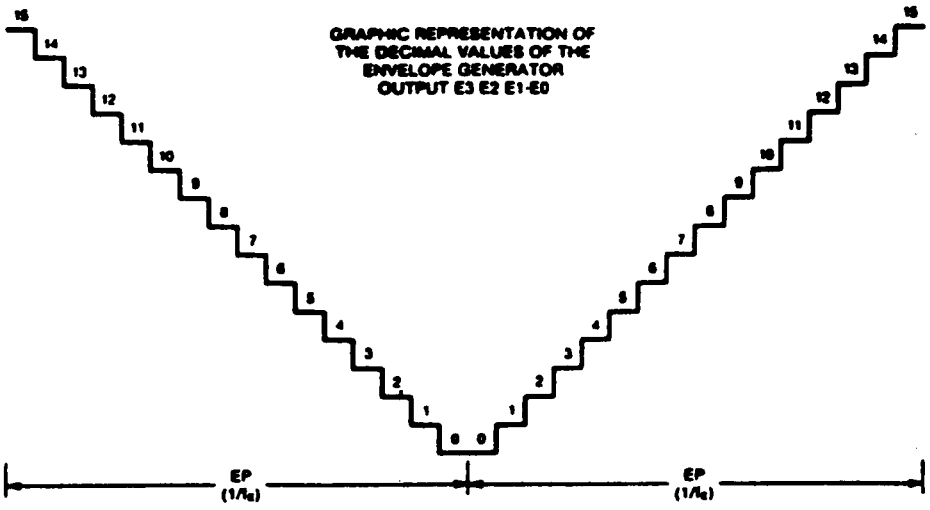


FIGURE 2.1.6-11

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:

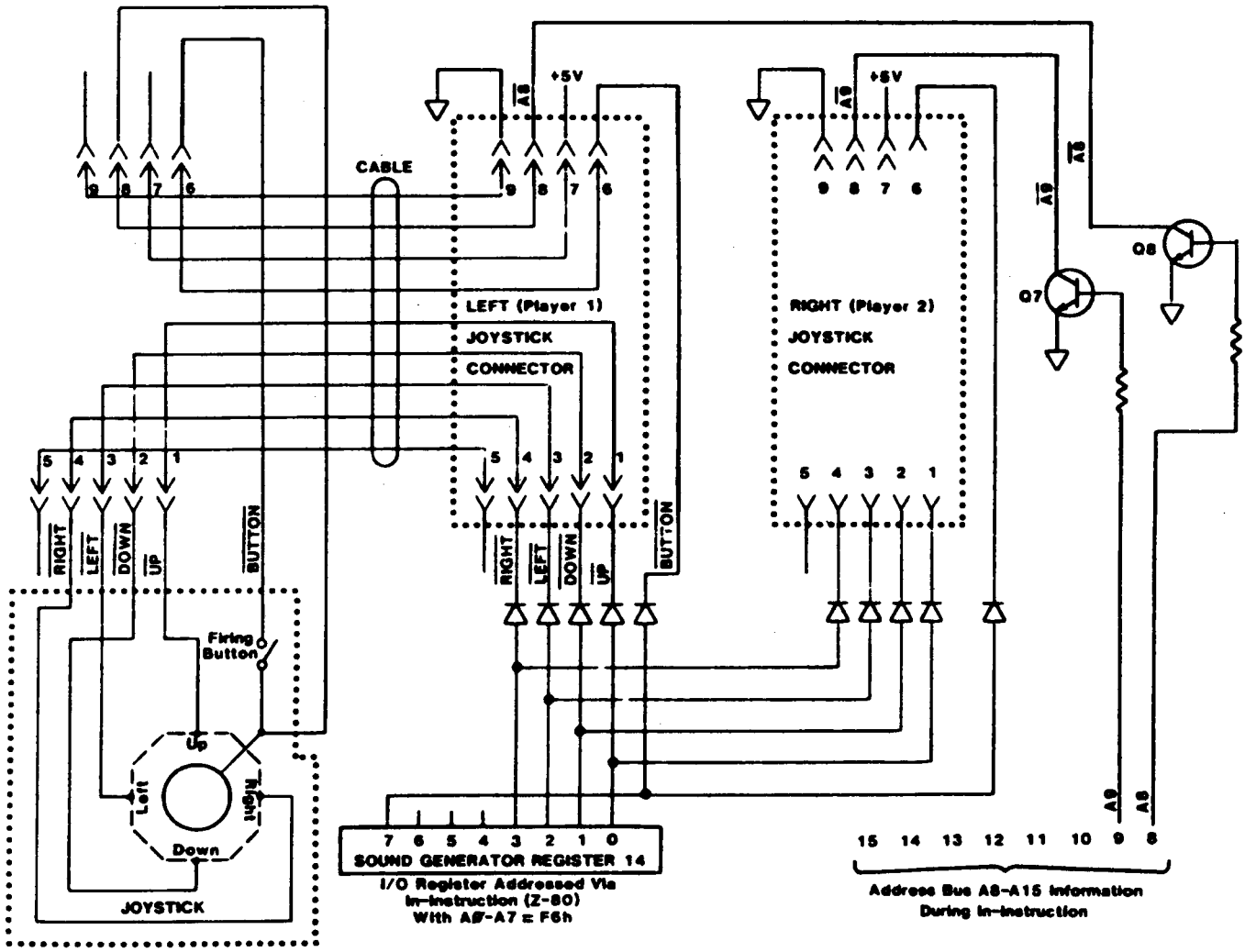
(1) Address A8 high turns on transistor Q8
(2) Q8 drives cable pin-8 low
(3) The movable center stick of the joystick in contact with the up-contact results in a conductive path from cable pin-8 to cable pin-1.
(4) Pin-1 low results in a 0 in bit position 0 of the I/O register via the isolation diode.

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

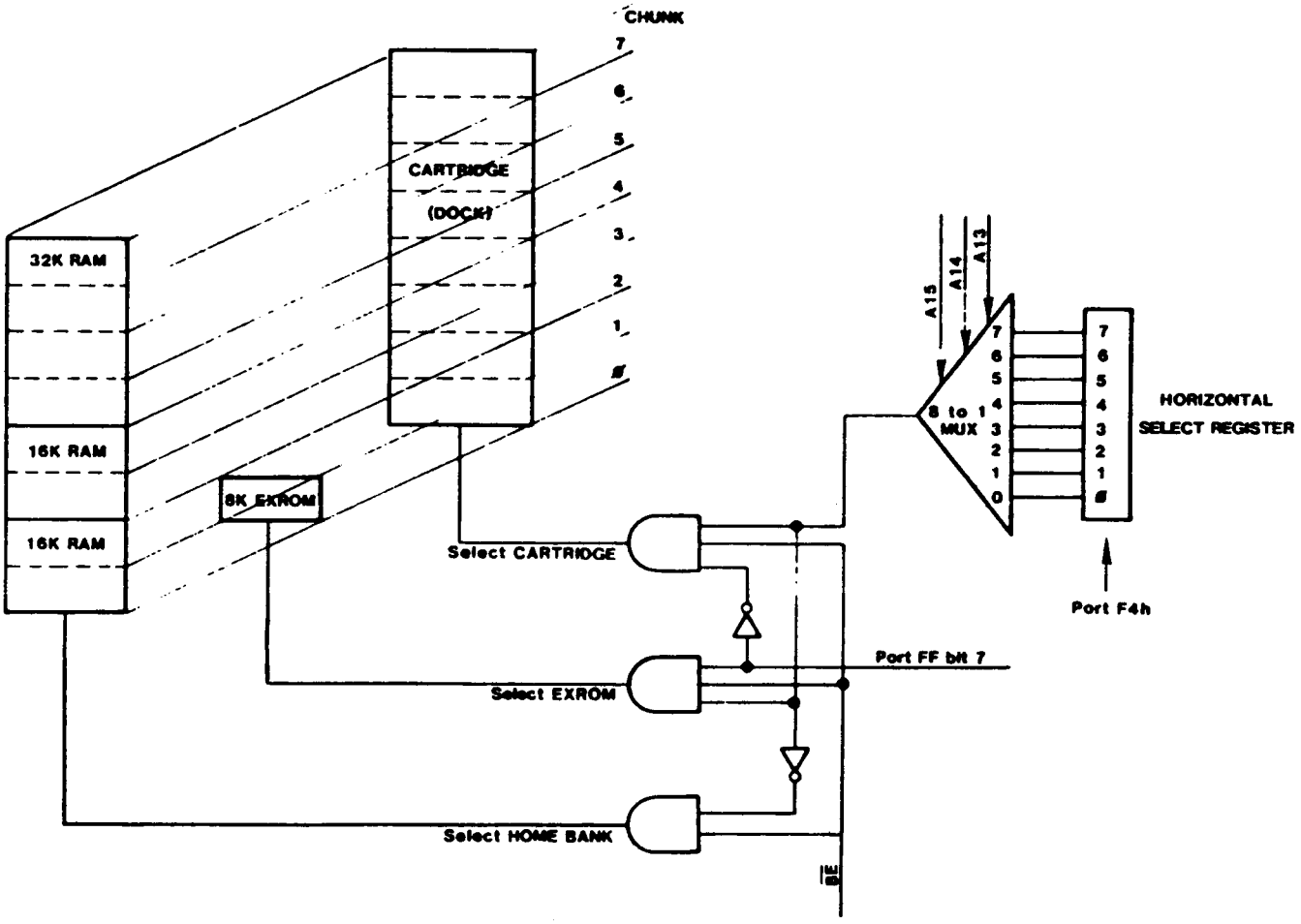
(2) SELECT EXROM of Figure 2.1.8-1 involves activating $\overline{EXROM}$ to its low active state

(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



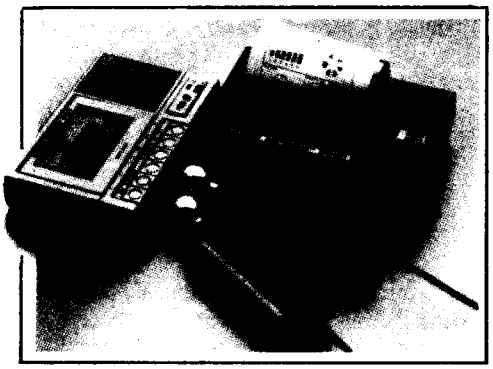


TABLE 2.1.8-1

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

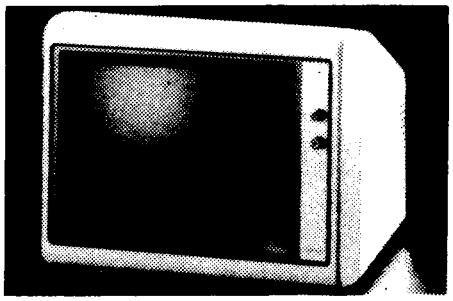
SCLD I/O PIN FUNCTION DEFINITIONS
(continued)

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.

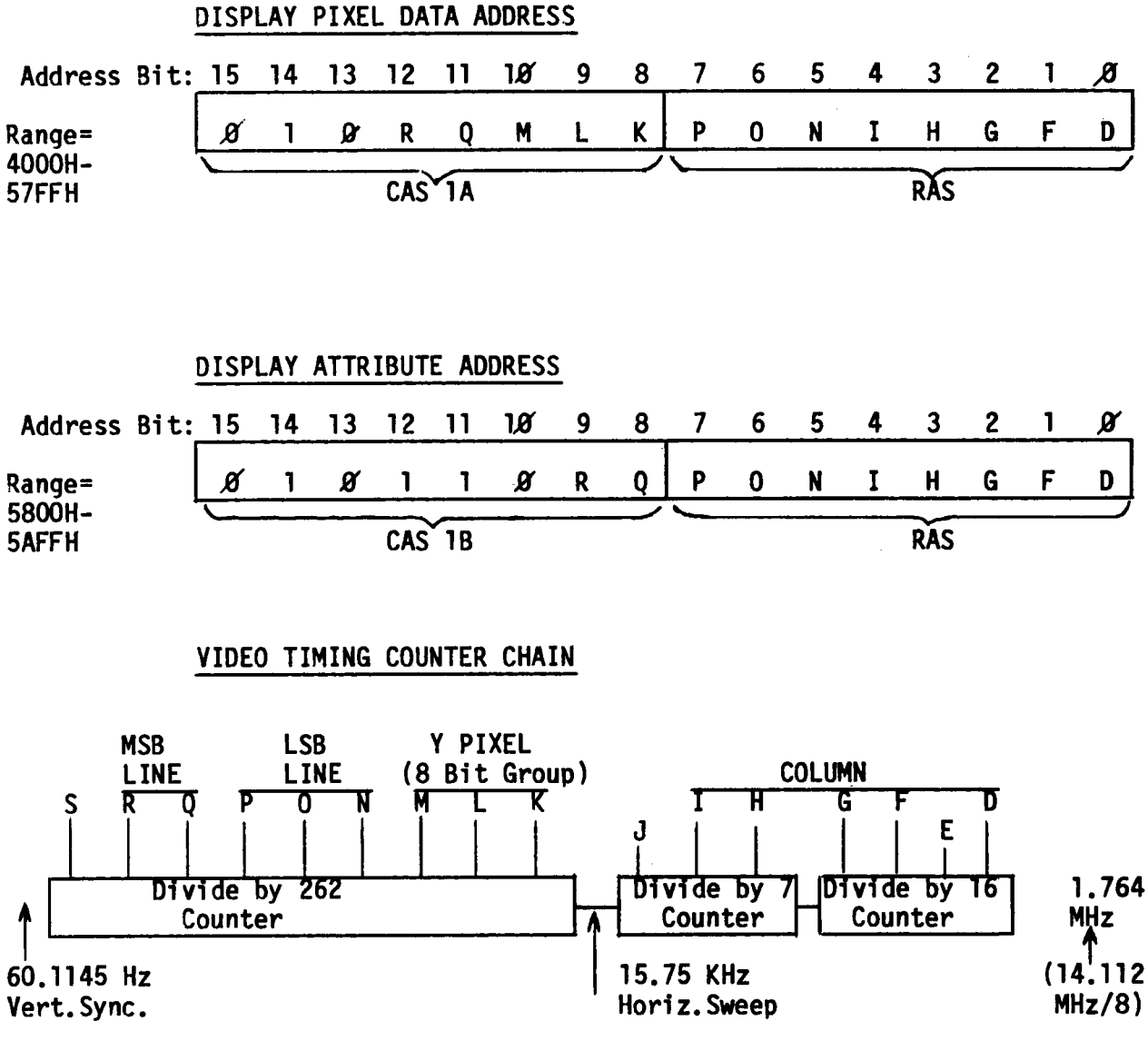
During each horizontal scan the video display processor accesses, in the standard video mode, 32 bytes of pixel data plus 32 bytes of attributes by 32 memory accesses reading 2 bytes per access in RAM page mode, i.e. the low order address bits are provided to the RAM once via RAS activation, then the 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 15.635 ms) the SCLD, if enabled by the INTEN bit (Bit 6) of I/O Port FFH, activates the $\overline{INT}$ signal which directly connects to the $\overline{INT}$ input to the 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

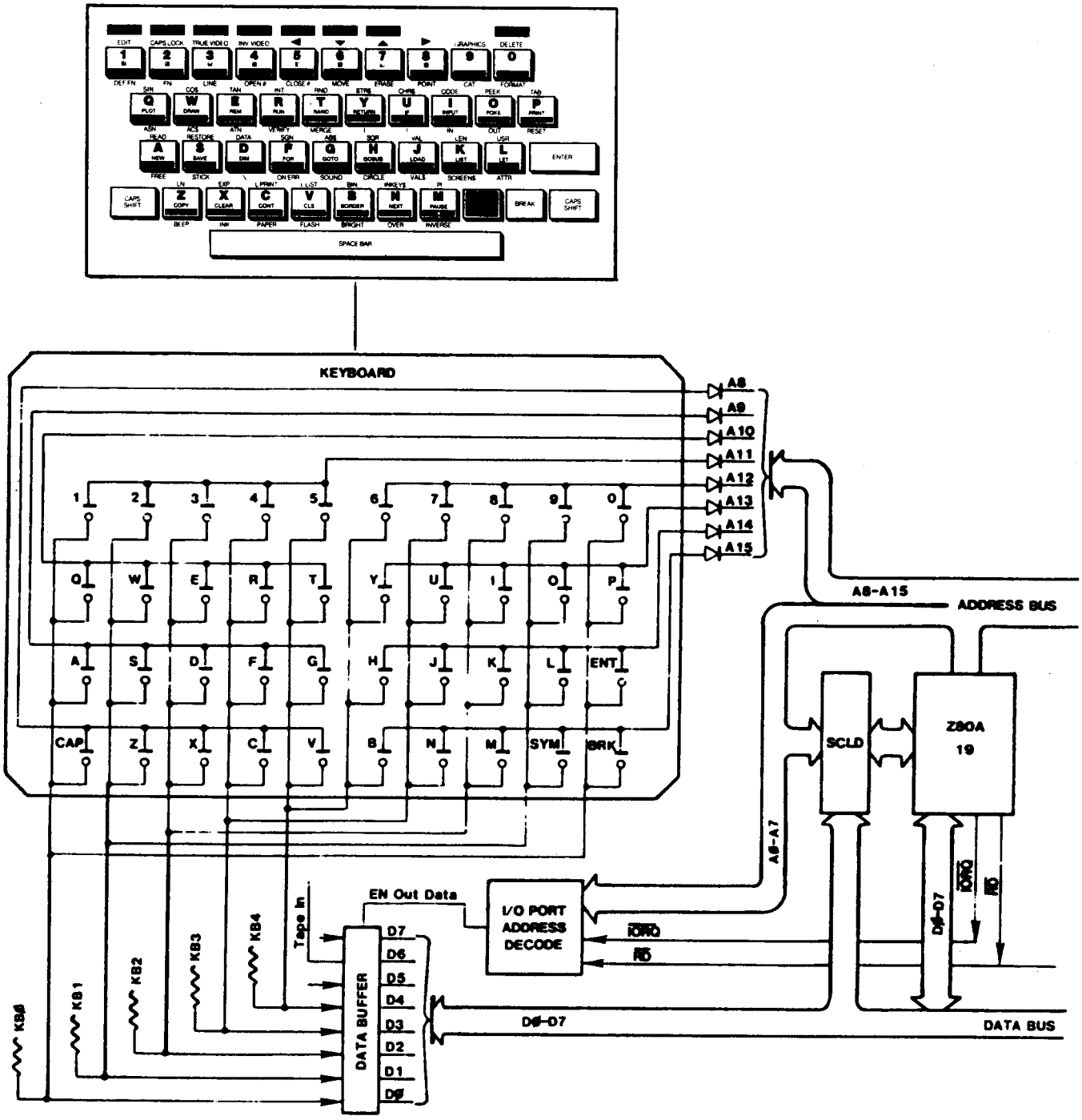


FIG. 2.1.10-1
DISPLAY FILE ORGANIZATION (NORMAL MODE)

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

2.1.11 Video Generation

2.1.11.1 Composite Video

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

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

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

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

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

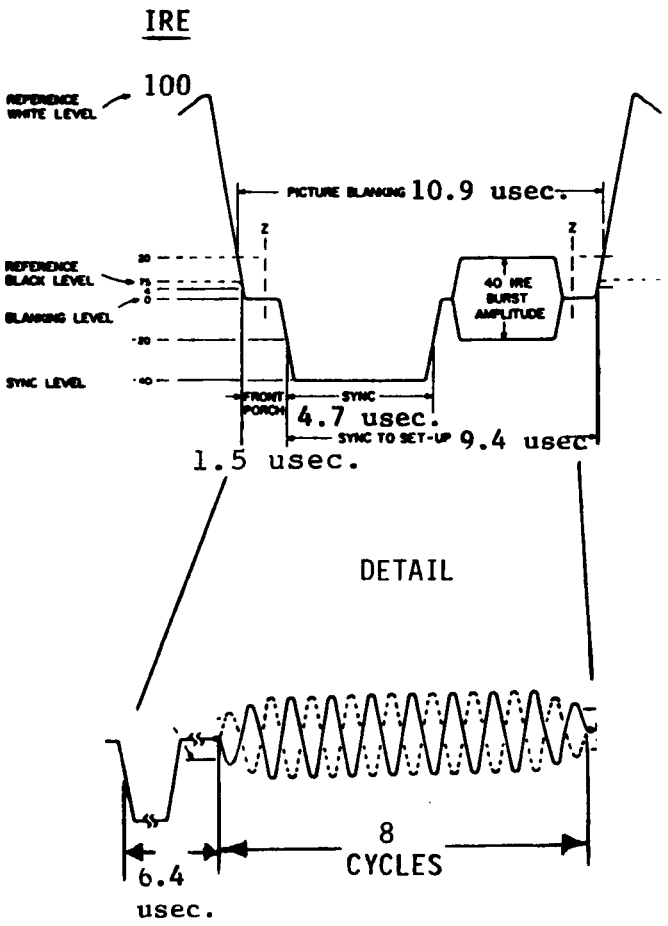
1. The color burst is not synchronous with the waveform since it is generated from the 3.579545 MHz crystal and the waveform is derived from the 14.112 MHz crystal. The result is observed ripples at color boundaries, e.g. green to magenta.

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

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

FIGURE 2.1.11-1

COMPOSITE VIDEO SIGNALS



2.1.11.2 RF Modulator

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

2.1.12 Cassette I/O

See Sections 2.1.13.2, 2.4.3 and 4.2.

2.1.13 Port Map

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

2.1.13.1 Display Enhancement Control (Port FFH)

The display enhancement control register within the SCLD controls:

a) Selection of Enhanced Video Modes

b) Ink selection for 64-Column Mode

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

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

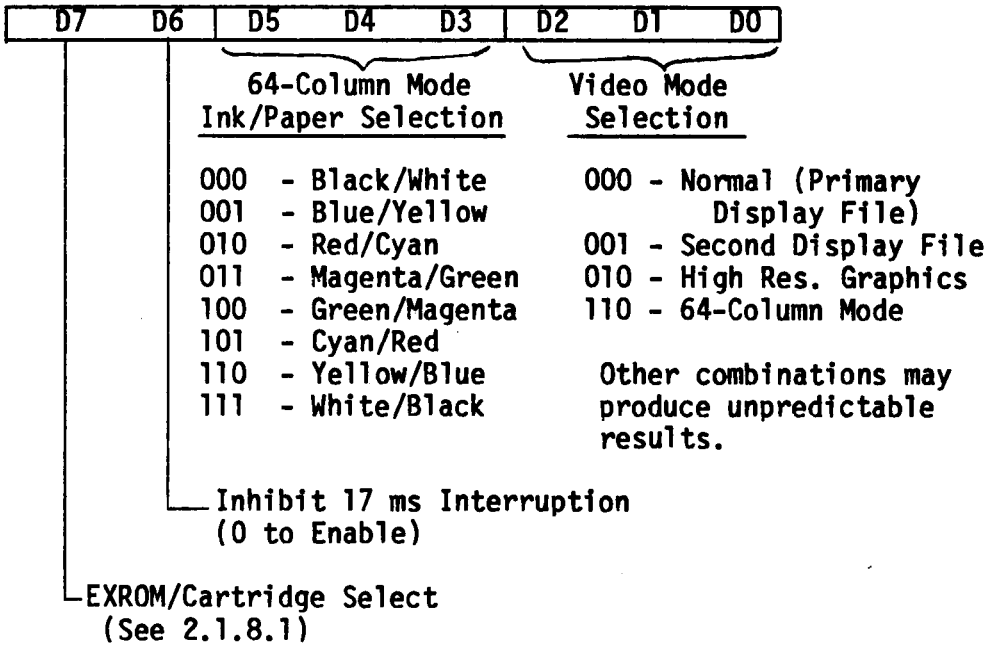


TABLE 2.1.13-1

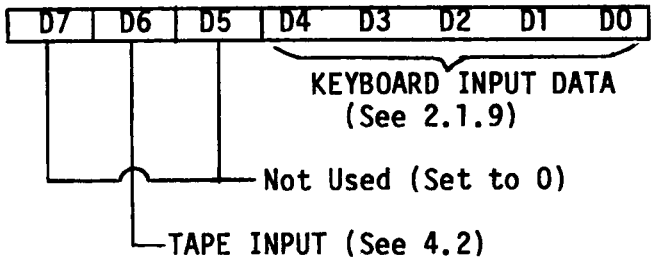
I/O PORT MAP

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

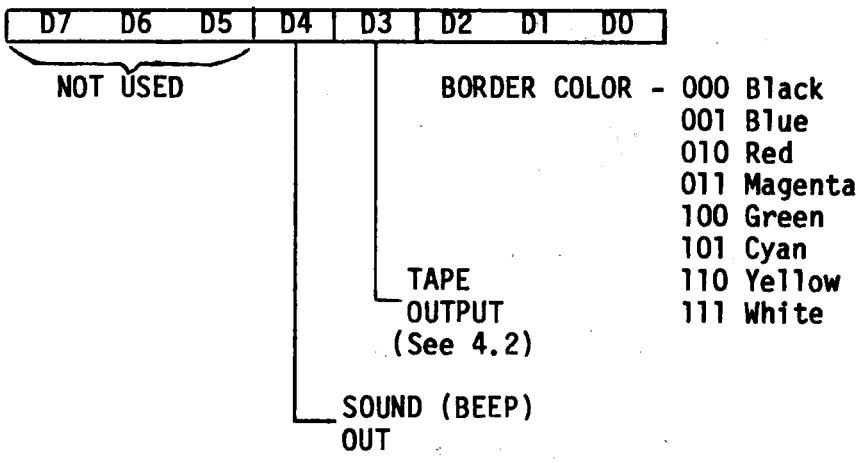
2.1.13.2 Keyboard/Tape I/O (Port FEH)

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

READ (IN)



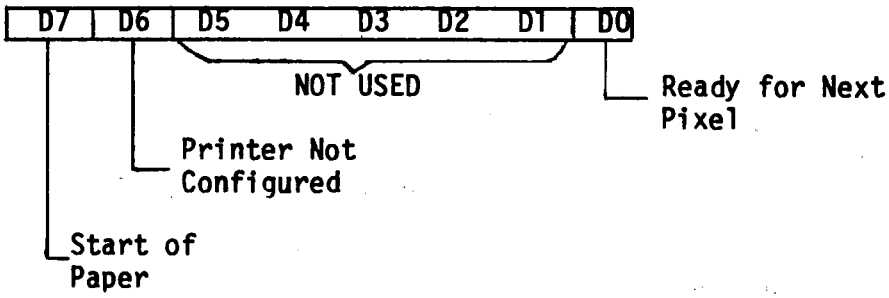
WRITE (OUT)



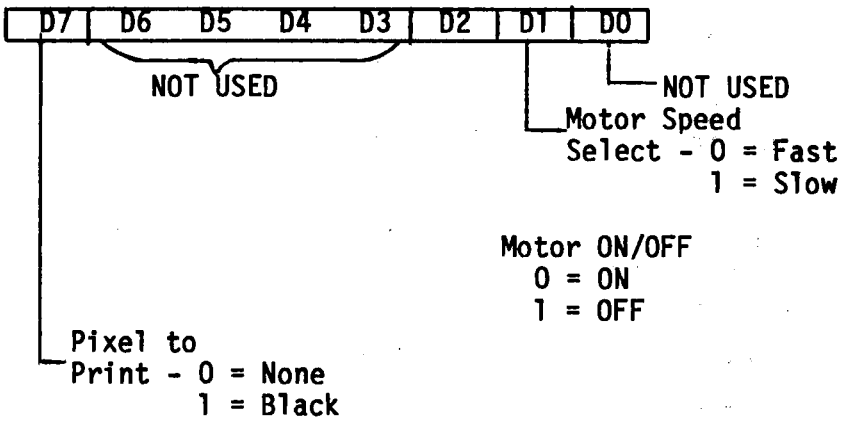
2.1.13.3 TS 2040 Printer (Port 1XXXXOXX)

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

READ (IN)



WRITE (OUT)



2.1.13.4 Sound Chip & Joystick (Ports F5H and F6H)

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

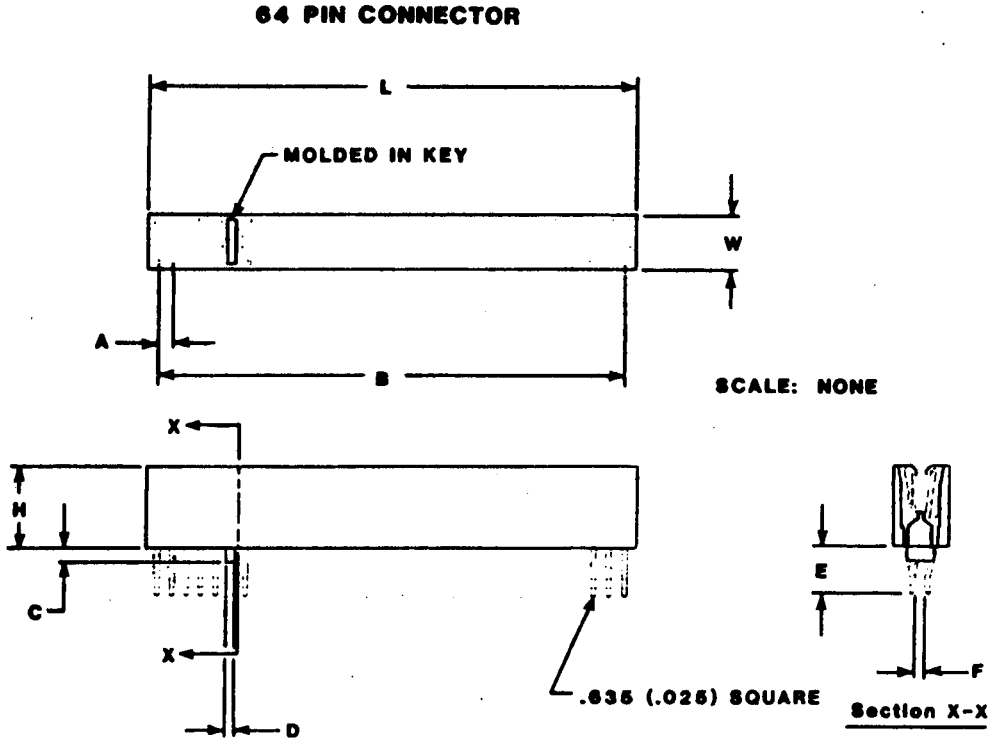
2.4.1 System Bus Connector - P1

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

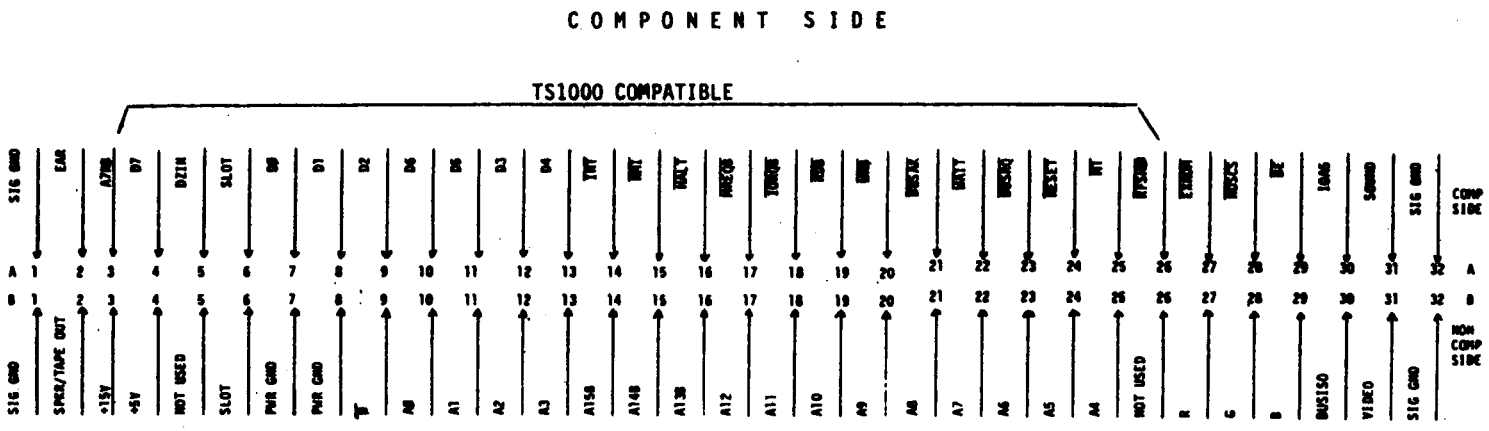
*All dimensions are in millimeters, dimensions shown (XX.X) are in inches.

NOTES:

1. INSULATOR MATERIAL: Insulator body shall be 30% glass-filled polyester and shall meet UL94V-0 requirements.
2. CONTACT MATERIAL: Contact material shall be phosphor bronze.
3. CONTACT FINISH: Contacts shall be selectively plated with gold, 0.00038 (.000015) thick over nickel on contact surfaces.
4. INSERTION FORCE: Insertion forces shall be 170.1-283.5 grams (6-10 oz) per contact pair using a 1.575 (.062) flat steel test blade
5. WITHDRAWAL FORCE: Withdrawal forces shall be 226.8-340.2 grams (8-12 oz) per contact pair using a 1.575 (.062) flat test blade.
6. NORMAL FORCE: Normal force shall be 85.05 grams (3 oz) minimum when mated with a 1.37 (.054) thick test board.
7. PURCHASE FROM: San Diego Microtronics INC. San Diego,CA 92123.

FIGURE 2.4.1-2

P1 CONNECTOR SIGNAL LAYOUT



(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	$\overline{INT}$	Interrupt Request (Active Low)
14B	A15B	Address Bus Bit 15, Buffered
15A	$\overline{NMI}$	Non-Maskable Int.(Active Low)
15B	A14B	Address Bus Bit 14, Buffered
16A	$\overline{HALT}$	CPU HALT Indicator (Active Low)
16B	A13B	Address Bus Bit 13, Buffered
17A	$\overline{MREQB}$	Memory Request (Active Low),Bfrd.
17B	A12	Address Bus Bit 12
18A	$\overline{IORQB}$	I/O Request (Active Low), Bfrd.
18B	A11	Address Bus Bit 11
19A	$\overline{RDB}$	Read (Active Low), Buffered
19B	A10	Address Bus Bit 10
20A	$\overline{WRB}$	Write (Active Low), Buffered
20B	A9	Address Bus Bit 9
21A	$\overline{BUSAK}$	Bus Acknowledge (Active Low)
21B	A8	Address Bus Bit 8
22A	$\overline{WAIT}$	CPU WAIT (Active Low)
22B	A7	Address Bus Bit 7
23A	$\overline{BUSRQ}$	Bus Request (Active Low)
23B	A6	Address Bus Bit 6
24A	$\overline{RESET}$	CPU Reset (Active Low)
24B	A5	Address Bus Bit 5
25A	$\overline{M1}$	CPU M1 State (Active Low)
25B	A4	Address Bus Bit 4
26A	$\overline{RFSHB}$	Refresh (Active Low),Buffered
26B	DZOUT	Daisy Out (Not Connected)
27A	$\overline{EXROM}$	Extension ROM Enable (Active Low)
27B	R	Color Signal - Red
28A	$\overline{ROSCS}$	ROS Chip Select (Active Low) (Dock Bank Enable)
28B	G	Color Signal - Green
29A	$\overline{BE}$	Bank Enable (Active Low)
29B	B	Color Signal - Blue
30A	IOA5	
30B	BUSISO	
31A	SOUND	Analog Sound Signal Output(0-5V)
31B	VIDEO	Composite Video Signal Output
32A	GND	Signal Ground
32B	GND	Signal Ground

NOTE: All A Pins are on component side of board
All B Pins are on non-component (soldering) side of board

TABLE -2.4.1-2

P1 SIGNAL ELECTRICAL CHARACTERISTICS

	OUTPUTS FROM TS2068				INPUTS TO TS2068			
MNEMONIC	CAPACITIVE LOADING MAX (PF)	V(OL) MAX VOLTS	I(LOAD) MAX (MA)	V(OH) MIN VOLTS	V(IL) MAX VOLTS	V(IH) MIN VOLTS	I IN (MAX) 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
$\overline{\text{RDB}}$	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
$\overline{\text{BE}}$	--	---	---	---	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
D1	30	0.4	1.8	2.4	0.8	2.0	20	120
D2	30	0.4	1.8	2.4	0.8	2.0	20	120
D3	30	0.4	1.8	2.4	0.8	2.0	20	120
D4	30	0.4	1.8	2.4	0.8	2.0	20	120
D5	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
D7	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	----	--
$\overline{\text{SOUND}}$	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.

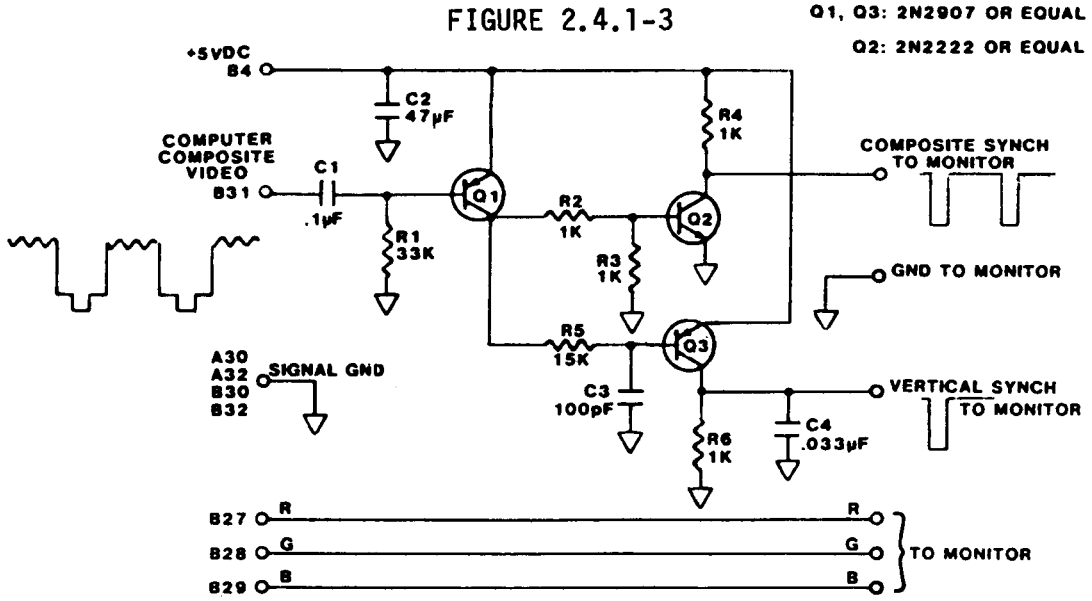
The Schematic of Figure 2.4.1-3 shows the required connections and electronics. Attachment is via the 64-pin keyed P1 connector. Shielding should not normally be required, but ferrite beads are recommended on each wire to minimize EMI, TVI, etc.

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

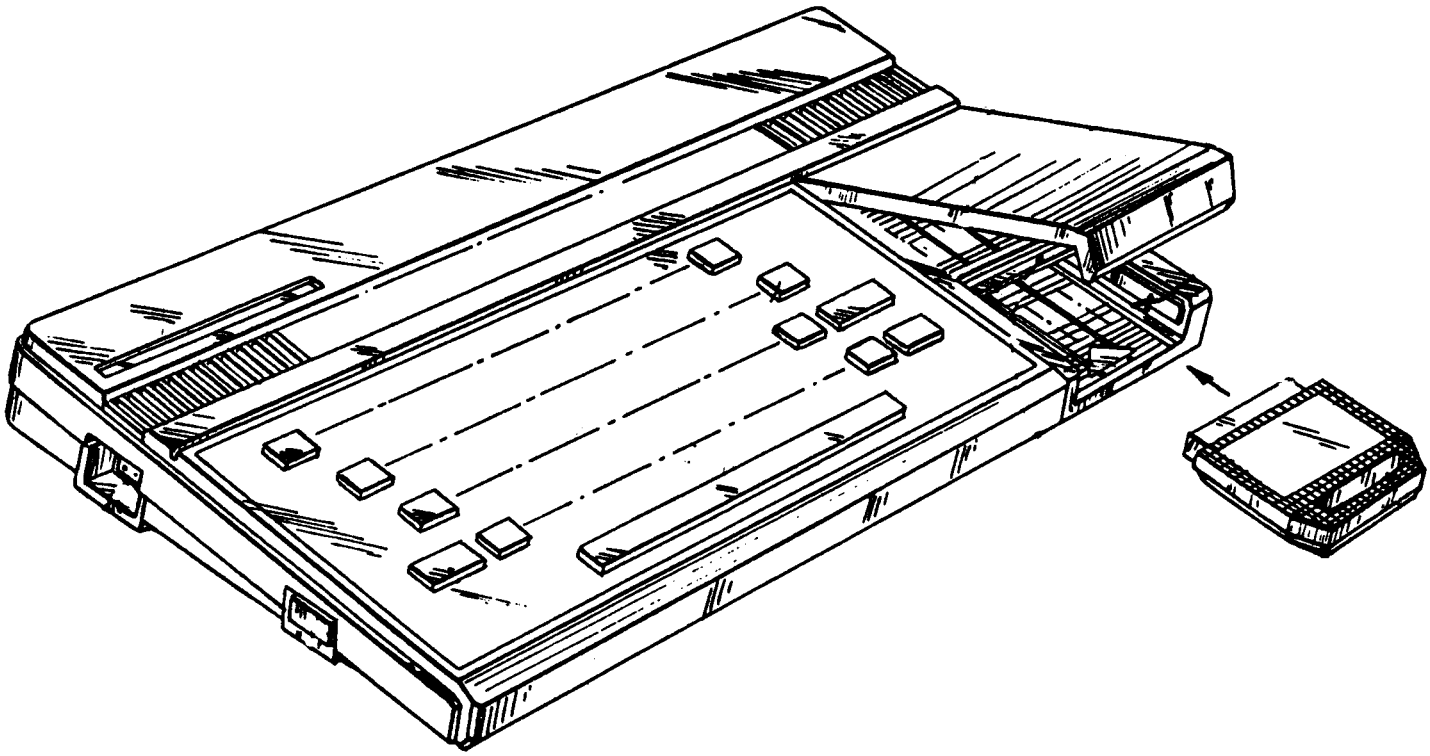
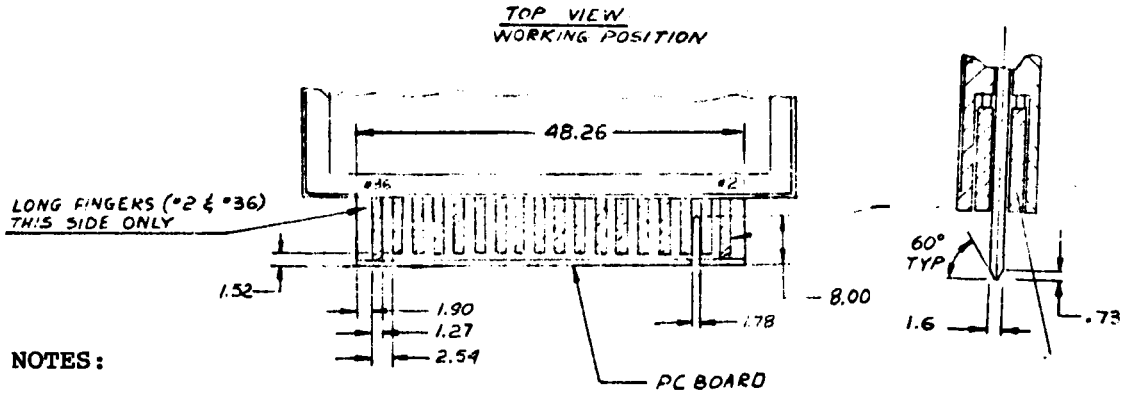


FIGURE 2.4.2-1

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.

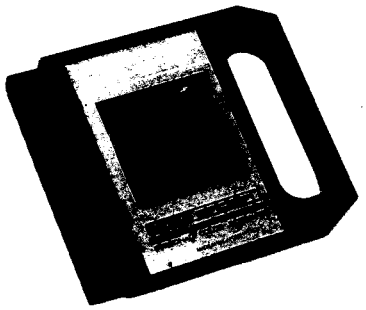


FIGURE 2.4.2-2

J4 SIGNAL LAYOUT

(View from Front)

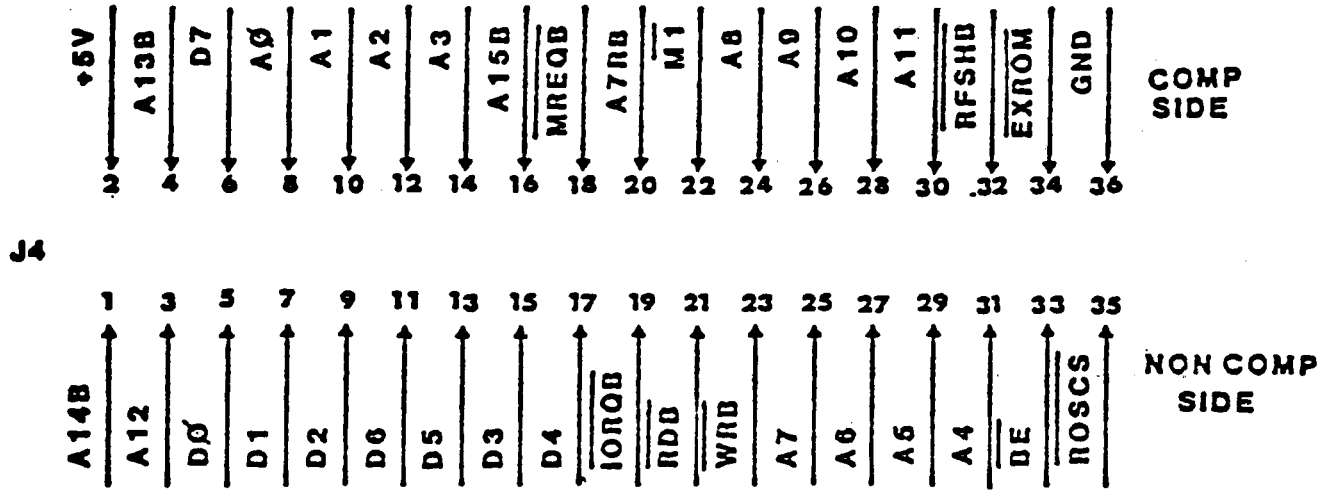


TABLE 2.4.2-1

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)
A15B	30	0.5	1.8	2.4	10				
A14B	30	0.5	1.8	2.4	10				
A13B	30	0.5	1.8	2.4	10				
A12	30	0.4	1.8	2.4	10				
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				
$\overline{A7RB}$	30	0.5	0.35	2.7					
$\overline{ROSCS}$	30	0.4	1.8	2.4	10				
$\overline{MREQB}$	30	0.5	1.8	2.4	10				
$\overline{RDB}$	30	0.5	1.8	2.4	10				
$\overline{IORQB}$	30	0.5	12	2.4	10				
$\overline{WRB}$	30	0.5	12	2.4	10				
$\overline{RFSHB}$	30	0.5	12	2.4	10				
$\overline{EXROM}$	30	0.5	12	2.4	10				
$\overline{MI}$	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.

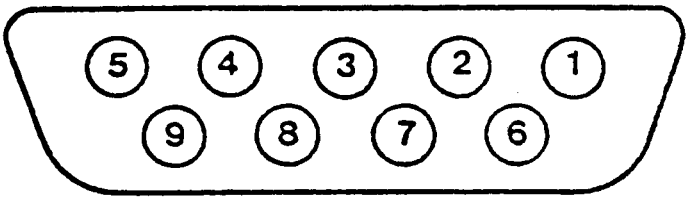
2.4.4 Joystick

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



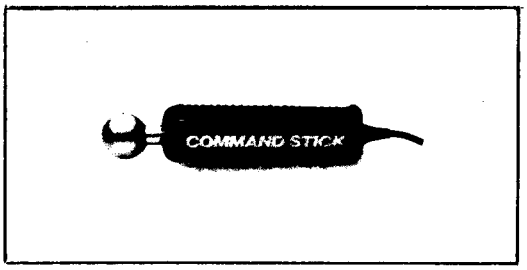


TABLE 2.4.4-1

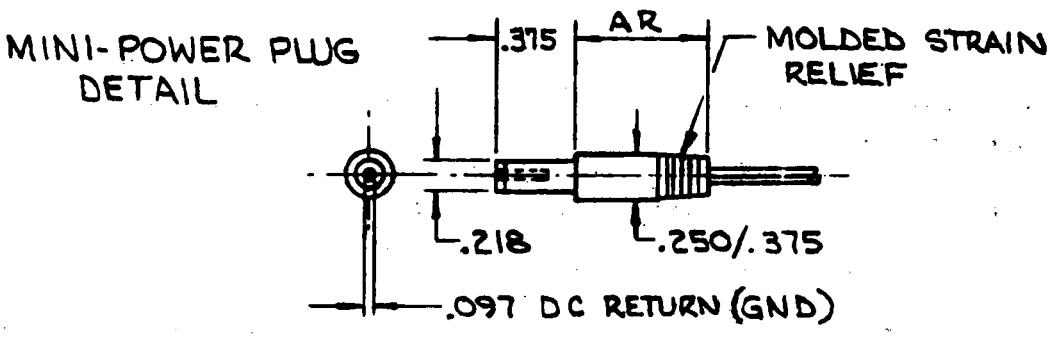
JOYSTICK CONNECTOR SIGNAL ASSIGNMENT

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

```
                                1   ;                                       ; EXTENSION ROM INTERFACE ROUTINE
        =00FC                   2   R_TAPE          EQU     00FCH       ; READ TAPE ROUTINE
        =0068                   3   W_TAPE          EQU     0068H       ; WRITE TAPE ROUTINE
        =0E8E                   4   CHNG_VID        EQU     0E8EH       ; CHANGE VIDEO MODE ROUTINE
        =5CC2                   5   VIDMOD          EQU     5CC2H       ; VIDEO MODE SYSTEM VARIABLE
                                6   ;
                                7   ;
                                8   ;                                       ; CALL READTP WITH REGISTERS SET
                                9   ;                                       ; UP FOR R_TAPE ROUTINE
                               10   ;
0000' 21 00FC                  11   READTP  LD      HL,R_TAPE           ; ADDRESS TO HL
0003' CD 002D'                 12           CALL    IFRTN               ; ENABLE EXT./EXECUTE RTN
0006' 18 17                    13           JR      EXIT                ; RESTORE HOME BANK AND RETURN
                               14   ;
                               15   ;                                       ; CALL WRITTP WITH REGISTERS SET
                               16   ;                                       ; UP FOR W_TAPE ROUTINE
                               17   ;
0008' 21 0068                  18   WRITTP  LD      HL,W_TAPE           ; ADDRESS TO HL
000B' CD 002D'                 19           CALL    IFRTN               ;
000E' 18 0F                    20           JR      EXIT
                               21   ;
                               22   ;                                       ; CALL CHGVID WITH DESIRED VIDEO
                               23   ;                                       ; MODE IN A
                               24   ;
0010' 21 0E8E                  25   CHGVID  LD      HL,CHNG_VID         ; ADDRESS TO HL
0013' F5                       26           PUSH    AF                  ; SAVE VIDEO MODE
0014' CD 002D'                 27           CALL    IFRTN
                               28   ;
                               29   ;                                       ; COMPENSATE FOR 'BUG' IN
                               30   ;                                       ; CHNG_VID RTN.WHICH SETS
                               31   ;                                       ; VIDMOD=0 INSTEAD OF 80
                               32   ;                                       ; WHEN BOTH DISPLAY FILES
                               33   ;                                       ; ARE OPEN
                               34   ;
                               35   ;
0017' F1                       36           POP     AF                  ; TEST VIDEO MODE
0018' FE 80                    37           CP      80H                 ; TEST IF 80
001A' 20 03                    38           JR      NZ,EXIT
001C' 32 5CC2                  39           LD      (VIDMOD),A          ; SET VIDMOD=80H
001F' 3A 002C'                 40   EXIT    LD      A,(HSSAVE)          ; GET PREV. HOR.SEL.
0022' D3 F4                    41           OUT     (0F4H),A            ; RESTORE
0024' DB FF                    42           IN      A,(0FFH)            ; READ PORT FF
0026' CB BF                    43           RES     7,A                 ; TURN OFF ROM SEL.
0028' D3 FF                    44           OUT     (0FFH),A
002A' FB                       45           EI
002B' C9                       46           RET
                               47   ;
002C' 00                       48   HSSAVE  DEFB    0                   ; SAVE HOR.SEL. (PORT 0F4H)
                               49   ;
002D' F3                       53   IFRTN   DI                          ; MASK INTERRUPTIONS
002E' F5                       54           PUSH    AF                  ; PRESERVE REG. A
002F' DB FF                    55           IN      A,(0FFH)            ; EXT.ROM SELECT BIT
0031' CB FF                    56           SET     7,A                 ; SEL. EXT.ROM
0033' D3 FF                    57           OUT     (0FFH),A
0035' DB F4                    58           IN      A,(0F4H)            ; HORIZONTAL SELECT FOR DOCK/EXT
0037' 32 002C'                 59           LD      (HSSAVE),A          ; SAVE
003A' 3E 01                    60           LD      A,1                 ; SELECT CHUNK 0 IN EXT.ROM
003C' D3 F4                    61           OUT     (0F4H),A            ;
003E' F1                       62           POP     AF                  ; RESTORE REG. A
003F' E9                       63           JP      (HL)                ; EXECUTE TARGET ROUTINE AND
                               64   ;                                       ; RETURN TO CALLER OF IFRTN
                               65           END
```

3.3 RAM Organization and Services

3.3.1 System Variables

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

3.3.2 System Configuration Table

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

3.3.3 Machine Stack

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

Note that the Machine Stack is located in the same memory area as the second display file. The CHNG_VID routine relocates the stack to the memory area from OF7C0H to OF8BFH, and modifies the Stack Pointer and MSTBOT (OF8C0H), 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: (0=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

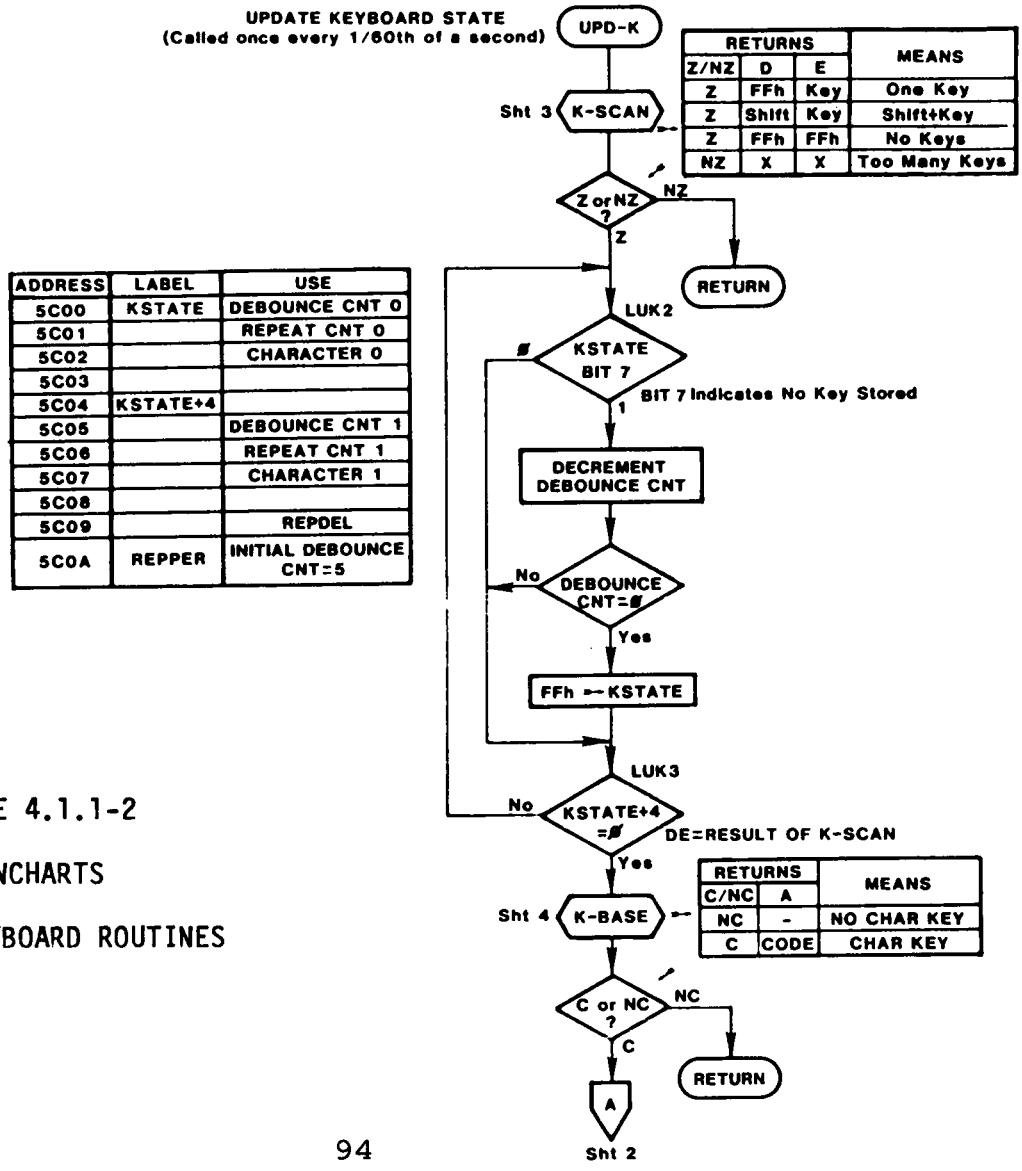
The low-level routines supporting keyboard input are executed every 1/60 of a second out of the Interruption Handler (Location 56 (38H)). The controlling routine is labelled UPD_K. This routine calls K_SCAN to determine if any key(s) are currently being depressed, controls the debouncing and repeat algorithms, 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

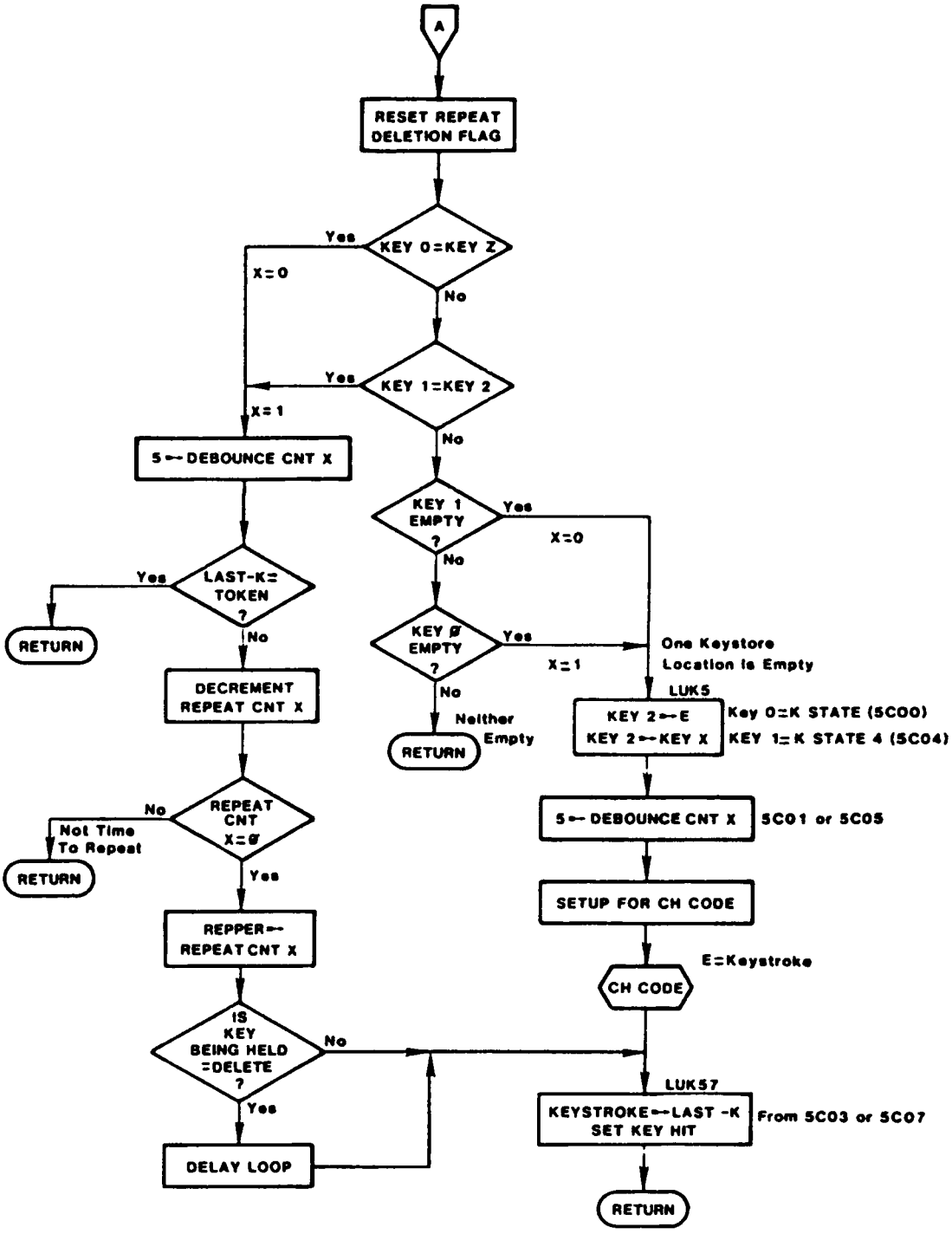
ADDRESS	LABEL	USE
5C00	KSTATE	DEBOUNCE CNT 0
5C01		REPEAT CNT 0
5C02		CHARACTER 0
5C03		
5C04	KSTATE+4	
5C05		DEBOUNCE CNT 1
5C06		REPEAT CNT 1
5C07		CHARACTER 1
5C08		
5C09		REPDEL
5C0A	REPPER	INITIAL DEBOUNCE CNT=5

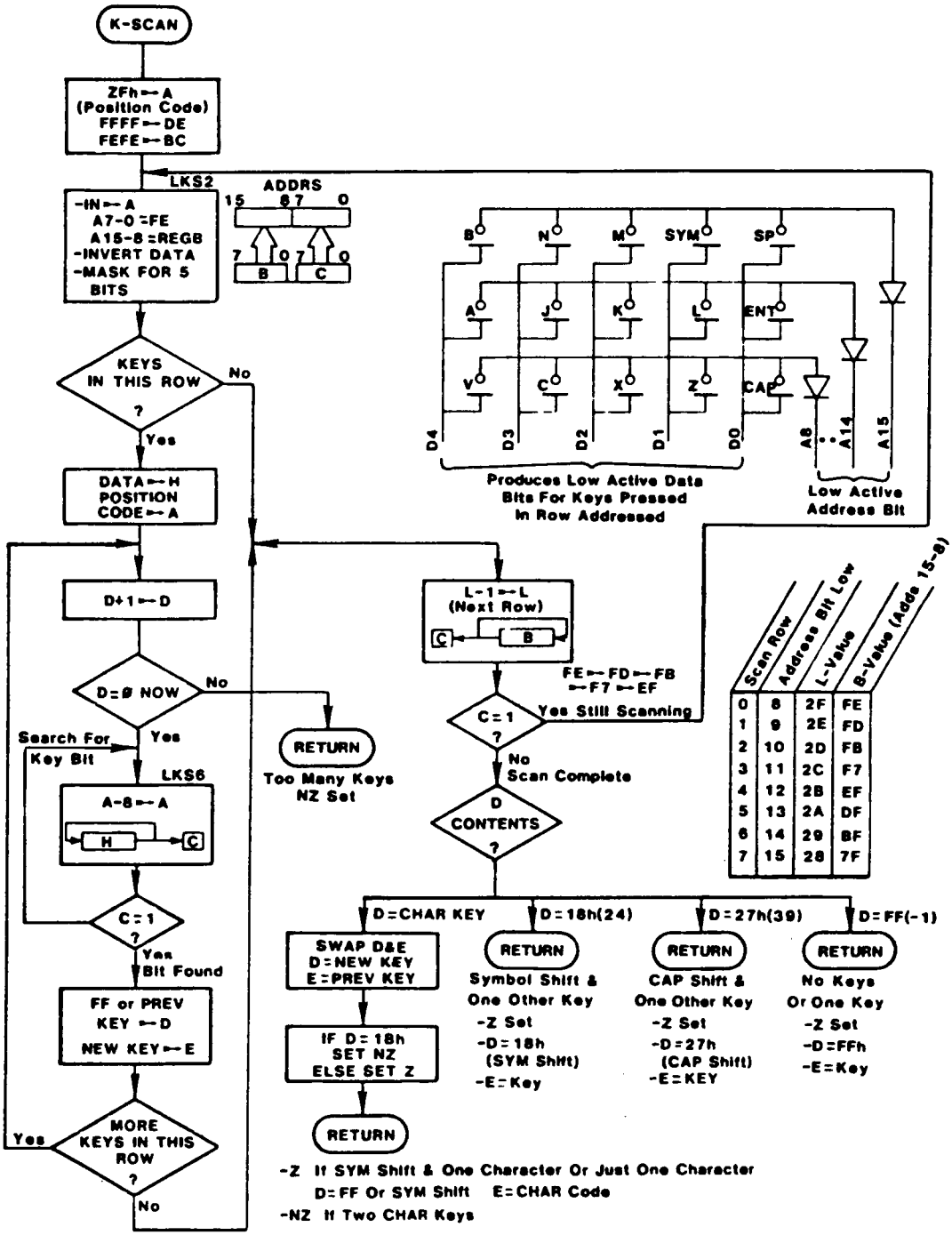
RETURNS		MEANS
C/NC	A	
NC	-	NO CHAR KEY
C	CODE	CHAR KEY

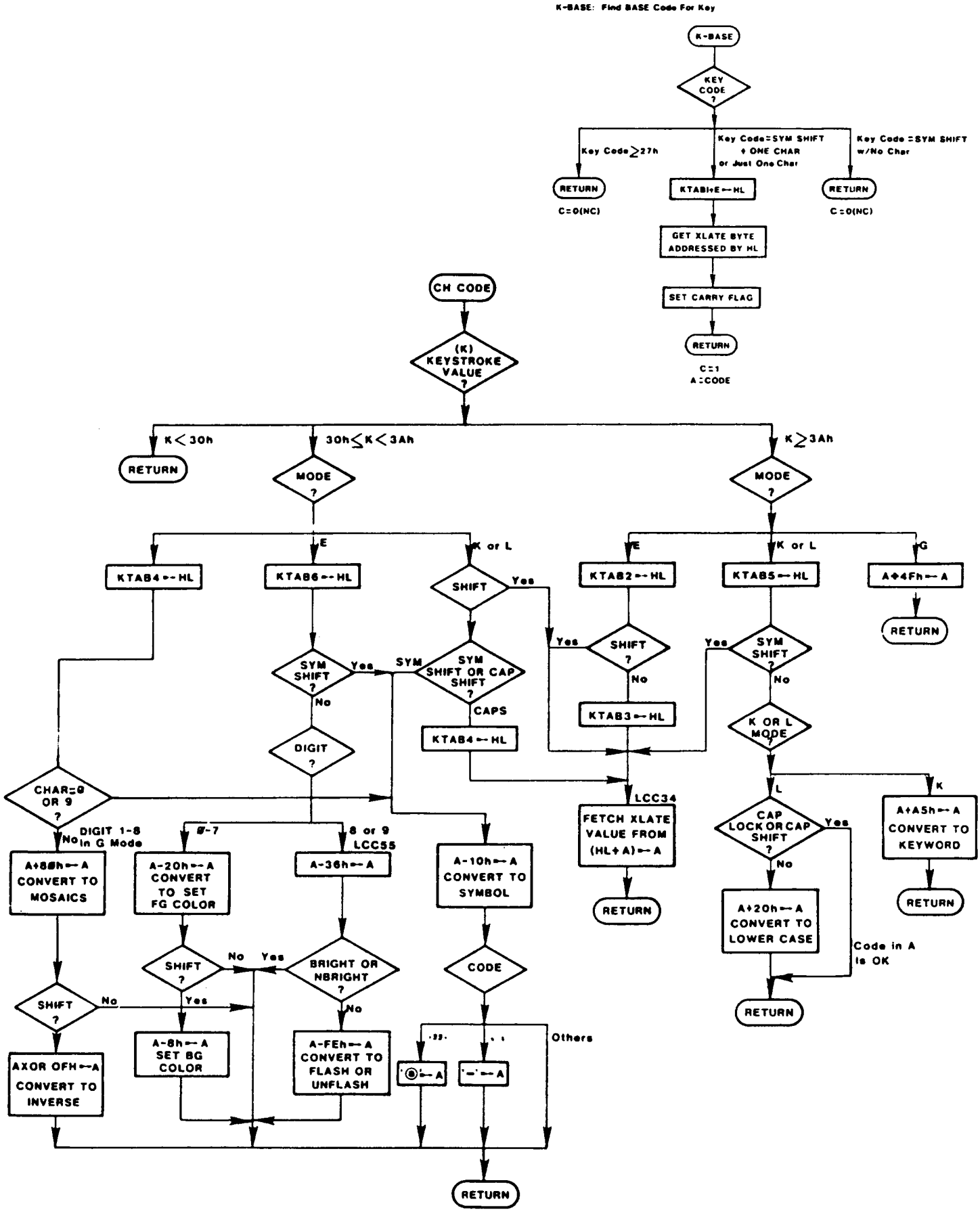
FIGURE 4.1.1-2

FLOWCHARTS

TS 2068 KEYBOARD ROUTINES


A
RESET REPEAT DELETION FLAG
KEY 0=KEY Z
Yes
X=0
No
KEY 1=KEY 2
Yes
X=1
No
5 ← DEBOUNCE CNT X
KEY 1 EMPTY ?
Yes
X=0
No
LAST-K= TOKEN ?
Yes
RETURN
No
KEY Ø EMPTY ?
Yes
X=1
One Keystore Location is Empty
No
Neither Empty
RETURN
DECREMENT REPEAT CNT X
LUK5
KEY 2 ← E
KEY 2 ← KEY X
Key 0=K STATE (5C00)
KEY 1=K STATE 4 (5C04)
REPEAT CNT X=Ø
No
Not Time To Repeat
RETURN
Yes
5 ← DEBOUNCE CNT X
5C01 or 5C05
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-8 ← A
H C
C=1
?
Yes
Bit Found
FF or PREV
KEY ← D
NEW KEY ← E
MORE
KEYS IN THIS
ROW
?
Yes
No
RETURN
Too Many Keys
NZ Set
L-1 ← L
(Next Row)
C B
FE ← FD ← FB
← F7 ← EF
C=1
?
Yes Still Scanning
No
Scan Complete
D
CONTENTS
?
B N M SYM SP
A J K L ENT
V C X Z CAP
D4 D3 D2 D1 D0
A8 A14 A15
Produces Low Active Data
Bits For Keys Pressed
In Row Addressed
Low Active
Address Bit
Scan Row
Address Bit Low
L-Value
B-Value (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
IF 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 Just One Character
D=FF Or SYM Shift E=CHAR Code
-NZ If Two CHAR Keys


K-BASE: Find BASE Code For Key
K-BASE
KEY CODE ?
Key Code≥27h
Key Code=SYM SHIFT + ONE CHAR or Just One Char
Key Code =SYM SHIFT w/No Char
RETURN
C=0(NC)
KTABLE←HL
RETURN
C=0(NC)
GET XLATE BYTE ADDRESSED BY HL
SET CARRY FLAG
RETURN
C=1
A=CODE
CH CODE
(K) KEYSTROKE VALUE ?
K<30h
30h≤K<3Ah
K≥3Ah
RETURN
MODE ?
MODE ?
E
K or L
E
K or L
G
KTAB4←HL
KTAB6←HL
SHIFT
Yes
KTAB2←HL
KTAB5←HL
A←4Fh←A
RETURN
SYM SHIFT ?
Yes
SYM
SYM SHIFT OR CAP SHIFT ?
Yes
SHIFT ?
Yes
SYM SHIFT ?
No
No
CAPS
No
KTAB3←HL
K OR L MODE ?
KTAB4←HL
DIGIT ?
CHAR=0 OR 9 ?
LCC34
L
K
No
DIGIT 1-8 in G Mode
8-7
8 or 9
LCC55
FETCH XLATE VALUE FROM (HL+A)←A
CAP LOCK OR CAP SHIFT ?
Yes
A+A5h←A CONVERT TO KEYWORD
A+80h←A CONVERT TO MOSAICS
A-20h←A CONVERT TO SET FG COLOR
A-36h←A
A-10h←A CONVERT TO SYMBOL
RETURN
No
A+20h←A CONVERT TO LOWER CASE
RETURN
RETURN
Code in A is OK
SHIFT ?
No
Yes
BRIGHT OR NBRIGHT ?
CODE
SHIFT ?
No
Yes
No
Yes
AXOR 0FH←A CONVERT TO INVERSE
A-8h←A SET BG COLOR
A-FEh←A CONVERT TO FLASH OR UNFLASH
'@'←A
'-'←A
Others
RETURN

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.

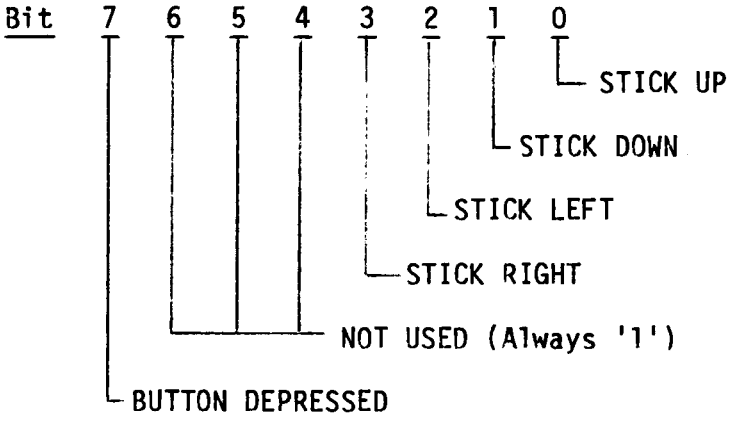
Sample routine:

```
GETJOY  LD    A,0EH             Load A = 14
        OUT   A,(0F5H)          Address the joystick port
        LD    B,playerno
        LD    C,0F5H            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

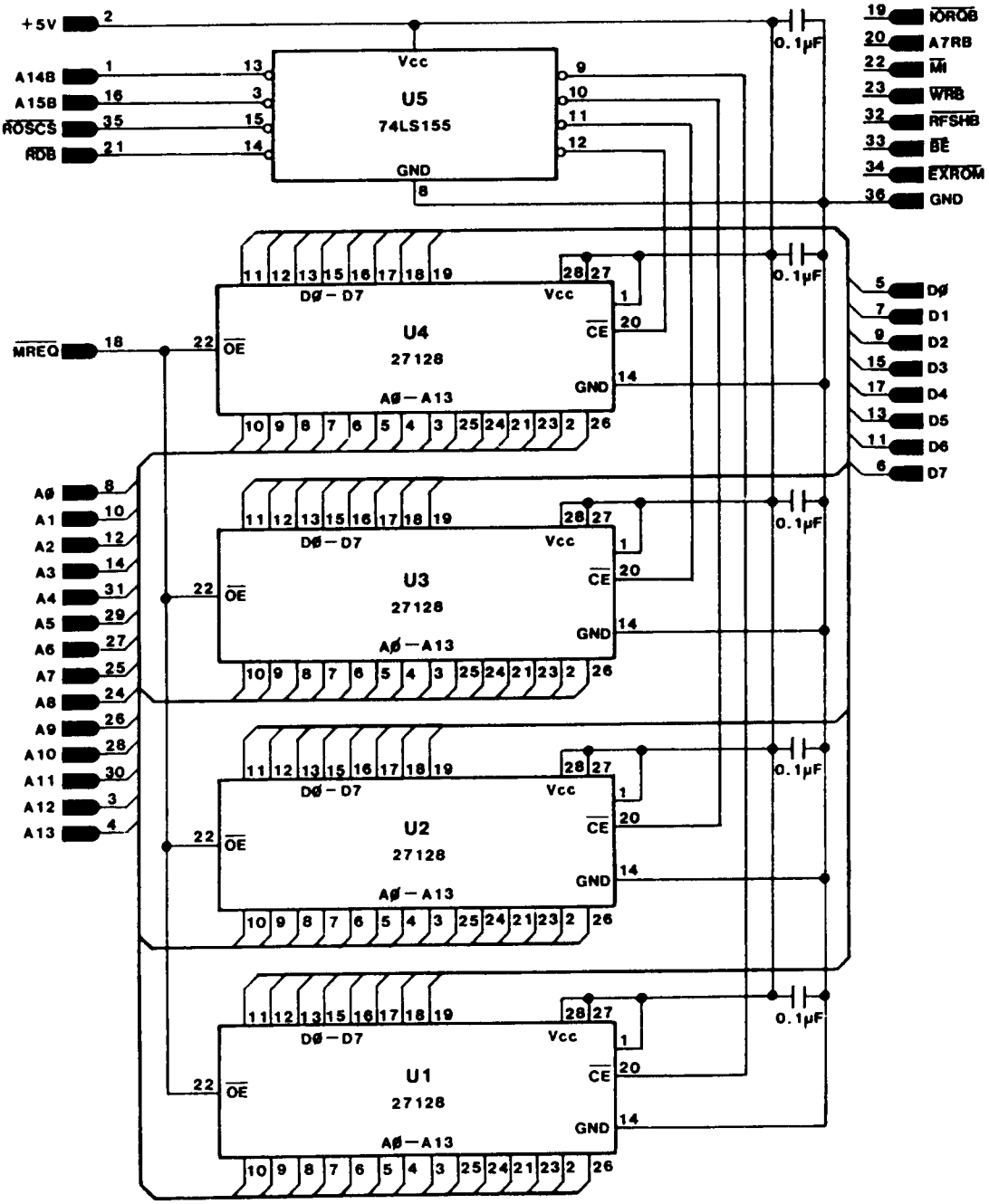
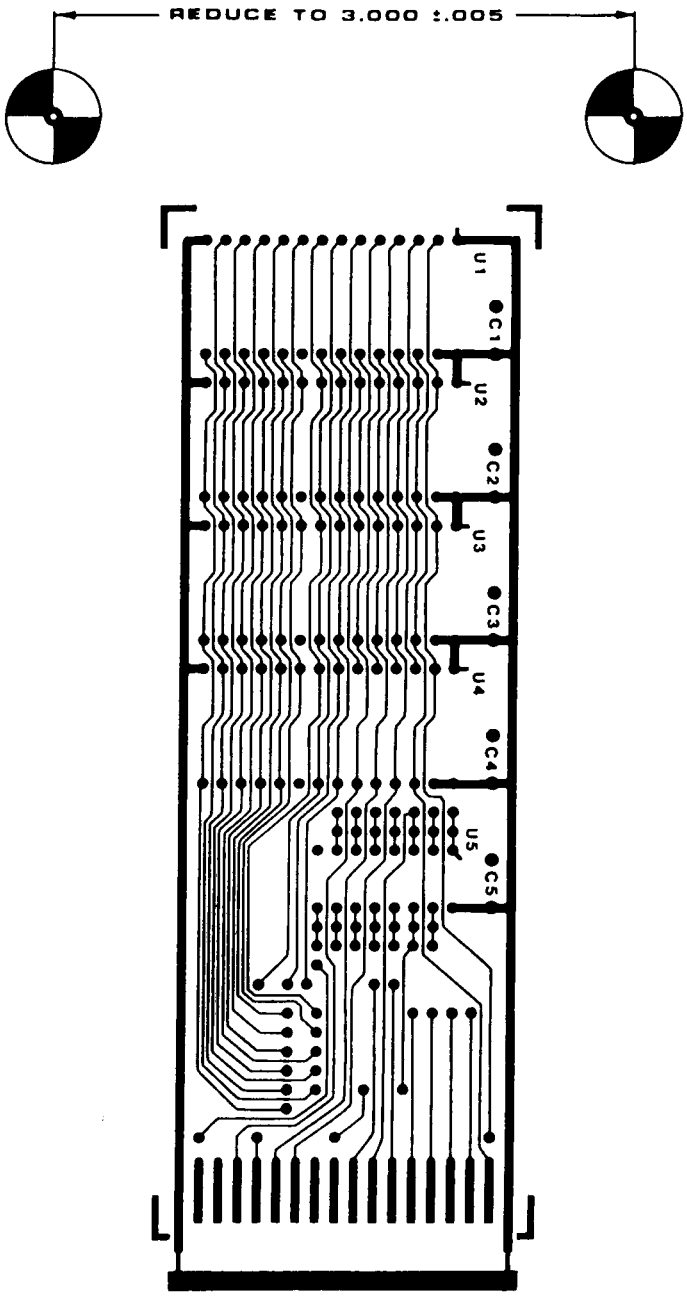


FIGURE 5.1-2

EPROM CARTRIDGE BOARD

COMPONENT SIDE ARTWORK



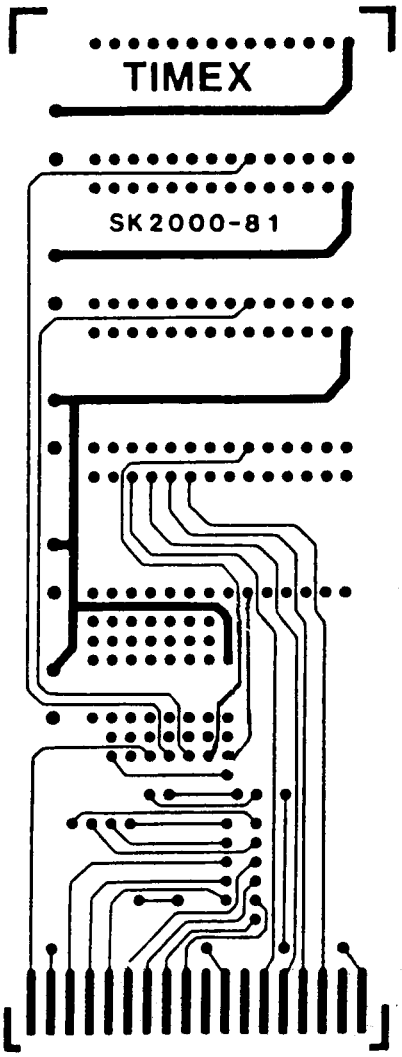
COMPONENT SIDE

SK2000-81

FIGURE 5.1-3

EPROM CARTRIDGE BOARD

SOLDER SIDE ARTWORK



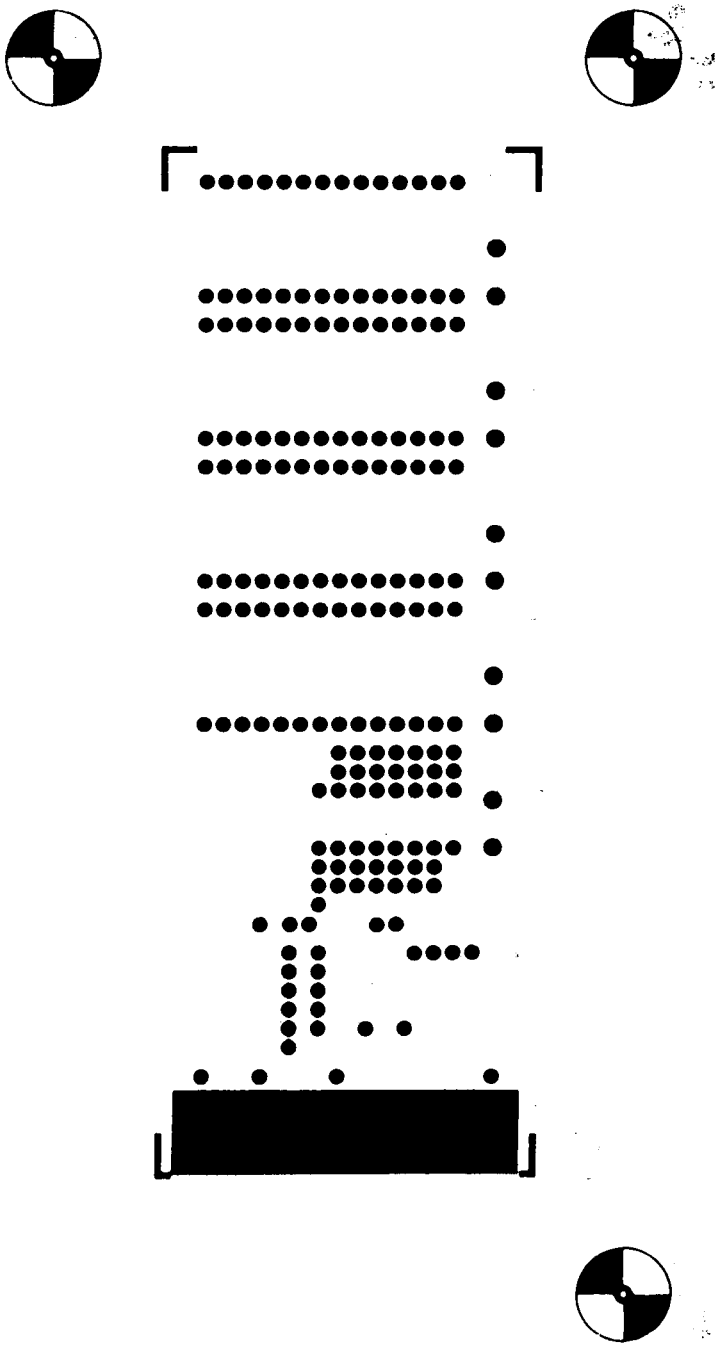
SOLDER SIDE

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.

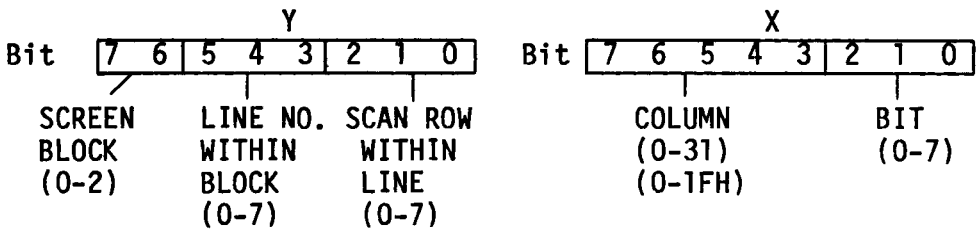
The two inputs are assumed to be as follows:

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        288E   EXPRN
GETAL         17CF   AROS
GET_EL        2D54   IDENT
GET_LN        1324   EDIT
GET_XY        2660   GRAPHS
GO_SUB,       1F99   SYNTAX
GR_COL        238C   SYNTWO
HIFLSH        241D   SYNTWO
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      1EDC     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_SYSVAR2.S
                  423  ;  HERE ARE SOME NEW SYSTEM VARIABLE DEFINITIONS
                  424  ;
                  425  ;
                  426  STKSZ           EQU     200H
                  427  SADDPT          EQU     0F5H    ;SOUND CHIP ADDR PORT
                  428  SDATPT          EQU     0F6H    ;SOUND CHIP DATA PORT
                  429  HS              EQU     40H
                  430  HS_LSN          EQU     HS
                  431  BNA             EQU     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     1FFFH
                  462  LAST_EXT_SVC    EQU     13
                  463  LAST_RAM_SVC    EQU     24
                  464  ;
                  465  ;
                  466  ;
6200              467                  ORG     6200H
                  468
6200  DD210000    469  DISPATCH        LD      IX, 0
6204  DD39        470                  ADD     IX, SP          ;IX = SP
6206  C5          471                  PUSH    BC              ;RESERVE A WORD ON THE STACK
6207  F5          472                  PUSH    AF              ;SAVE REGS
6208  C5          473                  PUSH    BC
6209  D5          474                  PUSH    DE
620A  E5          475                  PUSH    HL
620B  DD5E02      476                  LD      E, (IX+2)
620E  DD5603      477                  LD      D, (IX+3)       ;DE = SVC_CODE
6211  AF          478                  XOR     A
6212  CB23        479                  SLA     E
6214  CB12        480                  RL      D               ;DE = 2*DE
6216  17          481                  RLA                     ;A = JUMP FLAG
6217  210D00      482                  LD      HL, LAST_EXT_SVC
621A  CB25        483                  SLA     L
621C  CB14        484                  RL      H               ;HL = 2*HL
621E  A7          485                  AND     A
621F  ED52        486                  SBC     HL, DE          ;COMPARE HL AND DE
6221  3015        487                  JR      NC, D_EXT       ;IF DE <= HL
6223  211800      488                  LD      HL, LAST_RAM_SVC
6226  CB25        489                  SLA     L
6228  CB14        490                  RL      H
622A  A7          491                  AND     A
622B  ED52        492                  SBC     HL, DE
622D  380F        493                  JR      C, D_HOME
622F  06FF        494                  LD      B, 255          ;HERE FOR RAM-BASED SERVICES
6231  CD0564      495                  CALL    GET_STATUS      ;GET STATUS OF HOME BANK
6234  06FF        496                  LD      B, 255          ;BC = HOME BANK / HORIZ-SELECT
6236  180A        497                  JR      D_SAVE
6238  06FE        498  D_EXT           LD      B, 254          ;HERE FOR EXT. ROM BASED SERVICES
623A  0EFE        499                  LD      C, 0FEH
623C  1804        500                  JR      D_SAVE
623E  06FF        501  D_HOME          LD      B, 255          ;SET BANK_ENABLE PARMS FOR HOME
6240  0E00        502                  LD      C, 0
6242  F5          503  D_SAVE          PUSH    AF
6243  C5          504                  PUSH    BC              ;SAVE JUMP FLAG AND BANK_ENABLE PARMS
6244  21FF1F      505                  LD      HL, JMPTBL      ;CALC. ADDR OF TABLE ENTRY
6247  37          506                  SCF
6248  ED52        507                  SBC     HL, DE
624A  06FE        508                  LD      B, 254
624C  CD1663      509                  CALL    GET_WORD        ;READ TABLE ENTRY
624F  EB          510                  EX      DE, HL
6250  C1          511                  POP     BC
6251  F1          512                  POP     AF              ;RESTORE JUMP FLAG, ETC.
6252  A7          513                  AND     A
6253  281F        514                  JR      Z, D_CALL
6255  DD71FE      515                  LD      (IX-2), C       ;PUT BANK# AND HOR-SEL ON STACK
6258  DD70FF      516                  LD      (IX-1), B
625B  DD6E00      517                  LD      L, (IX)         ;SAVE RET ADDR
625E  DD6601      518                  LD      H, (IX+1)
6261  DD7403      519                  LD      (IX+3), H       ;PUT RET ADDR BACK ON STACK
```

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

```
                        619
6311    E1              620 LNI3:   POP HL
6312    F1              621         POP AF
6313    ED45            622         RETN
                        623 ;
                        624 ;
6315                    625 BS_MAX_BANK     DEFS    1       ;THIS IS A COPY OF MAX_BANK
                        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
```

```
638E   CB2F      720                SRA     A
6390   CB2F      721                SRA     A
6392   CB2F      722                SRA     A
6394   77        723                LD      (HL), A          ;WRITE MSN OF DATA
6395   3E07      724                LD      A, 7             ;RESTORE SOUND REGS
6397   D3F5      725                OUT     (SADDPT), A
6399   78        726                LD      A, B
639A   D3F6      727                OUT     (SDATPT), A
639C   3E0E      728                LD      A, 0EH
639E   D3F5      729                OUT     (SADDPT), A
63A0   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 (LEN_ADDR: D; NON_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_DOOR       ;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
```

```
641F  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   GS_HOME         LD      BC, 0           ;RETURN 0 FOR HOME BANK STATUS
6430  1818        830                   JR      GS_EXIT
6432  DBF4        831   GS_DOCK         IN      A, (DKHSPT)     ;RETURN DOCK BANK STATUS
6434  2F          832                   CPL
6435  47          833                   LD      B, A
6436  0E00        834                   LD      C, 0
6438  1810        835                   JR      GS_EXIT
643A  DBFF        836   GS_EXT          IN      A, (HREXPT)
643C  E680        837                   AND     80H             ;CLEAR ALL BITS EXCEPT BIT 7
643E  2F          838                   CPL
643F  07          839                   RLCA                    ;PUT ACTIVE LOW BIT IN BIT ZERO
6440  47          840                   LD      B, A
6441  DBF4        841                   IN      A, (DKHSPT)
6443  2F          842                   CPL
6444  E601        843                   AND     1
6446  B0          844                   OR      B
6447  47          845                   LD      B, A
6448  0E00        846                   LD      C, 0
644A  D1          847   GS_EXIT         POP     DE              ;RESTORE D, C
644B  F1          848                   POP     AF
644C  C9          849                   RET
                  850   ;
                  851   ;
                  852   ;  GET_CHUNK (ADDR: HL; MASK: A)
                  853   ;
                  854   ;
644D  C5          855   GET_CHUNK       PUSH    BC              ;SAVE B
644E  7C          856                   LD      A, H            ;CHUNK NUMBER = HIGH 3 BITS OF
644F  0605        857                   LD      B, 5            ; H SO SHIFT H RIGHT 5 BITS
6451  CB3F        858   GC_SHIFT        SRL     A
6453  10FC        859                   DJNZ    GC_SHIFT
6455  3C          860                   INC     A               ;CREATE MASK BY ROLLING A 1
6456  47          861                   LD      B, A            ; LEFT CHUNK NUMBER+1 TIMES.
6457  AF          862                   XOR     A               ; THE 1 COMES FROM THE CARRY
6458  37          863                   SCF                     ; FLAG
6459  17          864   GC_ROLL         RLA
645A  10FD        865                   DJNZ    GC_ROLL
645C  C1          866                   POP     BC              ;RESTORE B
645D  C9          867                   RET
                  868   ;
                  869   ;
                  870   ;  GET_BANK_NUMBER (ADDR: HL; BANK_NUMBER: A)
                  871   ;
                  872   ;
645E  C5          873   GET_NUMBER      PUSH    BC              ;SAVE REGS
645F  D5          874                   PUSH    DE
6460  CD4D64      875                   CALL    GET_CHUNK
6463  4F          876                   LD      C, A
6464  3A1563      877                   LD      A, (BS_MAX_BANK);GET LARGEST BANK NUMBER
6467  A7          878                   AND     A
6468  280A        879                   JR      Z, 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      OFFH
64BC  2806      939                 JR      Z, BE_EXT_OK
64BE  DBFF      940                 IN      A, (HREXPT)     !HERE FOR DOCK
64C0  CBBF      941                 RES     7, A
64C2  D3FF      942                 OUT     (HREXPT), A
64C4  79        943 BE_EXT_OK       LD      A, C
64C5  2F        944                 CPL
64C6  D3F4      945                 OUT     (DKHSPT), A     !ENABLE DOCK
64C8  184F      946                 JR      BE_EXIT
64CA  79        947 BE_NTDOCK       LD      A, B            !CHECK IF EXT.
64CB  FEFE      948                 CP      OFEH
64CD  201D      949                 JR      NZ, BE_NTEXT
64CF  DBFF      950                 IN      A, (HREXPT)     !HERE FOR EXT.
64D1  17        951                 RLA
64D2  CB19      952                 RR      C
64D4  2F        953                 CCF
64D5  1F        954                 RRA
64D6  D3FF      955                 OUT     (HREXPT), A
64D8  CB7F      956                 BIT     7, A
64DA  2009      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     (DKHSFT), 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      OFFH
6509  280E      984                 JR      Z, BE_EXIT      !IS HOME, SO DONE
650B  1680      985                 LD      D, BNA          !WRITE NEW EXP. BANK STATUS
650D  58        986                 LD      E, B
650E  CD5C63    987                 CALL    WRITE_BS_REG
6511  1640      988                 LD      D, HS
6513  79        989                 LD      A, C
6514  2F        990                 CPL
6515  5F        991                 LD      E, A
6516  CD5C63    992                 CALL    WRITE_BS_REG
6519  E1        993 BE_EXIT         POP     HL              !RESTORE REGISTERS
651A  D1        994                 POP     DE
651B  C1        995                 POP     BC
651C  F1        996                 POP     AF
651D  C9        997                 RET
                998 ;
                999 ;
               1000 ;  SAVE_BANK_STATUSES (STATUS_ADDR: IX)
               1001 ;
               1002 ;          PUSHES THE STATUS OF ALL BANKS ON THE STACK
               1003 ;
               1004 ;
651E  F5       1005 SAVE_STATUS    PUSH    AF              !SAVE REGS
651F  C5       1006                PUSH    BC
6520  D5       1007                PUSH    DE
6521  DBFF     1008                IN      A, (HREXPT)     !SAVE EXT. BANK STATUS
6523  00       1009                NOP                     !LEAVE BITS 0-6 ALONE! NOPS PUT IN
6524  00       1010                NOP                     ! TO KEEP ADDRS THE SAME
6525  DD7700   1011                LD      (IX), A
6528  DD23     1012                INC     IX
652A  DBF4     1013                IN      A, (DKHSPT)     !GET DOCK BANK STATUS
652C  DD7700   1014                LD      (IX), A
652F  DD23     1015                INC     IX
6531  3A1563   1016                LD      A, (BS_MAX_BANK)!GET NUMBER OF BANKS
6534  A7       1017                AND     A
6535  280D     1018                JR      Z, SS_EXIT
6537  47       1019                LD      B, A            !SET UP COUNTER
6538  58       1020 SS_LOOP        LD      E, B            !BANK NUMBER INTO E
6539  CD0564   1021                CALL    GET_STATUS      !GET BANK STATUS OF BANK #B
```

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

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

```
66A4  07        1223                  RLCA
66A5  0F        1224                  RRCA
66A6  3805      1225                  JR      C, MB_RV1       ;IF A < 0
66A8  EDB0      1226                  LDIR
66AA  09        1227                  ADD     HL, BC          ;INCREMENT POINTER
66AB  1805      1228                  JR      MB_UP1
66AD  EDB8      1229  MB_RV1          LDDR
66AF  A7        1230                  AND     A
66B0  ED42      1231                  SBC     HL, BC          ;DECREMENT POINTER
66B2  DD7506    1232  MB_UP1          LD      (IX+SRC_ADDR), L  ;STORE NEW POINTER VALUE
66B5  DD7407    1233                  LD      (IX+SRC_ADDR+1), H
66B8  C1        1234                  POP     BC
66B9  E1        1235                  POP     HL
66BA  E5        1236                  PUSH    HL
66BB  C5        1237                  PUSH    BC
66BC  DD4608    1238                  LD      B, (IX+DEST_BANK)
66BF  CD9964    1239                  CALL    BANK_ENABLE     ;SELECT DEST BANK
66C2  44        1240                  LD      B, H            ;MOVE FROM STACK TO DEST
66C3  4D        1241                  LD      C, L
66C4  DD5E04    1242                  LD      E, (IX+DEST_ADDR)
66C7  DD5605    1243                  LD      D, (IX+DEST_ADDR+1)
66CA  DD6E00    1244                  LD      L, (IX+BUF_PTR)
66CD  DD6601    1245                  LD      H, (IX+BUF_PTR+1)
66D0  07        1246                  RLCA
66D1  0F        1247                  RRCA
66D2  3805      1248                  JR      C, MB_RV2       ;IF A < 0
66D4  EDB0      1249                  LDIR
66D6  09        1250                  ADD     HL, BC          ;INCREMENT POINTER
66D7  1805      1251                  JR      MB_UP2
66D9  EDB8      1252  MB_RV2          LDDR
66DB  A7        1253                  AND     A
66DC  ED42      1254                  SBC     HL, BC          ;DECREMENT POINTER
66DE  DD7504    1255  MB_UP2          LD      (IX+DEST_ADDR), L  ;STORE NEW POINTER VALUE
66E1  DD7405    1256                  LD      (IX+DEST_ADDR+1), H
66E4  C1        1257                  POP     BC              ;RESTORE REGISTERS
66E5  D1        1258                  POP     DE
66E6  E1        1259                  POP     HL
66E7  C9        1260                  RET
                1261  ;
                1262  ;
                1263  ;  CREATE_BITMAP (ADDR: HL; BITMAP: A)
                1264  ;
                1265  ;
66E8  54        1266  CREATE_BITMAP   LD      D, H            ;SAVE START ADDR
66E9  5D        1267                  LD      E, L
66EA  DD4E02    1268                  LD      C, (IX+LENGTH)
66ED  DD4603    1269                  LD      B, (IX+LENGTH+1)
66F0  DD7E00    1270                  LD      A, (IX+DIRECTION)
66F3  07        1271                  RLCA                    ;CALCULATE END ADDR
66F4  0F        1272                  RRCA
66F5  3803      1273                  JR      C, CB_SUB       ;IF A < 0
66F7  09        1274                  ADD     HL, BC
66F8  1802      1275                  JR      CB_CONT
66FA  ED42      1276  CB_SUB          SBC     HL, BC
66FC  CD4D64    1277  CB_CONT         CALL    GET_CHUNK       ;GET END CHUNK BIT
66FF  2F        1278                  CPL
6700  47        1279                  LD      B, A
6701  EB        1280                  EX      DE, HL
6702  CD4D64    1281                  CALL    GET_CHUNK       ;GET START CHUNK BIT
6705  2F        1282                  CPL
6706  4F        1283                  LD      C, A
6707  A8        1284                  XOR     B
6708  2816      1285                  JR      Z, CB_EXIT
670A  79        1286                  LD      A, C            ;HERE IF START AND END CHUNKS D
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 BANKS' STATUS
6746  D5        1334                PUSH    DE
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  CDE866    1338                CALL    CREATE_BITMAP   ;GET SRC BITMAP
6752  F5        1339                PUSH    AF              ;SAVE ON STACK TEMPORARILY
6753  DD6E04    1340                LD      L, (IX+DEST_ADDR)
6756  DD6605    1341                LD      H, (IX+DEST_ADDR+1)
6759  CDE866    1342                CALL    CREATE_BITMAP   ;GET DEST BITMAP
675C  4F        1343                LD      C, A            ;C = DEST BITMAP
675D  F1        1344                POP     AF
675E  47        1345                LD      B, A            ;B = SRC BITMAP
675F  DD7E09    1346                LD      A, (IX+SRC_BANK)
6762  DD5608    1347                LD      D, (IX+DEST_BANK)
6765  BA        1348                CP      D               ;COMPARE SRC AND DEST BANK NUMBERS
6766  2005      1349                JR      NZ, XB_DIFF_BANKS
6768  78        1350                LD      A, B            ;HERE IF BANK NUMBERS ARE DIFFERENT
6769  A1        1351                AND     C
676A  47        1352                LD      B, A            ;B = UNION OR SRC AND DEST BITMAPS
676B  180B      1353                JR      XB_DO_MOVE
676D  78        1354  XB_DIFF_BANKS LD      A, B            ;CHECK FOR OVERLAP BETWEEN SRC AND
676E  B1        1355                OR      C               ; DEST CHUNKS
676F  FEFF      1356                CP      0FFH
6771  202D      1357                JR      NZ, XB_OVERLAP
6773  58        1358                LD      E, B            ;HERE IF NO OVERLAP
6774  42        1359                LD      B, D
6775  CD9964    1360                CALL    BANK_ENABLE     ;SELECT DEST BANK
6778  DD4609    1361  XB_DO_MOVE    LD      B, (IX+SRC_BANK)
677B  4B        1362                LD      C, E
677C  CD9964    1363                CALL    BANK_ENABLE     ;SELECT SRC BANK
677F  DD6E06    1364                LD      L, (IX+SRC_ADDR)
6782  DD6607    1365                LD      H, (IX+SRC_ADDR+1)
6785  DD5E04    1366                LD      E, (IX+DEST_ADDR)
6788  DD5605    1367                LD      D, (IX+DEST_ADDR+1)
678B  DD4E02    1368                LD      C, (IX+LENGTH)
678E  DD4603    1369                LD      B, (IX+LENGTH+1)
6791  DD7E00    1370                LD      A, (IX+DIRECTION)
6794  07        1371                RLCA
6795  0F        1372                RRCA
6796  3804      1373                JR      C, XB_REVERSE   ;IF A < 0
6798  EDB0      1374                LDIR
679A  1852      1375                JR      XB_EXIT
679C  EDB8      1376  XB_REVERSE    LDDR
679E  184E      1377                JR      XB_EXIT
67A0  21C05C    1378  XB_OVERLAP    LD      HL, MSTBOT
67A3  C5        1379                PUSH    BC
67A4  06FF      1380                LD      B, 255
67A6  CD1663    1381                CALL    GET_WORD
67A9  C1        1382                POP     BC
67AA  110002    1383                LD      DE, STKSZ
67AD  A7        1384                AND     A
67AE  ED52      1385                SBC     HL, DE          ;HL = ADDRESS OF STACK LIMIT
67B0  112000    1386                LD      DE, FREE_BYTES
67B3  19        1387                ADD     HL, DE
67B4  EB        1388                EX      DE, HL          ;DE = SP_NEW
67B5  210000    1389                LD      HL, 0
67B8  39        1390                ADD     HL, SP          ;HL = SP_OLD
67B9  13        1391                INC     DE              ;COMPARE SP_OLD WITH SP_NEW
67BA  A7        1392                AND     A
67BB  ED52      1393                SBC     HL, DE
67BD  3004      1394                JR      NC, XB_SPACE    ;IF SP_OLD - SP_NEW > 0
67BF  3E01      1395                LD      A, 1            ;RETURN ERROR CODE
67C1  182B      1396                JR      XB_EXIT
67C3  1B        1397  XB_SPACE     DEC     DE
67C4  EB        1398                EX      DE, HL
67C5  F9        1399                LD      SP, HL          ;SET SP TO SP_NEW
67C6  13        1400                INC     DE              ;DE = BUF_SZ
67C7  DD7E00    1401                LD      A, (IX+DIRECTION)  ;HL = BUF_PTR
67CA  DD7500    1402                LD      (IX+BUF_PTR), L    ;SAVE BUF_PTR ON STACK
67CD  DD7401    1403                LD      (IX+BUF_PTR+1), H
67D0  DD6E02    1404                LD      L, (IX+LENGTH)
67D3  DD6603    1405                LD      H, (IX+LENGTH+1)
67D6  A7        1406  XB_MOVE_LOOP  AND     A               ;HL = BYTES LEFT TO MOVE
67D7  ED52      1407                SBC     HL, DE          ;DE = BYTES TO MOVE THIS TIME
67D9  3805      1408                JR      C, XB_LAST_MOVE ;IF LESS THAN BUF_SZ BYTES LEFT
67DB  CD8C66    1409                CALL    MOVE_BYTES
67DE  18F6      1410                JR      XB_MOVE_LOOP
67E0  19        1411  XB_LAST_MOVE  ADD     HL, DE
67E1  EB        1412                EX      DE, HL
67E2  CD8C66    1413                CALL    MOVE_BYTES
67E5  EB        1414                EX      DE, HL
67E6  DD6E00    1415                LD      L, (IX+BUF_PTR)
67E9  DD6601    1416                LD      H, (IX+BUF_PTR+1)
67EC  19        1417                ADD     HL, DE          ;HL = BUF_PTR+BUF_SZ
67ED  F9        1418                LD      SP, HL          ;RESTORE STACK POINTER
67EE  AF        1419  XB_EXIT       XOR     A               ;RETURN CODE FOR SUCCESSFUL COMPLETION
67EF  DD210000  1420                LD      IX, 0
67F3  DD39      1421                ADD     IX, SP
67F5  CD4A65    1422                CALL    RESTORE_STATUS  ;RESTORE STATE AND RETURN ZERO CODE
```

```
67F8  DD23      1423                    INC     IX
67FA  DDF9      1424                    LD      SP, IX
67FC  E1        1425                    POP     HL              ;RESTORE REGS
67FD  D1        1426                    POP     DE
67FE  C1        1427                    POP     BC
67FF  F1        1428                    POP     AF
6800  DDE1      1429                    POP     IX              ;CLEAN UP PARMS
6802  DDE3      1430                    EX      (SP), IX
6804  DDE1      1431                    POP     IX
6806  DDE3      1432                    EX      (SP), IX
6808  DDE1      1433                    POP     IX
680A  DDE3      1434                    EX      (SP), IX
680C  DDE1      1435                    POP     IX
680E  DDE3      1436                    EX      (SP), IX
6810  DDE1      1437                    POP     IX
6812  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, 1
6820  D3F4      1451                    OUT     (DKHSPT), A
6822  F1        1452                    POP     AF
6823  E9        1453                    JP      (HL)
                1454                    END
```

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

                    1  DISPATCH         EQU     6200H
                    2  INT              EQU     62AEH
                    3  GET_WORD         EQU     6316H
                    4  PUT_WORD         EQU     633BH
                    5  GET_STATUS       EQU     6405H
                    6  GET_NUMBER       EQU     645EH
                    7  BANK_ENABLE      EQU     6499H
                    8  SAVE_STATUS      EQU     651EH
                    9  RESTORE_STATUS   EQU     654AH
                   10  BS_STACK         EQU     658EH
                   11  BS_SP            EQU     65CEH
                   12  GOTO_BANK        EQU     6572H
                   13  CALL_BANK        EQU     65D0H
                   14  MOVE_BYTES       EQU     668CH
                   15  CREATE_BITMAP    EQU     66E8H
                   16  XFER_BYTES       EQU     6722H
                   17
                   18  ;  HERE IS THE FIXUP TABLE FOR THE VIDEO MODE CHANGER.  IT DEFINES THE
                   19  ;  LOCATIONS IN RAM WHICH MUST BE UPDATED WHEN MOVED FROM CHUNK 3 TO CHUNK 7
                   20  ;  OR V CE-VERSA.  THE ADDRESSES IN THE TABLE ARE DEFINED AS CHUNK 3 ADDRESSES
                   21
                   22
1D00               23                      ORG     1D00H
                   24
                   25
1D00  3262         26  FIXTBL              DEFW    DISPATCH+32H
1D02  4D62         27                      DEFW    DISPATCH+4DH
1D04  7262         28                      DEFW    DISPATCH+72H
1D06  AB62         29                      DEFW    DISPATCH+0ABH
                   30
1D08  B862         31                      DEFW    INT+0AH
1D0A  CD62         32                      DEFW    INT+1FH
1D0C  D362         33                      DEFW    INT+25H
1D0E  DC62         34                      DEFW    INT+2EH
1D10  FB62         35                      DEFW    INT+4DH
                   36
1D12  1A63         37                      DEFW    GET_WORD+4H
1D14  2063         38                      DEFW    GET_WORD+0AH
1D16  2463         39                      DEFW    GET_WORD+0EH
1D18  2A63         40                      DEFW    GET_WORD+14H
1D1A  3563         41                      DEFW    GET_WORD+1FH
                   42
1D1C  3E63         43                      DEFW    PUT_WORD+3H
1D1E  4463         44                      DEFW    PUT_WORD+9H
1D20  4863         45                      DEFW    PUT_WORD+0DH
1D22  4E63         46                      DEFW    PUT_WORD+13H
1D24  5763         47                      DEFW    PUT_WORD+1CH
                   48
1D26  1764         49                      DEFW    GET_STATUS+12H
1D28  1E64         50                      DEFW    GET_STATUS+19H
1D2A  2864         51                      DEFW    GET_STATUS+23H
                   52
1D2C  6164         53                      DEFW    GET_NUMBER+3H
1D2E  6564         54                      DEFW    GET_NUMBER+7H
1D30  6D64         55                      DEFW    GET_NUMBER+0FH
```

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

APPENDIX B

System Variables Definition File

2068 HOME ROM

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

                   13  #EJECT
                   14  #INCL   SYSVAR
                   15  #PAGESIZE 54
                   16
                   17  RST:    MACRO #ROUT
                   18          RST #ROUT
                   19          ENDM
                   20
                   21  ASSERT: MACRO #COND
                   22          COND .NOT.(#COND)
                   23          ERROR IN ASSERTION #COND
                   24          ENDC
                   25          ENDM
                   26
                   27  ; SYSTEM VARIABLES
                   28
                   29  L_LEN   EQU 32          ; # CHARS PER LINE ON THE DISPLAY
                   30  TV_LNS: EQU 24          ; NO. OF LINES ON TV SCREEN
                   31  D_FILE: EQU 4000H       ; ADDRESS OF DISPLAY FILE
                   32  ATTRS:  EQU D_FILE+L_LEN*TV_LNS*8       ; SCREEN ATTRIBUTES
                   33  PRBUFF: EQU ATTRS+TV_LNS*L_LEN  ; PRINTER BUFFER
                   34          ASSERT PRBUFF.AND.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 (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 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 , 6=64 column mode)

From machine code: Register A
From BASIC: In VIMODE

Description:

Sets specified video mode, opening or closing the second display file where needed. This involves determining if there is enough free RAM to open the second display file and if so, moving the BASIC structures and machine language variables area up and the UDG area down to make space for the machine stack and 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 6)
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 (0-63)

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

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

Description:

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

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

- -

- -

Name: SETATT (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 (WRSTR9 from BASIC - String Identifier in PARAM9)

Input: Character Code String

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

From BASIC: String Variable Identifier in
System Variable PARAM9 - 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 PARAM9 prior to invoking WRSTR9, e.g.

```
 0005  LET a$="------string-------"
 0010  POKE 23747,CODE "a"
 0015  IF USR (WRSTR9)<>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 (SCRL9 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 (GETAT9 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 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: 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-100H)
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=3 (Screen Full) is
;                                       returned from the Write String
;                                       service WRSTRG (kWRSTRB).
```

- -

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
DFADDR	4000H
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       CRI80/11 version 10.36.14       16-Mar-84  13:813
        I064.SRC

                                    1           NAME ADL - ASC 001 64-COL.MODE SUPPORT
                                    2   !
                                    3   !
                                    4   !
                                    5   !
                                    6   !
                                    7   !
                                    8   !
                                    9   !
                                   10   !               ********************************
                                   11   !               *                              *
                                   12   !               *            TIMEX             *
                                   13   !               * APPLICATION DEVELOPMENT LIBRARY *
                                   14   !               *                              *
                                   15   !               * NAME: 64-COL.MODE SUPPORT     *
                                   16   !               *                              *
                                   17   !               * ASC NO#  001                  *
                                   18   !               * VERSION: 002                  *
                                   19   !               * AUTHOR:  C. CORCORAN          *
                                   20   !               * DATE:    12/15/83             *
                                   21   !               ********************************
                                   23           SUBTTL VERSION LEVEL CONTROL
                                   24   !
                                   25   !
                                   26   !       VERSION         DATE            COMMENTS
                                   27   !       -------         ----            --------
                                   28   !
                                   29   !         001           11/28/83        ORIGINAL
                                   30   !
                                   31   !         002           12/15/83        ADD STANDARD AND USER-DEFINED
                                   32   !                                       GRAPHICS TO WRCHAR AND GTCHAR
                                   33   !
                                   34   !                                       ADD WRSTRG (WRITE STRING)
                                   35   !                                       CAPABILITY
                                   37           SUBTTL DEFINITIONS
                                   38   !
                                   39   !
                                   40   !
                                   41   !                       ****DEFINITIONS****
                                   42   !
                                   43           INTERN  WRCHAR,WRSTRG,SETCUR,SETATT,SETMSK,CLRSCR,SCRCLL
                                   44           INTERN  GTCHAR,GETATT,GETCUR
                                   45           INTERN  WRCHRB,WRSTRB,SETCUB,SETATB,SETMSB,CLRSCB,SCRLB
                                   46           INTERN  GTCHRB,GETATB,GETCUB
                                   47           INTERN  SETMODE,SETMDB
                                   48   !
  =0018                            49   SCRSI           EQU     24      ; 24 LINES
  =1701                            50   SCINIT          EQU     1701H   ; SCROLL CONTROL INITIALIZATION
  =1800                            51   CLINIT          EQU     1800H   ; CLEAR SCREEN CTL. INIT.
  =3C00                            52   CHRSET          EQU     3C00H   ; ROM CHAR.TABLE-100H
  =0399'                           53   GRPHST          EQU     ((GRPHT)-100H) ;STD.GRAPHICS CHARS.
                                   54   ;
  =0BBE                            55   CHNGVID         EQU     0BBEH   ; ROUTINE IN ROM EXTENSION
  =1720                            56   RECLEN          EQU     1720H   ; ROUTINE IN HOME ROM
                                   57   ;
  =00FF                            58   HRESPT          EQU     0FFH    ; HOME ROM EXT.SELECT PORT (BIT 7)
  =00F4                            59   DAMSPT          EQU     0F4H    ; DOCK HORIZONTAL SELECT PORT
                                   60   ;
  =4C48                            61   VARS            EQU     4C48H   ; SYSTEM VARIABLES
  =4C65                            62   STKEND          EQU     4C65H   ;
  =4C78                            63   UDG             EQU     4C78H   ;
  =4C70                            64   COORDS          EQU     4C70H   ;
  =4C82                            65   RAMTOP          EQU     4C82H   ;
  =4C84                            66   PRAMT           EQU     4C84H   ;
  =4CC2                            67   VIDMOD          EQU     4CC2H   ;
  =4CC3                            68   PARAMS          EQU     4CC3H   ; (LOCATION 23747)
```

```
                        =6840           69  DRIVES          EQU     6840H   ;
                        =12C0           70  INSERTSZ        EQU     7800H-DRIVES
                        =0840           71  MOVESZ          EQU     DRIVES-6000H
                        =F7C0           72  DEST7           EQU     0FFFFH-MOVESZ+1
                        =97C0           73  PIX             EQU     DEST7-6000H
                                        74
                        =1000           75  PIXTBL          EQU     1000H
                                        76
                                        77
                                        78          SUBTTL INPUTS AND ENTRIES
                                        80  ;
                                        81  ;
                                        82  ;
                                        84  ;
                                        85  ;
                                        86  ;               **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                                        87  ;                                                               WITH
                                        88  ;                                                             MODULE LOADED
                                        89  ;                                                             AT 57344(6000H)
                                        90  ;                                                             DECIMAL ADDRESS
                                        91  ;                                                             ---------------
                                        92  ;
0000' 20                                93  DATAB           DEFB    20H     ; DATA BYTE FROM BASIC          57344
0001' C3 0030'                          94  WRCHRB:         JP      WRCH0   ; ENTRY TO WRITE CHARACTER      57345
0004' C3 008C'                          95  WRSTRB:         JP      WRSTR0  ; ENTRY TO WRITE STRING         57348
                                        96  ;                               ; (see PARAMS above)
0007' 0000                              97  LINCOL          DEFW    0       ; SET CURSOR PARAMETERS
                                        98  ;                               ;   LINCOL=COL. (0-63)          57351
                                        99  ;                               ;   LINCOL+1=LINE (0-23)        57352
0009' C3 0139'                         100  SETCUB:         JP      SETCB0  ; ENTRY TO SET CURSOR POSN.     57353
000C' 1800                             101  CLRCTL          DEFW    1800H   ; CLEAR SCREEN CONTROLS
                                       102  ;                               ;   CLRCTL=STARTING LINE NO.     57356
                                       103  ;                               ;   CLRCTL+1=LINE COUNT         57357
000E' C3 01F3'                         104  CLRSCB:         JP      CLRS80  ; ENTRY TO CLEAR SCREEN         57358
0011' 38                               105  ATTCTL          DEFB    38H     ; ATTRIBUTE CONTROL             57361
                                       106  ;                               ;   BIT 7=FLASH (0 IN 64-COL.)
                                       107  ;                               ;       6=BRIGHT(0 IN 64-COL.)
                                       108  ;                               ;       5
                                       109  ;                               ;       4=PAPER COLOR
                                       110  ;                               ;       3
                                       111  ;                               ;       2
                                       112  ;                               ;       1= INK COLOR
                                       113  ;                               ;       0
0012' C3 026E'                         114  SETATB:         JP      SETAB0  ; ENTRY TO SET ATTRIBUTES       57362
0015' 00                               115  MSKCTL          DEFB    0       ; MASK CONTROL FROM BASIC:      57365
                                       116  ;                               ;   BIT 2=INVERSE
                                       117  ;                               ;       0=OVER
0016' C3 0298'                         118  SETMSB:         JP      SETMB0  ; ENTRY TO SET MASK CONTROLS    57366
0019' 0000                             119  GETCTL          DEFW    0       ; "GET" CONTROL FROM BASIC      57369
                                       120  ;                               ; FORMAT IS AS FOR LINCOL
001B' C3 02A1'                         121  GTCHRB:         JP      GTCH80  ; ENTRY TO GET CHARACTER        57371
001E' C3 0321'                         122  GETATB:         JP      GETAB0  ; ENTRY TO GET ATTRIBUTE        57374
0021' C3 0326'                         123  GETCUB:         JP      GETCB0  ; ENTRY TO GET CURSOR POSN.     57377
0024' 00                               124  VIMODE          DEFB    0       ; VIDEO MODE CONTROL            57380
                                       125  ;                               ;       0=NORMAL
                                       126  ;                               ;       4=64-COL.MODE
0025' C3 0345'                         127  SETMOB:         JP      STMO80  ; ENTRY TO SET VIDEO MODE       57381
0028' 1701                             128  SCRCTL          DEFW    1701H   ; SCROLL CONTROL
                                       129  ;                               ;   SCRCTL=STARTING LINE NO.     57384
                                       130  ;                               ;   SCRCTL+1=LINE COUNT         57385
002A' C3 0148'                         131  SCRLB:          JP      SCRL80  ; ENTRY TO SCROLL               57386
                                       133          SUBTTL  OTHER VARIABLES
                                       134  ;
                                       135  ;
                                       136  ;
                                       137  ;
002D' 17                               138  BOTLN           DEFB    17H     ; BOTTOM LINE (LINE AFTER       57389
                                       139  ;                               ; WHICH SCREEN IS CONSIDERED
                                       140  ;                               ; FULL.  AUTOMATIC SCROLL
                                       141  ;                               ; WILL BE DONE IF SCRLCT
                                       142  ;                               ; DOES NOT DECREMENT TO 0.
002E' 01                               143  SCRLCT          DEFB    1       ; SCROLL COUNT - 1 PLUS NO.     57390
                                       144  ;                               ; OF TIMES AUTOMATIC SCROLL
                                       145  ;                               ; WILL BE DONE.
                                       146  ;
                                       147  ;
                                       148  ;
                                       149  ;
002F' 3C00                             150  CHTBL           DEFW    CHRSET  ; ADDRESS OF CHARACTER TABLE    57391
                                       151  ;                               ; (DEFAULTS TO TS2068 SYSTEM
                                       152  ;                               ;  ROM)
0031' 0399'                            153  GRTBL           DEFW    GRPHST  ; ADDRESS OF STD.GRAPHICS TBL.  57393
0033' 40                               154  LINLEN          DEFB    40H     ; LINE LENGTH (64)              57395
0034' 1841                             155  CURPOS          DEFW    1841H   ; CURRENT POSITION (INTERNAL     57396
                                       156  ;                               ;  FORMAT)
0036' 4000                             157  DFADRS          DEFW    4000H   ; CURRENT DISPLAY FILE ADDRS.   57398
0038' 00                               158  MASK0           DEFB    0       ; MASK TO BE APPLIED TO         57400
                                       159  ;                               ; DISPLAY CHARACTERS
0039' 38                               160  ATTRVT          DEFB    38H     ; CURRENT ATTRIBUTES            57401
                                       161  ;                               ;  (BLACK ON WHITE)
003A' 00                               162  GTINDX          DEFB    00H     ; INDEX FOR ADJUSTING           57402
                                       163  ;                               ; CH.CODE IN GTCHAR
003B' 0000                             164  STRGCT          DEFW    0       ; REMAINING COUNT FROM WRSTRG
                                       165  ;                               ; WHEN "SCREEN FULL"            57403
                                       167          SUBTTL  WRITE CHARACTER
                                       168  ;
                                       169  ;
003D'                                  170  WRCH0:                          ; HERE FROM BASIC ENTRY TO DISPLAY THE
                                       171                                  ; CHARACTER WHOSE CODE IS IN "DATA0"
003D' 3A 0000'                         172          LD   A,(DATAB)
                                       173  ;
0040'                                  174  WRCHAR:                         ; ENTRY WITH CODE IN A
0040' CD 0480'                         175          CALL  LDPOSN            ; LOAD REGISTERS FOR CURRENT POSITION
                                       176  ;                               ;   BC=CURSOR POSITION FROM "CURPOS"
                                       177  ;                               ;   HL=DISPLAY FILE ADDRESS FROM "DFADRS
                                       178  ;                               ;   A=CODE FOR CHAR. TO BE DISPLAYED
0043' FE 20                            179          CP    20H               ; TEST VALID RANGE
0045' DA 00C2'                         180          JP    C,INVPAR          ; < 20H
0048' FE A5                            181          CP    0A5H              ;
004A' D2 00C2'                         182          JP    NC,INVPAR         ; > A4H
004D' C5                               183          PUSH  BC                ; SAVE CURSOR POSITION
004E' FE 80                            184          CP    80H               ; TEST IF ASCII PRINTABLE (20H-7FH)
0050' 38 15                            185          JR    C,WRCH12          ; YES
```

```
0032*  FE 90                  106          CP    90H              ; TEST IF STD.GRAPHICS(80H-8FH)
0034*  38 09                  107          JR    C,WRCH11
0036*  ED 4B 5C7B            108          LD    BC,(UDG)         ; USER-DEFINED GRAPHICS(90H-A4H)
003A*  0B                    109          DEC   B                ; ADJUST ADDRESS-100H
003B*  D6 70                 110          SUB   70H              ; ADJUST FOR INDEX INTO TABLE
003D*  18 0C                 111          JR    WRCH13
003F*  ED 4B 0031*           112  WRCH11  LD    BC,(GRTBL)       ; ADDRESS OF GRAPHICS TABLE
0043*  D6 40                 113          SUB   40H              ; ADJUST FOR INDEX INTO TABLE
0045*  18 04                 114          JR    WRCH13
0047*  ED 4B 002F*           115  WRCH12  LD    BC,(CHTBL)       ; CHARACTER TABLE
004B*  0B                    116  WRCH13  EX    DE,HL            ; DE=POSN.IN DISPLAY FILE
004C*  26 00                 117          LD    H,0
004E*  6F                    118          LD    L,A              ; CODE IS INDEX INTO TABLE
004F*  29                    119          ADD   HL,HL
0050*  29                    120          ADD   HL,HL
0051*  29                    121          ADD   HL,HL
0052*  09                    122          ADD   HL,BC            ; HL=PIXEL PATTERN IN TABLE
0053*  C1                    123          POP   BC
0054*  EB                    124          EX    DE,HL
0055*  79                    125          LD    A,C              ; TEST IF END OF LINE
0056*  3D                    126          DEC   A
0057*  3A 0033*              127          LD    A,(LINLEN)
005A*  20 05                 128          JR    NZ,WRCH15
005C*  3C                    129          INC   A                ; START OF NEW LINE
005D*  05                    130          DEC   B                ; BUMP LINE NO.
005E*  4F                    131          LD    C,A              ; LINLEN+1=START OF LINE
005F*  18 01                 132          JR    WRCH15
0061*  3C                    133  WRCH14  INC   A
0062*  B9                    134  WRCH15  CP    C                ; TEST IF START OF LINE
0063*  D5                    135          PUSH  DE
0064*  CC 00C6*              136          CALL  Z,TVPULG         ; TEST IF OFF SCREEN
0067*  D1                    137          POP   DE
                             138                                 ; HERE TO PUT CHARACTER IN DISPLAY FILE
0068*  C5                    139  WRCH2   PUSH  BC               ; CURSOR POSITION
0069*  E5                    140          PUSH  HL               ; DISPLAY FILE ADRS.
006A*  3A 0038*              141          LD    A,(MASKB)
006D*  06 FF                 142          LD    B,-1
006F*  1F                    143          RRA
0070*  30 01                 144          JR    C,WRCH3          ; IF XORING NEW INTO OLD
0072*  04                    145          INC   B                ; B=-1 IF XORING  =0 IF NOT
0073*  1F                    146  WRCH3   RRA
0074*  1F                    147          RRA
0075*  9F                    148          SBC   A,A
0076*  4F                    149          LD    C,A              ; C=-1 FOR INVERSE  =0 IF NOT
0077*  3E 08                 150          LD    A,8              ; SCAN COUNT
0079*  A7                    151          AND   A
007A*  EB                    152          EX    DE,HL
007B*  08                    153  WRCH5   EX    AF,AF'
007C*  1A                    154          LD    A,(DE)           ; OLD BIT ROW
007D*  A0                    155          AND   B
007E*  AE                    156          XOR   (HL)
007F*  A9                    157          XOR   C                ; INVERT IF C=-1
0080*  12                    158          LD    (DE),A           ; UPDATE DISPLAY FILE
0081*  08                    159          EX    AF,AF'
0082*  14                    160          INC   D                ; NEXT SCAN
0083*  23                    161          INC   HL               ; NEXT BYTE IN CHAR.FILE
0084*  3D                    162          DEC   A                ; TEST IF MORE SCANS
0085*  20 F4                 163          JR    NZ,WRCH5
0087*  E1                    164  WRCH7   POP   HL
0088*  C1                    165          POP   BC
0089*  0C                    166          DEC   C                ; ADJUST CURSOR POSITION
008A*  79                    167          LD    A,C              ; TEST IF EVEN/ODD
008B*  CB 47                 168          BIT   0,A
008D*  28 06                 169          JR    NZ,CCLODD
008F*  7C                    170  CCLEVN  LD    A,H
0090*  F6 20                 171          OR    20H
0092*  67                    172          LD    H,A
0093*  18 05                 173          JR    WRCHNXT
0095*  7C                    174  CCLODD  LD    A,H
0096*  E6 DF                 175          AND   0DFH
0098*  67                    176          LD    H,A
0099*  23                    177          INC   HL
009A*  CC 0485*              178  WRCHNXT CALL  STPOSN           ; DPI NEXT BYTE
009D*  0E 00                 179  GOJORET LD    C,0              ; STORE NEW POSITION
009F*  06 00                 180  ERRRET  LD    B,0              ; RETURN BC=0
00A1*  C9                    181          RET                    ; ENTER HERE WITH C NON-ZERO
                             182  ;
00A2*  0E 01                 183  INVPAR  LD    C,1              ; RETURN BC=1 FOR INVALID PARAMETERS
00A4*  18 F9                 184          JR    ERRRET
                             185  ;
00A6*  3E 18                 186  TVPULG  LD    A,SCRSI          ; TEST IF NEW LINE IS OFF SCREEN
00A8*  96                    187          SUB   B                ; A=LINE NO.
00A9*  57                    188          LD    D,A              ; SAVE IN D
00AA*  3A 0020*              189          LD    A,(BOTLN) ; GET BOTTOM LINE
00AD*  BA                    190          CP    D
00AE*  D2 B456*              191          JP    NC,JPOPOSN       ; ON SCREEN - RETURN TO WRITE CHAR.
                             192                                 ;  VIA UPDATE AND STORE POSITION
00B1*  3A 0022*              193          LD    A,(SCRLCT)       ; TEST SCROLL COUNT
00B4*  3C                    194          DEC   A
00B5*  28 08                 195          JR    NZ,TVPUL1        ; DO AUTOMATIC SCROLL USING SCRCTL
00B7*  3E 01                 196          LD    A,1              ; REINITIALIZE TO 1
00B9*  32 0022*              197          LD    (SCRLCT),A
00BC*  C1                    198          POP   BC
00BD*  C1                    199          POP   BC               ; DISCARD RETURN TO WRITE CHAR.
00BE*  0E 03                 200          LD    C,3              ; RETURN BC=3 FOR SCREEN FULL
00C0*  18 DD                 201          JR    ERRRET
00C2*  32 0022*              202  TVPUL1  LD    (SCRLCT),A       ; UPDATE VARIABLE
00C5*  ED 4B 0020*           203          LD    BC,(SCRCTL)      ; GET SCROLL CONTROLS (STARTING LINE
                             204                                 ;   AND LINE COUNT)
00C9*  C3 B15F*              205          JP    SCRL0            ; RETURN TO WRITE CHAR. VIA SCROLL
                             206  ;
                             207          SUBTTL WRITE STRING
                             208  ;
                             209  ;                              ; HERE FROM BASIC ENTRY TO
                             210  ;                              ; WRITE CHARACTER STRING TO SCREEN.
                             211  ;                              ; STRING IDENTIFIER (CH.CODE) IN
                             212  ;                              ; SYSTEM VARIABLE PARAMS AT 5CC3H
00CC*  3A 5CC3               213  WRSTR0  LD    A,(PARAMS)       ; GET STRING ID
00CF*  E6 1F                 214          AND   1FH              ; MASK OFF UPPER BITS
00D1*  F6 40                 215          OR    40H              ; STRING VARIABLE IDENTIFIER
00D3*  57                    216          LD    D,A              ; SAVE IN D
00D4*  2A 5C4B               217          LD    HL,(VARS)        ; FIND STRING
00D7*  7E                    218  WRSTR1  LD    A,(HL)           ;
```

```
00F8' E6 7F              299           AND  07FH              ; TEST IF END OF VARS AREA
00FA' 28 35              300           JR   Z,NOSTRG          ; NO FIND - RETURN BC=2
00FC' BA                 301           CP   D                 ; TEST IF MATCH
00FD' 28 08              302           JR   Z,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' EB                 305           EX   DE,HL             ; ADRS. TO HL
0104' D1                 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' 78                 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,WRSTX1         ; EXIT IF INVALID CODE OR
                         329 ;                                ; IF SCREEN FULL
0119' C1                 330 WRSTR3    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 WRSTX1    LD   A,C               ; SAVE ERROR
0123' FE 01              340           CP   1                 ; TEST IF UNRECOGNIZED CODE
0125' 28 F2              341           JR   Z,WRSTR3          ; CONTINUE
0127' E1                 342           POP  HL                ; REMAINING COUNT TO HL
0128' 22 0038'           343           LD   (STRGCT),HL       ; STORE REMAINING COUNT
012B' E1                 344           POP  HL                ; ADDRESS TO HL
012C' 0E 03              345           LD   C,3               ; RETURN BC=3
012E' 06 00              346 WRSTX2    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   WRSTX2
                         351 ;
                         353           SUBTTL POSITION CONTROL
                         354 ;
0135'                    355 SETCB01:                         ; HERE FROM BASIC ENTRY.  PARAMETERS
                         356                                  ; IN LINCOL
0135' ED 4B 0007'        357           LD   BC,(LINCOL)
                         358 ;
0139'                    359 SETCUR:                          ; ENTRY WITH LINE/COL. IN BC REG.
0139' CD 043A'           360           CALL   TSTPAR          ; TEST PARAMETERS FOR VALIDITY
013C' CD 0448'           361           CALL   CONVFM          ; CONVERT TO INTERNAL FORMAT
013F' CD 0456'           362           CALL   UPDPOSN         ; CALCULATE AND STORE POSITION
0142' C3 008D'           363           JP     GOODRET         ; RETURN BC=0
                         364 ;
                         366           SUBTTL SCROLL
                         367 ;
                         368 ;
                         369 ;
0145'                    370 SCRL001:                         ; HERE FROM BASIC ENTRY TO SCROLL
                         371 ;                                ; SCREEN
0149' ED 4B 0028'        372           LD   BC,(SCRCTL)       ; GET CONTROL INFO.
                         373 ;
0149'                    374 SCROLL:                          ; HERE WITH CONTROLS IN BC
                         375 ;                                ;  B=NO. OF LINES
                         376 ;                                ;  C=STARTING LINE NO.
0149' 78                 377           LD   A,B               ; TEST VALIDITY
014A' A7                 378           AND  A                 ;
014B' CA 00C2'           379           JP   Z,INVPAR          ; ERROR IF COUNT=0
014E' 81                 380           ADD  A,C
014F' FE 01              381           CP   1
0151' DA 00C2'           382           JP   C,INVPAR          ; ERROR IF B+C<1
0154' FE 19              383           CP   25
0156' D2 00C2'           384           JP   NC,INVPAR         ; ERROR IF B+C>24
0159' CD 015F'           385           CALL SCRL0             ; DO SCROLL
015C' C3 008D'           386           JP   GOODRET           ; RETURN BC=0
                         387 ;
015F' C5                 388 SCRL0     PUSH BC
0160' 3E 19              389           LD   A,SCRSZ           ; GET STARTING LINE IN INTERNAL FORMAT
0162' 91                 390           SUB  C
0163' 47                 391           LD   B,A
0164' 3A 0033'           392           LD   A,(CLINLEN)
0167' 3C                 393           INC  A                 ; COL. 0
0168' 4F                 394           LD   C,A
0169' CD 0474'           395           CALL LMOV
016C' C1                 396           POP  BC
016D' C5                 397           PUSH BC                ; SAVE ORIG.BC FOR EXIT
016E' 7C                 398 TESTAC    LD   A,H               ; TEST IF AT START OF BLOCK
016F' A7                 399           AND  A
0170' 28 51              400           JR   Z,STBLK           ; YES
0172' 07                 401           RLCA
0173' 07                 402           RLCA
0174' 07                 403           RLCA
0175' 4F                 404           LD   C,A               ; GET NO. OF LINES THIS BLOCK
0176' 3E 08              405           LD   A,8
0178' 91                 406           SUB  C
0179' B8                 407           CP   B                 ; TEST AGAINST TOTAL LINES
017A' 38 3F              408           JR   C,GRBLK           ; TOTAL GREATER THAN THIS BLOCK
017C' 78                 409 INSDBLK   LD   A,B
017D' 06 00              410           LD   B,0               ; REMAINING COUNT
```

```
0178'  C5                411             PUSH  BC
0180'  47                412  LOOP       LD    B,A               ; B=LINES THIS BLOCK
0181'  91 00             413             LD    C,0               ; C=SCAN COUNT
0183'  78                414  LOOP0      LD    A,B
0184'  C5                415             PUSH  BC                ; SAVE SCAN COUNT
0185'  0F                416             @RCA
0186'  0F                417             @RCA
0187'  0F                418             @RCA                    ; CALCULATE NO. OF BYTES
0188'  4F                419             LD    C,A
0189'  26 00             420             LD    B,0               ; BC=32@@ OF LINES
018B'  EB                421  LOOP1     EX    DE,HL
018C'  21 FFE0           422             LD    HL,-20H           ; GET ADRS. OF PREV. LINE
018F'  19                423             ADD   HL,DE
0190'  EB                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'  CB 6F             431             BIT   5,A               ; TEST IF OP1
019A'  20 05             432             JR    NZ,NXTSC          ; NO
019C'  F6 20             433             OR    20H
019E'  67                434             LD    H,A               ; DO OP2
019F'  18 EA             435             JR    LOOP1
01A1'  E6 DF             436  NXTSC     AND   0DFH              ; BACK TO OP1
01A3'  67                437             LD    H,A
01A4'  24                438             INC   H                 ; NEXT SCAN LINE
01A5'  C1                439             POP   BC                ; SCAN COUNT
01A6'  0D                440             DEC   C
01A7'  20 DA             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'  28 00             446             LD    L,0               ; START OF NEXT BLOCK
01B0'  18 BC             447             JR    TESTAG            ; DO REMAINING LINE(S)
01B2'  C1                448  SCRLRT    POP   BC                ; ORIG.BC
01B3'  79                449             LD    A,C               ; STARTING LINE
01B4'  80                450             ADD   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 52             454             JR    CLRSC0            ; CLEAR VACATED LINE AND RETURN
                         455  ;
01BB'  4F                456  BBOLK     LD    C,A
01BC'  78                457             LD    A,B
01BD'  91                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'  79                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'  C5                467  STBLK0   PUSH  BC                ; SAVE SCAN COUNT
01C8'  EB                468  STBLK1   EX    DE,HL
01C9'  21 F8E0           469             LD    HL,-720H
01CC'  19                470             ADD   HL,DE
01CD'  EB                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,STBLK2
01DA'  F6 20             479             OR    20H               ; DO OP2
01DC'  67                480             LD    H,A
01DD'  18 E9             481             JR    STBLK1
01DF'  E6 DF             482  STBLK2    AND   0DFH              ; BACK TO OP1
01E1'  67                483             LD    H,A
01E2'  24                484             INC   H                 ; TO NEXT SCAN LINE
01E3'  C1                485             POP   BC
01E4'  0D                486             DEC   C                 ; TEST IF MORE SCANS
01E5'  20 E0             487             JR    NZ,STBLK0
01E7'  C1                488             POP   BC                ; REMAINING LINE COUNT
01E8'  78                489             LD    A,B
01E9'  A7                490             AND   A
01EA'  28 C6             491             JR    Z,SCRLRT
01EC'  11 F020           492             LD    DE,-720H          ; ADJUST TO LINE 1
01EF'  19                493             ADD   HL,DE
01F0'  C3 01AE'          494             JP    TESTAG            ; BC=LINE COUNT
                         495                                     ; HL=OP ADDRESS
                         496  ;
                         498             SUBTTL  CLEAR SCREEN
                         499  ;
                         500  ;
01F3'                    501  CLRSB0:                            ; HERE FROM BASIC ENTRY TO CLEAR SCREEN
01F3'  ED 4B 000C'       502             LD    BC,(CLRCTL)       ; GET CONTROL INFO.
                         503  ;
01F7'                    504  CLRSCN:                           ; ENTRY WITH BC=LINE COUNT AND
                         505                                     ; STARTING LINE NO. FOR CLEAR
01F7'  78                506             LD    A,B
01F8'  A7                507             AND   A                 ; TEST VALIDITY
01F9'  CA 00C2'          508             JP    Z,INVPAR          ; ERROR IF COUNT=0
01FC'  81                509             ADD   A,C
01FD'  FE 01             510             CP    1
01FF'  DA 00C2'          511             JP    C,INVPAR          ; ERROR IF B+C<1
0202'  FE 19             512             CP    25
0204'  D2 00C2'          513             JP    NC,INVPAR         ; ERROR IF B+C>24
0207'  CD 020D'          514             CALL  CLRSC0            ; DO SERVICE
020A'  C3 00BD'          515             JP    GOODRET           ; RETURN BC=0
                         516  ;
                         517  ;
020D'  C5                518  CLRSC0  PUSH  BC
020E'  3E 18             519             LD    A,SCRSZ           ; CONVERT TO INTERNAL FORMAT
```

```
0210'  91                  920           SUB   C
0211'  47                  921           LD    B,A
0212'  3A 0033'            922           LD    A,(LINLEN)
0215'  3C                  923           INC   A                 ; SET TO COL.0
0216'  4F                  924           LD    C,A
0217'  CD 0474'            925           CALL  LNBU              ; GET ADDRESS
021A'  50                  926           LD    D,B
021B'  59                  927           LD    E,C               ; SAVE POSITION
021C'  C1                  928           POP   BC                ; ORIG. BC
021D'  D5                  929           PUSH  DE
021E'  78                  930  CLRSC1   LD    A,L               ; GET # OF LINES THIS BLOCK
021F'  07                  931           RLCA
0220'  07                  932           RLCA
0221'  07                  933           RLCA
0222'  4F                  934           LD    C,A
0223'  3E 00               935           LD    A,0
0225'  91                  936           SUB   C
0226'  B8                  937           CP    B                 ; COMPARE TO TOTAL LINE COUNT
0227'  38 3C               938           JR    C,CLRSC7
0229'  78                  939  CLRSC2   LD    A,B               ; NO. OF LINES
022A'  06 00               940           LD    B,0               ; REMAINING LINES
022C'  C5                  941           PUSH  BC
022D'  47                  942  CLRSC3   LD    B,A
022E'  0E 08               943           LD    C,8               ; SCAN COUNT
0230'  78                  944  CLRSC4   LD    A,B
0231'  C5                  945           PUSH  BC
0232'  E6 07               946           AND   7
0234'  0F                  947           RRCA
0235'  0F                  948           RRCA
0236'  0F                  949           RRCA
0237'  4F                  950           LD    C,A               ; BC=32*NO. OF LINES
0238'  06 00               951           LD    B,0               ; (=0 IF 256 FOR 8 LINES)
023A'  0D                  952           DEC   C                 ; ADJUST
023B'  C5                  953  CLRSC5   PUSH  BC
023C'  E5                  954           PUSH  HL
023D'  54                  955           LD    D,H
023E'  5D                  956           LD    E,L
023F'  36 00               957           LD    (HL),00           ; CLEAR DP
0241'  13                  958           INC   DE
0242'  ED B0               959           LDIR
0244'  E1                  960           POP   HL
0245'  C1                  961           POP   BC
0246'  7C                  962           LD    A,H
0247'  CB 6F               963           BIT   5,A
0249'  20 05               964           JR    NZ,CLRSC6
024B'  F6 20               965           OR    20H
024D'  67                  966           LD    H,A               ; DO DP2
024E'  18 E8               967           JR    CLRSC5
0250'  E6 DF               968  CLRSC6   AND   0DFH             ; RESTORE DP1
0252'  67                  969           LD    H,A
0253'  24                  970           INC   H                 ; NEXT SCAN ROW
0254'  C1                  971           POP   BC                ; POP SCAN COUNT
0255'  0D                  972           DEC   C
0256'  20 D0               973           JR    NZ,CLRSC4
0258'  C1                  974           POP   BC                ; TOTAL LINE COUNT
0259'  78                  975           LD    A,B
025A'  A7                  976           AND   A
025B'  28 05               977           JR    Z,CLRSX1
025D'  2E 00               978           LD    L,0               ; HL=START OF NEXT BLOCK
025F'  C3 021E'            979           JP    CLRSC1            ; B=REMAINING LINE COUNT
0262'  C1                  980  CLRSX1   POP   BC                ; POSITION
0263'  C3 0456'            981           JP    UPDPOSN           ; RETURN VIA UPDATE AND STORE POSITION
0266'  4F                  982  CLRSC7   LD    C,A               ; SAVE NO. LINES THIS BLOCK
0267'  78                  983           LD    A,B
0268'  91                  984           SUB   C                 ; ADJUST TOTAL LINE COUNT
0269'  47                  985           LD    B,A
026A'  C5                  986           PUSH  BC                ; REMAINING LINE COUNT
026B'  79                  987           LD    A,C               ; LINES THIS BLOCK
026C'  18 BF               988           JR    CLRSC3            ;
                           990           SUBTTL ATTRIBUTE CONTROL
                           991  ;
                           992  ;
                           993  ;
026E'                      994  SETABO:                          ; HERE FROM BASIC ENTRY TO SET ATTRIBUTES
026E'  3A 0011'            995           LD    A,(ATTCTL)        ; GET CONTROL INFO.
                           996  ;
0271'                      997  SETATT:                          ; ENTRY WITH ATTRIBUTE BYTE IN A
0271'  E6 07               998           AND   7                 ; GET INK SELECTION
0273'  4F                  999           LD    C,A               ; SAVE
0274'  3A 5CC2            1000           LD    A,(VIDMOD)        ; TEST IF DP2 OPEN
0277'  A7                  1001          AND   A
0278'  CA 00C2'            1002          JP    Z,INVPAR          ; INVALID IF DP2 NOT OPEN
027B'  79                  1003          LD    A,C
027C'  17                  1004          RLA
027D'  17                  1005          RLA
027E'  17                  1006          RLA                     ; SHIFT TO H/W POSITION
027F'  F5                  1007          PUSH  AF                ; SAVE ROTATED VALUE
0280'  F6 06               1008          OR    6                 ; SET H/W MODE
0282'  47                  1009          LD    B,A               ; VALUE FOR H/W
0283'  DB FF               1010          IN    A,(HREXPT)        ;
0285'  E6 C0               1011          AND   0C0H              ; PRESERVE BITS 7 AND 6
0287'  B0                  1012          OR    B                 ; SET VIDEO MODE VALUE
0288'  D3 FF               1013          OUT   (HREXPT),A        ; SET H/W
028A'  F6 40               1014          OR    40H               ; UPDATE VIDMOD
028C'  32 5CC2             1015          LD    (VIDMOD),A        ;
028F'  F1                  1016          POP   AF                ; INK SELECTION IN BITS 3-5
0290'  2F                  1017          CPL                     ; GET COMPLEMENTARY COLOR
0291'  B1                  1018          OR    C                 ; COMBINE WITH INK
0292'  32 0039'            1019          LD    (ATTBYT),A        ; SAVE IN VARIABLE
0295'  C3 0080'            1020          JP    GOODRET           ; RETURN BC=0
                           1021 ;
                           1022 ;
                           1023 ;
0298'                      1024 SETMSO:                          ; HERE FROM BASIC ENTRY TO SET MASK
0298'  3A 0015'            1025          LD    A,(MSKCTL)        ; GET CONTROL INFO.
                           1026 ;
0298'                      1027 SETMSK:                          ; ENTRY WITH MASK VALUE IN A
0298'  32 0038'            1028          LD    (MASKB),A         ; STORE MASK FOR APPLICATION TO FUTURE
029E'  C3 0080'            1029          JP    GOODRET           ; DISPLAYS
                           1030 ;
```

```
                                    SUBTTL  'GET' ROUTINES
                                ;
                                ;
02A1'                           GTCHB01:                        ; HERE FROM BASIC ENTRY TO GET CHARACTER
                                                                ; (RETURNS CCDE FOR CHARACTER AT
                                                                ;  POSITION SPECIFIED IN GETCTL)
02A1' ED 4B 0019'                       LD      BC,(GETCTL)     ; GET CONTROL INFO.
                                ;
02A5'                           GTCHAR1:                        ; ENTRY WITH POSITION IN BC
02A5' CD 043A'                          CALL    TSTPAR          ; TEST PARAMETERS
02A8' CD 0448'                          CALL    CONVRW          ; CONVERT TO INTERNAL FORMAT
02AB' CD 0439'                          CALL    CALCPOS         ; GET DISPLAY FILE ADRS.
02AE' ED 5B 002F'                       LD      DE,(CHTBL)      ; CHAR.TABLE
02B2' 06 60                             LD      B,96            ; NO. OF PRINTABLE CHARACTERS
02B4' 3E 80                             LD      A,80H           ; SET ADJUSTMENT INDEX
02B6' 32 003A'                  GTCH1   LD      (GTINDX),A
02B9' 14                                INC     D               ; ADJUST TO START OF TABLE
02BA' E8                                EX      DE,HL           ; DF ADRS.IN DEICHAR.SET IN HL
02BB' C5                        GTCH2   PUSH    BC              ; SAVE COUNT
02BC' D5                                PUSH    DE              ; SAVE ADRS. IN DF
02BD' E5                                PUSH    HL              ; SAVE ADRS. IN CHAR.TABLE
02BE' 1A                                LD      A,(DE)          ; SCAN ROW FROM DF
02BF' AE                                XOR     (HL)            ; TEST AGAINST CHAR. SET
02C0' 28 10                             JR      Z,GTCH3         ; MATCH
02C2' F5                                PUSH    AF
02C3' 3A 003A'                          LD      A,(GTINDX)
02C6' FE 90                             CP      90H             ; TEST IF STD.GRAPHICS
02C8' 20 03                             JR      NZ,GTCH23
02CA' F1                                POP     AF
02CB' 18 27                             JR      GTCH4           ; DO NOT TEST FOR INVERSE
02CD' F1                        GTCH23  POP     AF
02CE' 3C                                INC     A               ; TEST INVERSE
02CF' 20 23                             JR      NZ,GTCH4
02D1' 3D                                DEC     A               ; A=-1 FOR INVERSE
02D2' 4F                        GTCH3   LD      C,A             ; C=0 FOR MATCH  -1 FOR INVERSE
02D3' 06 07                             LD      B,7
02D5' 14                        GTCH31  INC     D               ; NEXT SCAN IN DF
02D6' 23                                INC     HL              ; NEXT BYTE IN CHAR.SET
02D7' 1A                                LD      A,(DE)
02D8' AE                                XOR     (HL)
02D9' A9                                XOR     C
02DA' 20 10                             JR      NZ,GTCH4        ; NO MATCH
02DC' 10 F7                             DJNZ    GTCH31  ; FOR NEXT SCAN
                                                        ;
                                                        ; HERE IF MATCH WITH (SP)=CHAR IN CHAR SET
                                                        ; (SP+2)=CHAR.IN DISPLAY FILE (SP+9)=CHAR.
                                                        ; COUNT
02DE' 79                                LD      A,C     ; A=-1 IF MATCH ON INVERSE
02DF' C1                                POP     BC
02E0' C1                                POP     BC
02E1' C1                                POP     BC
02E2' 4F                                LD      C,A     ; B=COUNT C=-1 IF MATCH ON INVERSE
02E3' 3A 003A'                          LD      A,(GTINDX)      ; ADJUST CHAR.CODE
02E6' 90                                SUB     B       ; A=CHAR.CODE
02E7' FE 20                             CP      20H     ; TEST IF SPACE
02E9' 20 05                             JR      NZ,GTCH32
02EB' 3C                                INC     C       ; TEST IF MATCH ON INVERSE SPACE
02EC' 20 02                             JR      NZ,GTCH32
02EE' 3E 8F                             LD      A,8FH   ; SET CODE FOR GRAPHICS BLOCK
02F0' 4F                        GTCH32  LD      C,A
02F1' 06 00                             LD      B,0
02F3' C9                                RET             ; CHAR.CODE IN A AND IN BC
                                ;
02F4' E1                        GTCH4   POP     HL      ; HERE WHEN NO MATCH
02F5' 11 0008                           LD      DE,8    ; LOOK AT NEXT CHAR. IN SET
02F8' 19                                ADD     HL,DE           ; CHAR. SET
02F9' D1                                POP     DE              ; CHAR. IN DF
02FA' C1                                POP     BC              ; CHAR.COUNT
02FB' 10 BE                             DJNZ    GTCH2   ; LOOK AGAIN
                                                        ;
                                                        ; HERE TO TEST IF DONE
02FD' EB                                EX      DE,HL   ; DF ADRS.TO HL
02FE' 3A 003A'                          LD      A,(GTINDX)
0301' FE 80                             CP      80H     ; TEST IF STD.CHR.SET
0303' 20 0A                             JR      NZ,GTCH5        ; TRY GRAPHICS
0305' ED 5B 0031'                       LD      DE,(GRTBL)      ; STD.GRAPHICS TABLE
0309' 06 10                             LD      B,16    ; NO. OF ENTRIES
030B' 3E 90                             LD      A,90H   ; INDEX
030D' 18 A7                             JR      GTCH1   ;
030F' FE 90                     GTCH5   CP      90H     ; TEST IF STD.GRAPHICS
0311' 20 08                             JR      NZ,GTCH6        ; DONE IF NOT
0313' ED 5B 5C7B                        LD      DE,(UDG)        ; TRY USER-DEFINED GRAPHICS AREA
0317' 15                                DEC     D       ; ADJUST ADDRESS-100H
0318' 06 15                             LD      B,21    ; NO OF ENTRIES
031A' 3E A5                             LD      A,0A5H  ; INDEX
031C' 18 98                             JR      GTCH1
031E' 48                        GTCH6   LD      C,B     ; HERE WHEN NO MATCH ANYWHERE
031F' AF                                XOR     A       ; SET BC AND A=ZERO
0320' C9                                RET
                                ;
                                ;                       ; GET ATTRIBUTE BYTE
                                ;                       ; NOTE THAT IN 64-COL.MODE ALL SCREEN POSITIONS
                                ;                       ; HAVE THE SAME ATTRIBUTES, THEREFORE IT IS
                                ;                       ; NOT NECESSARY TO USE THE 'GETCTL' POSITION
                                ;                       ; PARAMETERS
                                ;
0321'                           GETAB01:                ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                                ;
0321' 3A 0039'                  GETATT: LD      A,(ATTRBYT) ; GET ATTRIBUTE BYTE
0324' 4F                                LD      C,A     ; PUT IN BC
0325' 06 00                             LD      B,0
0327' C9                                RET
                                ;                       ; GET CURSOR POSITION
0328'                           GETC01:                 ; HERE FROM BASIC ENTRY
```

```
0328'                                   736   GETCUR:
0328'  ED 4B 0034'                     737           LD   BC,(CURPOS)        ; GET INTERNAL POSITION
032C'  CD 0448'                         738           CALL CONVPM             ; CONVERT TO USER FORMAT
032F'  3A 0033'                         739           LD   A,(LINLEN)
0332'  B9                               740           CP   C                  ; TEST IF END OF LINE(=64)
0333'  20 08                           741           JR   NZ,GETC1           ; NO
0335'  0E 00                           742           LD   C,0                ; NEXT POSN.START OF NEXT LINE
0337'  78                              743           LD   A,B
0338'  3C                              744           INC  A                  ; BUMP TO NEXT LINE
0339'  47                              745           LD   B,A
033A'  FE 18                           746           CP   24                 ; TEST IF OFF SCREEN
033C'  38 02                           747           JR   C,GETC1            ; NO
033E'  06 17                           748           LD   B,23               ; POSN. IS AT BOTTOM LINE
0340'  ED 43 0007'                     749   GETC1   LD   (LINCOL),BC        ; STORE FOR BASIC PROGRAM
0344'  C9                              750           RET                     ; VALUES ALSO IN BC
                                       751   ;
                                       753           SUBTTL  VIDEO MODE CONTROL
                                       754   ;
                                       755   ;
0345'                                  756   STMODO:                 ; HERE FROM BASIC ENTRY WITH MODE IN
                                       757                           ; 'VIMODE'
0345'  3A 0024'                        758           LD  A,(VIMODE)
                                       759   ;
0348'                                  760   SETMODE:                ; ENTRY WITH MODE IN A
                                       761   ;
0348'  A7                              762           AND  A
0349'  28 4A                           763           JR   Z,SETM01   ; TEST FOR VALIDITY
034B'  FE 06                           764           CP   6
034D'  C2 00C2'                        765           JP   NZ,INVPAR  ; INVALID PARAMETERS
0350'  21 1701                        766           LD   HL SCINIT  ; INITIALIZE VARIABLES
0353'  22 0028'                        767           LD   (SCRCTL),HL ; SCROLL CONTROL
0356'  3E 17                           768           LD   A,23       ;
0358'  32 002D'                        769           LD   (BOTLN),A  ; BOTTOM LINE
035B'  3E 01                           770           LD   A,1        ;
035D'  32 002E'                        771           LD   (SCRLCT),A ; SCROLL COUNT
0360'  3E 40                           772           LD   A,64       ;
0362'  32 0033'                        773           LD   (LINLEN),A ; LINE LENGTH
0365'  21 1800                        774           LD   HL,CLINIT  ;
0368'  22 000C'                        775           LD   (CLRCTL),HL ; CLEAR SCREEN CONTROL
036B'  AF                              776           XOR  A
036C'  32 0038'                        777           LD   (MASKB),A  ; MASK BYTE
036F'  32 0035'                        778           LD   (STRGCT),A ; MRSTRG REMAINING COUNT
0372'  32 003C'                        779           LD   (STRGCT+1),A ;
0375'  21 3C00                        780           LD   HL,CHRSET  ; STD.CHAR.SET
0378'  22 002F'                        781           LD   (CHTBL),HL ;
037B'  21 0395'                        782           LD   HL,GRPHST  ; STD. GRAPHICS SET
037E'  22 0031'                        783           LD   (GRTBL),HL ;
                                       784   ;                       ;
                                       785   ;                       ; INITIALIZE BASIC INPUTS
0381'  32 0000'                        786           LD   (DATAB),A  ; DATA BYTE
0384'  32 0007'                        787           LD   (LINCOL),A ; COLUMN
0387'  32 0008'                        788           LD   (LINCOL+1),A ; LINE
038A'  32 0015'                        789           LD   (MSKCTL),A ; MASK CONTROL
038D'  32 0019'                        790           LD   (GETCTL),A ; 'GET' COLUMN
0390'  32 001A'                        791           LD   (GETCTL+1),A ; 'GET' LINE
0393'  3E 06                           792           LD   A,6        ; SET H/W TO 64-COL.MODE
                                       793   ;
0395'  47                              794   SETM01  LD   B,A        ; SAVE MODE
0396'  3A 9CC2                        795           LD   A,(VIDMOD)
0399'  A7                              796           AND  A          ; TEST CURRENT MODE
039A'  20 1D                           797           JR   NZ,SETM02
039C'  B0                              798           OR   B          ; CURRENT MODE=0  TEST IF NEW MODE=0
039D'  CA 0C9D'                        799           JP   Z,GOODRET  ; RETURN BC=0
03A0'  21 12C9                        800           LD   HL,INSERTSZ
03A3'  11 0840                        801           LD   DE,MOVESZ
03A6'  19                              802           ADD  HL,DE      ; HL=TOTAL ROOM NEEDED
03A7'  ED 5B 9C65                     803           LD   DE,(STKEND)
03AB'  19                              804           ADD  HL,DE      ; ADD MEMORY IN USE
03AC'  DA 03F6'                        805           JP   C,EXTNRM   ; NOT ENOUGH MEMORY TO OPEN DP2
03AF'  ED 5B 9C82                     806           LD   DE,(RAMTOP)
03B3'  A7                              807           AND  A
03B4'  ED 52                           808           SBC  HL,DE
03B6'  D2 03F6'                        809           JP   NC,EXTNRM  ; NOT ENOUGH MEMORY
03B9'  CD 0421'                        810   SETM02  CALL ENBLEXT    ; ENABLE ROM EXT.
03BC'  78                              811           LD   A,B        ; MODE TO A
03BD'  CD 03F8'                        812           CALL GETVAL     ; H/W VALUE TO B; VIDMOD VALUE TO C
03C0'  C5                              813           PUSH BC         ; SAVE
03C1'  78                              814           LD   A,B
03C2'  CD 0E8E                        815           CALL CHNGVID    ; CALL RTN. TO OPEN/CLOSE 2ND DP
03C5'  CD 0428'                        816           CALL ENBLHOME   ; REENABLE HOME BANK
03C8'  C1                              817           POP  BC
03C9'  79                              818           LD   A,C
03CA'  32 9CC2                        819           LD   (VIDMOD),A; UPDATE VIDMOD
03CD'  CB 77                           820   UPDATE  BIT  6,A                ; UPDATE SYSTEM VARS. BASED ON MODE
03CF'  CA 0080'                        821           JP   Z,GOODRET          ; DONE IF MODE 0
03D2'  3E C1                           822   SXT4    LD   A,0C1H             ; 64/80-COL. MODE
03D4'  B0                              823           OR   B                  ; STORE ATTRIBUTE BYTE BASED
03D5'  2F                              824           CPL                     ; ON INK SELECTION
03D6'  CB 38                           825           SRL  B
03D8'  CB 38                           826           SRL  B
03DA'  CB 38                           827           SRL  B
03DC'  B0                              828           OR   B
03DD'  32 0039'                        829           LD   (ATTBYT),A         ; ATTRIBUTE BYTE
03E0'  32 0011'                        830           LD   (ATTCTL),A         ; BASIC PARAMETER
03E3'  3A 0033'                        831   UPDT1   LD   A,(LINLEN)         ; GET LINE LENGTH
03E6'  3C                              832           INC  A
03E7'  4F                              833           LD   C,A                ; COLUMN
03E8'  06 18                           834           LD   B,SCRSZ            ; BC=HOME POSITION (LN.0/COL.0)
03EA'  21 0000                        835           LD   HL,0
03ED'  22 9C7D                        836           LD   (CDCPOS),HL        ; INITIALIZE PLOT POSITION
03F0'  CD 0456'                        837           CALL UPCPOSN            ; UPDATE AND STORE HOME POSITION
03F3'  C3 0080'                        838           JP   GOODRET            ; RETURN BC=0
03F6'  0E 02                           839   EXTNRM  LD   C,2                ; RETURN BC=2 FOR NO ROOM
03F8'  C3 008F'                        840           JP   ERRRET
                                       841   ;
```

```
                        843           SUBTTL  INTERNAL SUB-RTNS.
                        844   !
                        845   !
                        846   !
                        847                                   ! INPUT:  VIDEO MODE IN A
                        848                                   ! OUTPUT: H/W SETTING IN B
                        849                                   !         VIDMOD SETTING IN C
                        850                                   ! RETURNS NZ STATUS IF MODE INVALID
                        851                                   !
                        852                                   !  VALID VALUES:  INPUT     H/W     VIDMOD
                        853                                   !                 -----     ---     ------
                        854                                   !   @=DISPLAY FILE ACTIVE AT SCREEN
                        855                                   !
                        856                                   !    DF1 ONLY       0        0        0
                        857                                   !   DF1@& DF2      80        0       80
                        858                                   !    DF1 & DF2@     1        1       81
                        859                                   !    HIGH RES.GR.   2        2        2
                        860                                   !  64/80-COLUMN  86 - 3E  86 - 3E   46 - 7E
03F8'  4F               861   GETVAL  LD    C,A
03FC'  E8 47            862           BIT   0,A
03FE'  28 16            863           JR    NZ,TST01
0400'  FE 80            864           CP    80H
0402'  28 14            865           JR    Z,SET0
0404'  CB 7F            866           BIT   7,A
0406'  C0               867           RET   NZ
0407'  47               868           LD    B,A
0408'  FE 02            869           CP    2
040A'  C8               870           RET   Z
040B'  E6 C6            871           AND   0C6H
040D'  EE 06            872           XOR   6
040F'  C0               873           RET   NZ
0410'  3E 40            874           LD    A,40H
0412'  81               875           ADD   C
0413'  4F               876           LD    C,A
0414'  AF               877           XOR   A
0415'  C9               878           RET
0416'  FE 01            879   TST01   CP    1
0418'  C0               880           RET   NZ
0419'  06 01            881           LD    B,1
041B'  0E 01            882           LD    C,01H
041D'  C9               883           RET
041E'  AF               884   SET0    XOR   A
041F'  47               885           LD    B,A
0420'  C9               886           RET
                        887   !
                        888   !
                        889                                   ! ROUTINES TO ENABLE CHUNK 0 IN ROM EXT.
                        890                                   ! FOR ACCESS TO CHNGVID ROUTINE
0421'  F5              891   ENBLEXT PUSH  AF
0422'  DB FF            892           IN    A,(MREXPT)
0424'  F6 08            893           OR    08H         ; SELECT ROM EXT.
0426'  D3 FF            894           OUT   (MREXPT),A
0428'  3E 03            895           LD    A,3
042A'  D3 F4            896           OUT   (DKMSPT),A
042C'  F1               897           POP   AF
042D'  C9               898           RET
                        899   !
042E'  F5               900   ENBLMEM  PUSH  AF
042F'  AF               901            XOR   A
0430'  D3 F4            902            OUT   (DKMSPT),A
0432'  DB FF            903            IN    A,(MREXPT)
0434'  E6 3F            904            AND   03FH
0436'  D3 FF            905            OUT   (MREXPT),A
0438'  F1               906            POP   AF
0439'  C9               907            RET
                        908   !
                        909                                   ! TEST PARAMETER VALIDITY
                        910                                   ! IF NOT VALID, DISCARDS RETURN
                        911                                   ! AND EXITS VIA INVPAR
043A'  3E 17            912   TSTPAR  LD    A,23            ! TEST LINE >23
043C'  B8               913           CP    B
043D'  30 04            914           JR    NC,TSTPR1
043F'  F1               915   PARERR  POP   AF
0440'  C3 00C2'         916           JP    INVPAR
0443'  3A 0033'         917   TSTPR1  LD    A,(LINLEN)
0446'  3C               918           INC   A
0447'  B9               919           CP    C           ; TEST COL. >63
0448'  38 F8            920           JR    C,PARERR
044A'  C9               921           RET
                        922   !                               ! CONVERT LINE/COL. FROM USER TO
                        923                                   ! INTERNAL FORMAT AND VICE VERSA
                        924                                   ! USER FORMAT:  LINES 0-23
                        925                                   !                 COL.0-63
                        926                                   ! INTERNAL FORMAT: LINES 24-1
                        927                                   !                    COL. 65-2
                        928                                   !                  (COL.1 END OF LINE)
0448'  3A 0033'         929   CONVFM  LD    A,(LINLEN)
044B'  3C               930           INC   A
044C'  91               931           SUB   C
044D'  4F               932           LD    C,A
044E'  3E 18            933           LD    A,SCRSZ
0450'  90               934           SUB   B
0451'  47               935           LD    B,A
0452'  C9               936           RET
                        937   !
                        938   !
                        939   !
0453'                   940   UPDPOSN                         ! ENTER HERE WITH BC=LINE/COL.IN
                        941                                   ! INTERNAL FORMAT
0453'  CD 0458'         942           CALL    CALCPOS         ! CALCULATE POSITION
0456'  18 2A            943           JR      STPOSN          ! STORE UPDATED POSITION AND RETURN
                        944   !
                        945   !
                        946   !                               ! ROUTINE TO CALCULATE POSITION IN DP
```

```
                                947                             ; RETURNS DP ADRS. IN HL.  PRESERVES
                                948                             ; LINE/COLUMN POSITION IN BC
                                949     ;
0458' CD 0474'                  950     CALCPOS CALL    LNBU            ; GET DISPLAY FILE ADRS.
045B' 3A 0033'                  951             LD      A,(LINLEN)      ; TEST WHICH DP
045E' 3C                        952             INC     A
045F' 91                        953             SUB     C
0460' CB 47                     954             BIT     0,A
0462' 28 06                     955             JR      Z,CALCP1
0464' F5                        956             PUSH    AF
0465' 3E 20                     957             LD      A,20H
0467' B4                        958             OR      H               ; IN DP2
0468' 67                        959             LD      H,A
0469' F1                        960             POP     AF
046A' A7                        961     CALCP1  AND     A
046B' 1F                        962             RRA                     ; COL/2 IS POSITION IN DISPLAY FILE
046C' 5F                        963             LD      E,A
046D' 16 00                     964             LD      D,0
046F' 19                        965             ADD     HL,DE
0470' C9                        966             RET
                                967     ;
                                968     ;
0471'                           969     LNBU:                           ; GET DISPLAY FILE ADDRESS
                                970                                     ; FOR START OF LINE IN B
0471' 3E 18                     971             LD  A,SCRSZ
0473' 90                        972             SUB  B
0474' 57                        973             LD   D,A
0475' 0F                        974             RRCA
0476' 0F                        975             RRCA
0477' 0F                        976             RRCA
0478' E6 E0                     977             AND  0E0H
047A' 6F                        978             LD   L,A
047B' 7A                        979             LD   A,D
047C' E6 18                     980             AND  18H
047E' F6 40                     981             OR   40H
0480' 67                        982             LD   H,A
0481' C9                        983             RET
                                984
0482'                           985     STPOSN: :                       ; STORE CURSOR POSITION
0482' ED 43 0034'               986             LD  (CURPOS),BC
0486' 22 0036'                  987             LD  (DPADRS),HL
0489' C9                        988             RET
                                989     ;
048A'                           990     LDPOSN: :                       ; LOAD CURSOR POSITION
048A' ED 4B 0034'               991             LD  BC,(CURPOS)
048E' 2A 0036'                  992             LD  HL,(DPADRS)
0491' C9                        993             RET
                                994     ;
                                995     ;
                                996     ;
                                997     ;
                                999             SUBTTL GRAPHICS CHAR.SET
                               1000     ;
                               1001     ;                              ;PIXEL PATTERNS FOR STD.GRAPHICS SET
                               1002     ;                              ; 0=BACKGROUND(PAPER) 1=FOREGROUND(INK)
                               1003     ;
0492' 00                       1004     GRPST   defb 00000000b          ; code 80       graphics space
0493' 00                       1005             defb 00000000b
0494' 00                       1006             defb 00000000b
0495' 00                       1007             defb 00000000b
0496' 00                       1008             defb 00000000b
0497' 00                       1009             defb 00000000b
0498' 00                       1010             defb 00000000b
0499' 00                       1011             defb 00000000b
                               1012     ;
049A' 0F                       1013             defb 00001111b          ; code 81       graphics
049B' 0F                       1014             defb 00001111b
049C' 0F                       1015             defb 00001111b
049D' 0F                       1016             defb 00001111b
049E' 00                       1017             defb 00000000b
049F' 00                       1018             defb 00000000b
04A0' 00                       1019             defb 00000000b
04A1' 00                       1020             defb 00000000b
                               1021     ;
                               1022
04A2' F0                       1023             defb 11110000b          ; code 82       graphics
04A3' F0                       1024             defb 11110000b
04A4' F0                       1025             defb 11110000b
04A5' F0                       1026             defb 11110000b
04A6' 00                       1027             defb 00000000b
04A7' 00                       1028             defb 00000000b
04A8' 00                       1029             defb 00000000b
04A9' 00                       1030             defb 00000000b
                               1031
04AA' FF                       1032             defb 11111111b          ; code 83       graphics
04AB' FF                       1033             defb 11111111b
04AC' FF                       1034             defb 11111111b
04AD' FF                       1035             defb 11111111b
04AE' 00                       1036             defb 00000000b
04AF' 00                       1037             defb 00000000b
04B0' 00                       1038             defb 00000000b
04B1' 00                       1039             defb 00000000b
                               1040
                               1041     ;
04B2' 00                       1042             defb 00000000b          ; code 84       graphics
04B3' 00                       1043             defb 00000000b
04B4' 00                       1044             defb 00000000b
04B5' 00                       1045             defb 00000000b
04B6' 0F                       1046             defb 00001111b
04B7' 0F                       1047             defb 00001111b
04B8' 0F                       1048             defb 00001111b
04B9' 0F                       1049             defb 00001111b
                               1050     ;
```

APPENDIX C-2

80 COLUMN MODE

Date: 12/16/83

ASC Number: 002

Version: 001

Author: C. Corcoran/D. 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 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 (ED00H/57344).

- -

Application Services

- -

Name: SETMODE (SETM0B 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 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: 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 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 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 - Contains 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 Edit 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 7800H-7AFFH (30720-31487) is not used by the video mode H/W in 80-Column mode.

```
                                1           NAME AGL - ASC 002 80-CCL.MODE SUPPORT
                                2   ;
                                3   ;
                                4   ;
                                5   ;
                                6   ;
                                7   ;
                                8   ;
                                9   ;           ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               10   ;           ;                                   ;
                               11   ;           ;              TIMEX                ;
                               12   ;           ; APPLICATION DEVELOPMENT LIBRARY   ;
                               13   ;           ;                                   ;
                               14   ;           ; NAME: 80-CCL.MODE SUPPORT         ;
                               15   ;           ;                                   ;
                               16   ;           ; ASC NO: 002                       ;
                               17   ;           ; VERSION: 001                      ;
                               18   ;           ; AUTHOR: C. CORCORAN/D. BOYLE      ;
                               19   ;           ; DATE:   12/14/83                  ;
                               20   ;           ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                               21
                               22           SUBTTL DEFINITIONS
                               23   ;
                               24   ;
                               25   ;                       ;;;;;DEFINITIONS;;;;;
                               26   ;
                               27           INTERN  WRCHAR, WRSTRG, SETCUR, SETATT, SETMSK, CLRSCN, SCRCLL
                               28           INTERN  GTCHAR, GETATT, GETCUR
                               29   ;
                               30           INTERN  WRCHRB, WRSTRB, SETCUB, SETATB, SETMSB, CLRSCB, SCRLB
                               31           INTERN  GTCHRB, GETATB, GETCUB
                               32   ;
                               33           INTERN  SETMODE, SETMOB
                               34   ;
                               35   ;
0018                           36   SCRSZ       EQU     24      ; 24 LINES
1701                           37   SCINIT      EQU     1701H   ; SCROLL CONTROL INITIALIZATION
1800                           38   CLINIT      EQU     1800H   ; CLEAR SCREEN CTL. INIT.
0478'                          39   CHRSET      EQU     ((CHRST)-100H)  ;CHAR.TABLE-100H
0778'                          40   GRPHST      EQU     ((GRPST)-100H) ; GRAPHICS TABLE
                               41   ;
0E08                           42   CHNGVID     EQU     0E08H   ; ROUTINE IN ROM EXTENSION
1720                           43   RECLEN      EQU     1720H   ; ROUTINE IN HOME ROM
                               44   ;
00FF                           45   HOMEXPT     EQU     0FFH    ; HOME ROM EXT.SELECT PORT (BIT 7)
00F4                           46   DOCMSPT     EQU     0F4H    ; DOCK HORIZONTAL SELECT PORT
                               47   ;
5C48                           48   VARS        EQU     5C4BH   ; SYSTEM VARIABLES
5C65                           49   STKEND      EQU     5C65H   ;
5C78                           50   UDG         EQU     5C7BH   ;
5C7D                           51   COORDS      EQU     5C7DH   ;
5C82                           52   PRAMT       EQU     5C82H   ;
5C84                           53   PRAMT       EQU     5C84H   ;
5CC2                           54   VIDMOD      EQU     5CC2H   ;
5CC3                           55   PARAMS      EQU     5CC3H   ; (Location 23747)
4840                           56   DRIVES      EQU     4840H   ;
1FC8                           57   INSERTSZ    EQU     7800H-DRIVES
0840                           58   MOVESZ      EQU     DRIVES-6000H
07C8                           59   DEST7       EQU     0FFFFH-MOVESZ+.
47C8                           60   PIX         EQU     DEST7-4800H
                               61
1000                           62   PIXTBL      EQU     1000H
                               63
                               64
                               65
                               66   ;
                               67   ;
                               68   ;
                               69   ;           ;;PARAMETER INPUTS AND ENTRY POINTS FROM BASIC;;
                               70   ;                                                                   WITH
                               71   ;                                                                MODULE LOADED
                               72   ;                                                                AT 57344(E000H)
                               73   ;                                                                DECIMAL ADDRESS
                               74   ;                                                                ---------------
                               75   ;
0000' 20                       76   DATAB       DEFB    20H     ; CHAR.CODE FROM BASIC              57344
0001' C3 003F'                 77   WRCHRB:     JP      WRCHD   ; ENTRY TO WRITE CHARACTER          57345
0004' C3 0171'                 78   WRSTRB:     JP      WRSTRD  ; ENTRY TO WRITE STRING             57348
                               79   ;                           ; (see PARAMS above)
0007' 0000                     80   LINCCL      DEFW    0       ; SET CURSOR PARAMETERS
                               81                               ;   LINCCL=COL. (0-79)              57351
                               82                               ;   LINCOL+1=LINE (0-23)            57352
0009' C3 0184'                 83   SETCUB:     JP      SETCBD  ; ENTRY TO SET CURSOR POSN.         57353
000C' 1800                     84   CLRCTL      DEFW    1800H   ; CLEAR SCREEN CONTROLS
                               85                               ;   CLRCTL=STARTING LINE NO.       57356
                               86                               ;   CLRCTL+1=LINE COUNT           57357
000E' C3 0278'                 87   CLRSCB:     JP      CLRSBD  ; ENTRY TO CLEAR SCREEN             57358
0011' 38                       88   ATTCTL      DEFB    38H     ; ATTRIBUTE CONTROL                 57361
                               89                               ;   BIT 7=FLASH (0 IN 80-COL.)
                               90                               ;       6=BRIGHT(0 IN 80-COL.)
                               91                               ;       5
                               92                               ;       4=PAPER COLOR
                               93                               ;       3
                               94                               ;       2
                               95                               ;       1= INK COLOR
                               96                               ;       0
0012' C3 02F3'                 97   SETATB:     JP      SETABD  ; ENTRY TO SET ATTRIBUTES           57362
0015' 00                       98   MSKCTL      DEFB    0       ; MASK CONTROL FROM BASIC:          57365
                               99                               ;   BIT 2=INVERSE
                              100                               ;       0=OVER
0016' C3 031D'                101   SETMSB:     JP      SETMBD  ; ENTRY TO SET MASK CONTROLS        57366
0019' 8C00                    102   GETCTL      DEFW    0       ; 'GET' CONTROL FROM BASIC          57369
                              103                               ; FORMAT IS AS FOR LINCOL
001B' C3 0326'                104   GTCHRB:     JP      GTCHBD  ; ENTRY TO GET CHARACTER            57371
001E' C3 03C0'                105   GETATB:     JP      GETABD  ; ENTRY TO GET ATTRIBUTE            57374
0021' C3 03D4'                106   GETCUB:     JP      GETCBD  ; ENTRY TO GET CURSOR POSN.         57377
0024' 00                      107   VIMODE      DEFB    0       ; VIDEO MODE CONTROL                57380
                              108                               ;       0=NORMAL
                              109                               ;       8=80-CCL.MODE
0025' C3 03F1'                110   SETMDB:     JP      STMDBD  ; ENTRY TO SET VIDEO MODE           57381
0028' 1701                    111   SCRCTL      DEFW    1701H   ; SCROLL CONTROL
                              112                               ;   SCRCTL=STARTING LINE NO.       57384
                              113                               ;   SCRCTL+1=LINE COUNT           57385
002A' C3 01CA'                114   SCRLB:      JP      SCRLBD  ; ENTRY TO SCROLL                   57386
```

```
                                116             SUBTTL  OTHER VARIABLES
                                117     ;
0020'  17                       118     BOTLN           DEFB    17H     ; BOTTOM LINE CLINE AFTER       97389
                                119                                     ; WHICH SCREEN IS CONSIDERED
                                120                                     ; FULL.  AUTOMATIC SCROLL
                                121                                     ; WILL BE DONE IF SCRLCT
                                122                                     ; DOES NOT DECREMENT TO 0.
0021'  01                       123     SCRLCT          DEFB    1       ; SCROLL COUNT - 1 PLUS NO.     97390
                                124                                     ; OF TIMES AUTOMATIC SCROLL
                                125                                     ; WILL BE DONE.
0022'  0478'                    126     CNTBL           DEFW    CHRSET  ; ADDRESS OF CHARACTER TABLE    97391
0024'  0778'                    127     GRTBL           DEFW    GRPHST  ; ADDRESS OF STD.GRAPHICS TBL.  97393
0026'  50                       128     LINLEN          DEFB    50H     ; LINE LENGTH(80 OR 85)         97395
0027'  1891                     129     CURPOS          DEFW    1891H   ; CURRENT POSITION              97396
                                130     ;                               ; (INTERNAL FORMAT)
0029'  4000                     131     DPADRS          DEFW    4000H   ; CURRENT DISPLAY FILE ADDRS.   97398
002B'  00                       132     MASK0           DEFB    0       ; MASK TO BE APPLIED TO         97400
                                133     ;                               ; DISPLAY CHARACTERS
002C'  38                       134     ATTRVT          DEFB    38H     ; CURRENT ATTRIBUTES            97401
                                135     ;                               ; (38=BLACK ON WHITE)
002D'  00                       136     GTINDX          DEFB    00H     ; INDEX FOR ADJUSTING           97402
                                137     ;                               ; CH.CODE IN GTCHAR
002E'  0000                     138     STRGCT          DEFW    0H      ; REMAINING COUNT FROM WRSTRG
                                139     ;                               ; WHEN "SCREEN FULL"            97403
0030'  01                       140     MARGIN          DEFB    1       ; MARGIN ADJUST (0-2)           97405
0031'  07                       141     DPBIT           DEFB    7       ; BIT POSN.IN LINE(7,1,3,5)     97406
                                142             SUBTTL  WRITE CHARACTER
                                143     ;
003F'                           144     WRCH0I                          ; HERE FROM BASIC ENTRY TO DISPLAY THE
                                145                                     ; CHARACTER WHOSE CODE IS IN "DATA0"
003F'  3A 0000'                 146             LD      A,(DATA0)
                                147     ;
0042'                           148     WRCHAR1                         ; ENTRY WITH CODE IN A
0042'  CD 0598'                 149             CALL    LDPOSN          ; LOAD REGISTERS FOR CURRENT POSITION
                                150                                     ;   BC=CURSOR POSITION FROM "CURPOS"
                                151                                     ;   HL=DISPLAY FILE ADDRESS FROM "DPADRS
                                152                                     ;   A=CODE FOR CHAR. TO BE DISPLAYED
0045'  FE 20                    153             CP      20H             ; TEST VALID RANGE
0047'  DA 0147'                 154             JP      C,INVPAR        ; < 20H
004A'  FE A5                    155             CP      0A5H            ;
004C'  D2 0147'                 156             JP      NC,INVPAR       ; > A4H
004F'  C5                       157             PUSH    BC              ; SAVE CURSOR POSITION
0050'  FE 80                    158             CP      80H             ; TEST IF ASCII PRINTABLE (20H-7FH)
0052'  38 15                    159             JR      C,WRCH12        ; YES
0054'  FE 10                    160             CP      90H             ; TEST IF STD.GRAPHICS(80-8FH)
0056'  38 09                    161             JR      C,WRCH11
0058'  ED 4B 9C78               162             LD      BC,(UDG)        ; USER -DEFINED GRAPHICS(90-A4H)
005C'  05                       163             DEC     B               ; ADJUST ADDRESS-100H
005D'  D6 70                    164             SUB     70H             ; ADJUST FOR INDEX INTO TABLE
005F'  18 0C                    165             JR      WRCH13
0061'  ED 4B 0031'              166     WRCH11  LD      BC,(GRTBL)      ; STD. GRAPHICS TABLE
0065'  D6 60                    167             SUB     60H             ; ADJUST FOR INDEX INTO TABLE
0067'  18 04                    168             JR      WRCH13
0069'  ED 4B 003F'              169     WRCH12  LD      BC,(CNTBL)      ; CHARACTER TABLE
006D'  E8                       170     WRCH13  EX      DE,HL           ; DE=POSN.IN DISPLAY FILE
006E'  26 00                    171             LD      H,0
0070'  6F                       172             LD      L,A             ; CODE IS INDEX INTO TABLE
0071'  29                       173             ADD     HL,HL
0072'  29                       174             ADD     HL,HL
0073'  29                       175             ADD     HL,HL
0074'  09                       176             ADD     HL,BC           ; HL=PIXEL PATTERN IN TABLE
0075'  C1                       177             POP     BC
0076'  E8                       178             EX      DE,HL           ; DE=PIXEL PATTERN IN TABLE/HL=DP
0077'  79                       179             LD      A,C             ; TEST IF END OF LINE
0078'  3D                       180             DEC     A
0079'  3A 0033'                 181             LD      A,(CLINLEN)
007C'  20 05                    182             JR      NZ,WRCH14
007E'  3C                       183             INC     A               ; START OF NEW LINE
007F'  05                       184             DEC     B               ; BUMP LINE NO.
0080'  4F                       185             LD      C,A             ; LINLEN+1=START OF LINE
0081'  18 01                    186             JR      WRCH15
0083'  3C                       187     WRCH14  INC     A
0084'  B9                       188     WRCH15  CP      C               ; TEST IF START OF LINE
0085'  C5                       189             PUSH    BC
0086'  CC 0148'                 190             CALL    Z,TVPULB        ; TEST IF PAST "BOTTOM" LINE
0089'  01                       191             POP     BC
                                192     ;                               ; HERE TO PUT CHARACTER IN DISPLAY FILE
008A'  C5                       193     WRCH2   PUSH    BC              ; CURSOR POSITION
008B'  E5                       194             PUSH    HL              ; DP ADDRESS
008C'  3A 0038'                 195             LD      A,(MASK8)       ; GET OVER AND INVERSE CONTROLS
008F'  06 FF                    196             LD      B,-1
0091'  1F                       197             RRA
0092'  38 01                    198             JR      C,WRCH3         ; IF XORING NEW INTO OLD
0094'  04                       199             INC     B               ; B=-1 IF XORING  =0 IF NOT
0095'  1F                       200     WRCH3   RRA
0096'  1F                       201             RRA
0097'  9F                       202             SBC     A,A
0098'  4F                       203             LD      C,A             ; C=-1 FOR INVERSE  =0 IF NOT
0099'  3E 08                    204             LD      A,8             ; SCAN COUNT
009B'  08                       205     WRCH5   EX      AF,AF'          ; SAVE SCAN COUNT IN ALTERNATE
009C'  C5                       206             PUSH    BC              ; SAVE MASK VALUES
009D'  D5                       207             PUSH    DE              ; SAVE CHAR.TABLE ADRS
009E'  DD 21 0000               208             LD      IX,0            ; SAVE PTR.TO STACK
00A2'  DD 39                    209             ADD     IX,SP
00A4'  56                       210             LD      D,(HL)          ; READ SCAN LINE FROM DP
00A5'  E5                       211             PUSH    HL              ; SAVE ADDRESS
00A6'  7C                       212             LD      A,H
00A7'  EE 20                    213             XOR     20H             ; TOGGLE TO ALTERNATE DP
00A9'  67                       214             LD      H,A
00AA'  CB 6F                    215             BIT     5,A             ; TEST WHICH DP
00AC'  20 01                    216             JR      NZ,WRCH6        ; IN DP1
00AE'  23                       217             INC     HL              ; NEXT BYTE
00AF'  5E                       218     WRCH6   LD      E,(HL)          ; READ SCAN LINE FROM ALTERNATE DP
0000'  E8                       219             EX      DE,HL           ; SCANS TO HL. LAST DP POS.IN DE
0001'  7B                       220             LD      A,E             ; TEST IF "OVER" IS ACTIVE
0002'  A7                       221             AND     A
0003'  20 16                    222             JR      NZ,WJXOR        ; YES-RETAIN OLD DATA
0005'  3A 0038'                 223             LD      A,(DPBIT)       ; NO- CLEAR OLD DATA AT CHAR.POSN.
0008'  06 07                    224             SUB     7               ;
000A'  80 44                    225             NEG
000C'  01 03FF                  226             LD      BC,03FFH        ; MASK
                                227
```

```
00BF' 28 08              220          JR    Z,NORR1          ; NO NEED TO ADJUST(CHAR.STARTS AT BIT 7)
00C1' 37                 221          SCF
00C2' CB 18              222  NEXT1   RR    B                ; SHIFT MASK 'A' TIMES
00C4' CB 19              223          RR    C                ; (6 0'S START AT BIT 1 OF B, BIT 3 OF B,
00C6' 3D                 224          DEC   A                ;  OR BIT 5 OF B)
00C7' 20 F9              225          JR    NZ,NEXT1
00C9' 7C                 226  NORR1   LD    A,H              ; SCAN LINE FROM DF
00CA' A0                 227          AND   B
00CB' 67                 228          LD    H,A              ; RETAIN ADJACENT CHAR.PIXELS
00CC' 7D                 229          LD    A,L              ; CLEAR CURRENT CHAR.PIXELS
00CD' A1                 230          AND   C
00CE' 6F                 231          LD    L,A              ; HL=OLD BIT ROW
00CF' 3A 0038'           232  NOXOR   LD    A,(DFBIT)        ; GET STARTING BIT POSN.
0002' D6 07              233          SUB   7
0004' ED 44              234          NEG
0006' E5                 235          PUSH  HL               ; SAVE SCANS
0007' DD 6E 00           236          LD    L,(IX)           ; PTR. TO CHAR.SET
000A' DD 66 01           237          LD    H,(IX+1)
000D' 46                 238          LD    B,(HL)           ; SCAN LINE FROM CHAR.SET TO BC
00DE' 0E 00              239          LD    C,0              ; (PIXELS IN B, BITS 7-2)
00E0' F5                 240          PUSH  AF               ; SAVE A AND FLAGS
00E1' 78                 241          LD    A,B              ; PIXELS FROM CHAR. SET
00E2' E6 FC              242          AND   0FCH             ; LIMIT TO 6 PIXELS
00E4' 47                 243          LD    B,A              ;
00E5' F1                 244          POP   AF               ; RESTORE A AND FLAGS
00E6' E1                 245          POP   HL               ; RESTORE SCANS
00E7' 28 08              246          JR    Z,NORR2          ; NO SHIFT NEEDED
00E9' A7                 247          AND   A                ; CLEAR CARRY
00EA' CB 18              248  NEXT2   RR    B                ; SHIFT 'A' TIMES
00EC' CB 19              249          RR    C
00EE' 3D                 250          DEC   A
00EF' 20 F9              251          JR    NZ,NEXT2
00F1' 7C                 252  NORR2   LD    A,H              ; SCAN FROM DF
00F2' A8                 253          XOR   B                ; INSERT/COMBINE NEW CHAR.
00F3' 67                 254          LD    H,A
00F4' 7D                 255          LD    A,L
00F5' A9                 256          XOR   C
00F6' 6F                 257          LD    L,A              ; DF XOR CH.SET->HL
00F7' DD 7E 02           258          LD    A,(IX+2)         ; TEST IF INVERSE ACTIVE
00FA' A7                 259          AND   A
00FB' 28 14              260          JR    Z,NOINVERT
00FD' 3A 003E'           261          LD    A,(DFBIT)
0100' D6 07              262          SUB   7
0102' ED 44              263          NEG
0104' 01 FC00            264          LD    BC,0FC00H        ; MASK TO INVERT
0107' 28 08              265          JR    Z,NORR3
0109' A7                 266          AND   A
010A' CB 18              267  NEXT3   RR    B                ; SHIFT MASK 'A' TIMES
010C' CB 19              268          RR    C
010E' 3D                 269          DEC   A
010F' 20 F9              270          JR    NZ,NEXT3
0111' 7C                 271  NORR3   LD    A,H
0112' A8                 272          XOR   B                ; INVERT CHARACTER
0113' 67                 273          LD    H,A
0114' 7D                 274          LD    A,L
0115' A9                 275          XOR   C
0116' 6F                 276          LD    L,A
0117' EB                 277  NOINVERT EX    DE,HL            ; DF ADRS. TO HL-SCAN LINE IN DE
0118' 73                 278          LD    (HL),E           ; WRITE SCAN LINE TO DF
0119' E1                 279          POP   HL               ; ALTERNATE DF
011A' 72                 280          LD    (HL),D           ; WRITE SCAN LINE TO DF
011B' D1                 281          POP   DE               ; CHAR.SET
011C' C1                 282          POP   BC               ; OVER/INVERSE INDICATORS
011D' 08                 283          EX    AF,AF'           ; RESTORE SCAN COUNT
011E' 24                 284          INC   H                ; NEXT SCAN IN DF
011F' 13                 285          INC   DE               ; NEXT BYTE IN CHAR.FILE
0120' 3D                 286          DEC   A                ; TEST IF MORE SCANS
0121' C2 0098'           287          JP    NZ,WRCHS
0124' E1                 288  WRCHT   POP   HL               ; TOP SCAN IN DF
0125' C1                 289          POP   BC               ; CURSOR POSITION
0126' 0D                 290          DEC   C                ; ADJUST CURSOR POSITION
0127' 3A 003E'           291          LD    A,(DFBIT)        ; ADJUST BIT POSITION
012A' D6 06              292          SUB   6
012C' 32 003E'           293          LD    (DFBIT),A        ; STORE UPDATED POSITION
012F' 30 0E              294          JR    NC,WRCHXT        ; NEXT CHAR.IN SAME BYTE
0131' C6 08              295          ADD   A,8              ; NEXT CHAR.IN NEXT BYTE
0133' 32 003E'           296          LD    (DFBIT),A        ; STORE ADJUSTED VALUE
0136' 7C                 297          LD    A,H              ; ADJUST DF ADRS TO NEXT BYTE
0137' EE 20              298          XOR   20H              ; TOGGLE TO ALTERNATE DF
0139' 67                 299          LD    H,A
013A' CB 6F              300          BIT   5,A              ; TEST WHICH DF
013C' 20 01              301          JR    NZ,WRCHXT        ; DONE IF DF2
013E' 23                 302          INC   HL               ; NEXT BYTE IN DF1
013F' CC 0950'           303  WRCHXT  CALL  STPOSN           ; STORE NEW POSITION
0142' 0E 00              304  GOODRET LD    C,0              ; RETURN BC=0
0144' 06 00              305  ERRRET  LD    B,0              ; ENTER HERE WITH C NON-ZERO
0146' C9                 306          RET
                         307  ;
0147' 0E 01              308  INVPAR  LD    C,1              ; RETURN BC=1 FOR INVALID PARAMETERS
0149' 18 F9              309          JR    ERRRET
                         310  ;
014B' 3E 18              311  TVPUL0  LD    A,SCRSZ          ; TEST IF NEW LINE IS OFF SCREEN
014D' 90                 312          SUB   B                ; A=LINE NO.
014E' 57                 313          LD    D,A              ; SAVE IN D
014F' 3A 002D'           314          LD    A,(BOTLN) ; GET BOTTOM LINE
0152' BA                 315          CP    D
0153' D2 0507'           316          JP    NC,UPDPOSN       ; ON SCREEN - RETURN TO WRITE CHAR.
                         317                                 ;  VIA UPDATE AND STORE POSITION
0156' 3A 0028'           318          LD    A,(SCRLCT)       ; TEST SCROLL COUNT
0159' 3C                 319          DEC   A
015A' 28 08              320          JR    NZ,TVPUL1        ; DO AUTOMATIC SCROLL USING SCRCTL
015C' 3E 01              321          LD    A,1              ; REINITIALIZE TO 1
015E' 32 0028'           322          LD    (SCRLCT),A
0161' C1                 323          POP   BC
0162' C1                 324          POP   BC               ; DISCARD RETURN TO WRITE CHAR.
0163' 0E 03              325          LD    C,3              ; RETURN BC=3 FOR SCREEN FULL
0165' 18 DD              326          JR    ERRRET
0167' 31 0028'           327  TVPUL1  LD    (SCRLCT),A       ; UPDATE VARIABLE
016A' ED 4B 0029'        328          LD    BC,(SCRCTL)      ; GET SCROLL CONTROLS (STARTING LINE
                         329                                 ;  AND LINE COUNT)
016E' C3 0184'           330          JP    SCRL0            ; RETURN TO WRITE CHAR. VIA SCROLL
                         331  )
```

```
                                341             SUBTTL WRITE STRING
                                342     ;
                                343     ;                               ; HERE FROM BASIC ENTRY TO
                                344     ;                               ; WRITE CHARACTER STRING TO SCREEN.
                                345     ;                               ; STRING IDENTIFIER (CH.CODE) IN
                                346     ;                               ; SYSTEM VARIABLE PARAMS AT 5CC3H
0171' 3A 5CC3                   347     WRSTR0  LD      A,(PARAMS)
0174' E6 1F                     348             AND     1FH             ; MASK OFF UPPER BITS
0176' F6 40                     349             OR      40H             ; STRING VARIABLE IDENTIFIER
0178' 57                        350             LD      D,A             ; SAVE IN D
0179' 2A 5C4B                   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 35                     354             JR      Z,NOSTRG        ; NO FIND - RETURN BC=2
0181' BA                        355             CP      D               ; TEST IF MATCH
0182' 28 08                     356             JR      Z,WRSTR2        ; FOUND STRING
0184' D5                        357             PUSH    DE              ; SAVE STRING ID
0185' CD 1720                   358             CALL    RECLEN          ; RETURNS ADRS.OF NEXT VAR.IN DE
0188' EB                        359             EX      DE,HL           ; ADRS. TO HL
0189' D1                        360             POP     DE              ; RESTORE STRING ID
018A' 18 F0                     361             JR      WRSTR1
018C' 23                        362     WRSTR2  INC     HL              ; GET LENGTH
018D' 4E                        363             LD      C,(HL)
018E' 23                        364             INC     HL
018F' 46                        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' 7E                        375     WRSTRG  LD      A,(HL)          ; GET CODE
0195' E5                        376             PUSH    HL              ; SAVE ADRS. AND
0196' C5                        377             PUSH    BC              ; COUNT
0197' CD 0042'                  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 EE                     389             JR      NZ,WRSTRG       ; WRITE NEXT CHAR.
01A6' C9                        390             RET                     ; RETURN BC=0
                                391     ;
01A7' 79                        392     WRSTX1  LD      A,C             ; SAVE ERR#?
01A8' FE 01                     393             CP      1               ; TEST IF UNRECOGNIZED CODE
                                394     ;
01AA' 28 F2                     395             JR      Z,WRSTR3        ; CONTINUE
01AC' E1                        396             POP     HL              ; REMAINING COUNT TO HL
01AD' 22 0039'                  397             LD      (STRGCT),HL     ; STORE REMAINING COUNT
01B0' E1                        398             POP     HL              ; ADDRESS TO HL
01B1' 0E 03                     399             LD      C,3             ; RETURN BC=3 FOR SCREEN FULL
01B3' 06 00                     400     WRSTX2  LD      B,0             ;
01B5' C9                        401             RET
                                402     ;
01B6' 0E 02                     403     NOSTRG  LD      C,2             ; RETURN BC=2 IF STRING NOT FOUND
01B8' 18 F9                     404             JR      WRSTX2
                                405     ;
                                406     ;
                                407     ;
                                408             SUBTTL POSITION CONTROL
                                409     ;
01BA'                           410     SETCB0: ;                       ; HERE FROM BASIC ENTRY.  PARAMETERS
                                411     ;                               ; IN LINCOL
01BA' ED 4B 0007'               412             LD   BC,(LINCOL)
                                413     ;
01BE'                           414     SETCUR: ;                       ; ENTRY WITH LINE/COL. IN BC REG.
01BE' CD 0488'                  415             CALL   TSTPAR           ; TEST PARAMETERS FOR VALIDITY
01C1' CD 04FC'                  416             CALL   CONVFM           ; CONVERT TO INTERNAL FORMAT
01C4' CD 0507'                  417             CALL   UPCPOSN          ; CALCULATE AND STORE POSITION
01C7' C3 0142'                  418             JP     GOODRET          ; RETURN BC=0
                                419     ;
                                420             SUBTTL SCROLL
                                421     ;
                                422     ;
01CA'                           423     SCRLBG: ;                       ; HERE FROM BASIC ENTRY TO SCROLL
                                424     ;                               ; SCREEN
01CA' ED 4B 0026'               425             LD   BC,(SCRCTL)        ; GET CONTROL INFO.
                                426     ;
01CE'                           427     SCROLL: ;                       ; HERE WITH CONTROLS IN BC
                                428     ;                               ;  B=NO. OF LINES
                                429     ;                               ;  C=STARTING LINE NO.
01CE' 78                        430             LD   A,B                ; TEST VALIDITY
01CF' A7                        431             AND  A                  ;
01D0' CA 0147'                  432             JP   Z,INVPAR           ; ERROR IF COUNT=0
01D3' 81                        433             ADD  A,C
01D4' FE 01                     434             CP   1
01D6' DA 0147'                  435             JP   C,INVPAR           ; ERROR IF B+C<1
01D9' FE 19                     436             CP   25
01DB' D2 0147'                  437             JP   NC,INVPAR          ; ERROR IF B+C>24
01DE' CD 01E4'                  438             CALL  SCRL0             ; DO SCROLL
01E1' C3 0142'                  439             JP    GOODRET           ; RETURN BC=0
                                440     ;
01E4' C5                        441     SCRL0   PUSH  BC
01E5' 3E 18                     442             LD    A,SCRSZ           ; GET STARTING LINE IN INTERNAL FORMAT
01E7' 91                        443             SUB   C
01E8' 47                        444             LD    B,A
01E9' 3A 0033'                  445             LD    A,(LINLEN)
01EC' 3C                        446             INC   A                 ; COL. 0
01ED' 4F                        447             LD    C,A
01EE' CC 033F'                  448             CALL  LNBU
01F1' C1                        449             POP   BC
01F2' C5                        450             PUSH  BC                ; SAVE ORIG.BC FOR EXIT
01F3' 7D                        451     TESTAD  LD    A,L               ; TEST IF AT START OF BLOCK
01F4' A7                        452             AND   A
01F5' 28 01                     453             JR    Z,STBLK           ; YES
01F7' 07                        454             RLCA
01F8' 07                        455             RLCA
01F9' 07                        456             RLCA
```

```
01FA'  4F                457             LD    C,A               ; GET NO. OF LINES THIS BLOCK
01FB'  3E 00             458             LD    A,0
01FD'  91                459             SUB   C
01FE'  B8                460             CP    B                 ; TEST AGAINST TOTAL LINES
01FF'  38 3F             461             JR    C,GRBLK           ; TOTAL GREATER THAN THIS BLOCK
0201'  78                462  INSDBLK    LD    A,B
0202'  06 00             463             LD    B,0               ; REMAINING COUNT
0204'  C5                464             PUSH  BC
0205'  47                465  LCOP       LD    B,A               ; B=LINES THIS BLOCK
0206'  0E 08             466             LD    C,8               ; C=SCAN COUNT
0208'  78                467  LCOP0      LD    A,B
0209'  C5                468             PUSH  BC                ; SAVE SCAN COUNT
020A'  0F                469             RRCA
020B'  0F                470             RRCA
020C'  0F                471             RRCA                    ; CALCULATE NO. OF BYTES
020D'  4F                472             LD    C,A
020E'  06 00             473             LD    B,0               ; BC=32## OF LINES
0210'  E8                474  LOOP1      EX    DE,HL
0211'  21 FFE0           475             LD    HL,-20H           ; GET ADRS. OF PREV. LINE
0214'  19                476             ADD   HL,DE
0215'  E8                477             EX    DE,HL             ; TO DE
0216'  C5                478             PUSH  BC          ; SAVE COUNT
0217'  E5                479             PUSH  HL          ; SAVE ADDRESS
0218'  ED B0             480             LDIR              ; DO MOVE
021A'  E1                481             POP   HL
021B'  C1                482             POP   BC
021C'  7C                483             LD    A,H
021D'  CB 6F             484             BIT   5,A               ; TEST IF DP1
021F'  20 05             485             JR    NZ,NXTSC          ; NO
0221'  F6 20             486             OR    20H
0223'  67                487             LD    H,A               ; DO DP2
0224'  18 EA             488             JR    LOOP1
0226'  E6 DF             489  NXTSC     AND   0DFH              ; BACK TO DP1
0228'  67                490             LD    H,A
0229'  24                491             INC   H                 ; NEXT SCAN LINE
022A'  C1                492             POP   BC                ; SCAN COUNT
022B'  0D                493             DEC   C
022C'  20 DA             494             JR    NZ,LOOP0          ; MORE SCANS
022E'  C1                495             POP   BC                ; RESIDUAL LINE COUNT
022F'  78                496             LD    A,B
0230'  A7                497             AND   A
0231'  28 04             498             JR    Z,SCRLRT          ; DONE
0233'  2E 00             499             LD    L,0               ; START OF NEXT BLOCK
0235'  18 BC             500             JR    TESTAD            ; DO REMAINING LINE(S)
0237'  C1                501  SCRLRT     POP   BC                ; ORIG.BC
0238'  79                502             LD    A,C               ; STARTING LINE
0239'  80                503             ADD   A,B               ; ADD NO. OF LINES MOVED
023A'  3D                504             DEC   A                 ; LAST LINE SCROLLED
023B'  4F                505             LD    C,A
023C'  06 01             506             LD    B,1               ; CLEAR 1 LINE
023E'  18 92             507             JR    CLRSC0            ; CLEAR VACATED LINE AND RETURN
                         508  ;
0240'  4F                509  GRBLK      LD    C,A
0241'  78                510             LD    A,B
0242'  91                511             SUB   C                 ; ADJUST TOTAL LINE COUNT BY NO.
0243'  47                512             LD    B,A               ; OF LINES THIS BLOCK
0244'  C5                513             PUSH  BC
0245'  79                514             LD    A,C               ; LOAD A NO. OF LINES THIS BLOCK
0246'  18 BB             515             JR    LOOP
                         516  ;
0248'  05                517  STBLK      DEC   B                 ; ADJUST TOTAL LINE COUNT-1
0249'  C5                518             PUSH  BC
024A'  0E 08             519             LD    C,8               ; SCAN COUNT
024C'  C5                520  STBLK0     PUSH  BC                ; SAVE SCAN COUNT
024D'  EB                521  STBLK1     EX    DE,HL
024E'  21 F8E0           522             LD    HL,-720H
0251'  19                523             ADD   HL,DE
0252'  EB                524             EX    DE,HL             ; ADRS. OF PREV. LINE
0253'  01 0020           525             LD    BC,32             ; BYTE COUNT
0256'  E5                526             PUSH  HL
0257'  ED B0             527             LDIR                    ; DO MOVE
0259'  E1                528             POP   HL
025A'  7C                529             LD    A,H
025B'  CB 6F             530             BIT   5,A
025D'  20 05             531             JR    NZ,STBLK2
025F'  F6 20             532             OR    20H               ; DO DP2
0261'  67                533             LD    H,A
0262'  18 E9             534             JR    STBLK1
0264'  E6 DF             535  STBLK2     AND   0DFH              ; BACK TO DP1
0266'  67                536             LD    H,A
0267'  24                537             INC   H                 ; TO NEXT SCAN LINE
0268'  C1                538             POP   BC
0269'  0D                539             DEC   C                 ; TEST IF MORE SCANS
026A'  20 E0             540             JR    NZ,STBLK0
026C'  C1                541             POP   BC                ; REMAINING LINE COUNT
026D'  78                542             LD    A,B
026E'  A7                543             AND   A
026F'  28 C6             544             JR    Z,SCRLRT
0271'  11 F820           545             LD    DE,-7E0H          ; ADJUST TO LINE 1
0274'  19                546             ADD   HL,DE
0275'  C3 01F3'          547             JP    TESTAD            ; BC=LINE COUNT
                         548                                     ; HL=DP ADDRESS
                         549  ;
                         550             SUBTTL CLEAR SCREEN
                         551  ;
                         552  ;
0278'                    553  CLRS90I                            ; HERE FROM BASIC ENTRY TO CLEAR SCREEN
0278'  ED 4B 000C'       554             LD    BC,(CLRCTL)       ; GET CONTROL INFO.
                         555  ;
027C'                    556  CLRSCNI                            ; ENTRY WITH BC=LINE COUNT AND
                         557                                     ; STARTING LINE NO. FOR CLEAR
027C'  78                558             LD    A,B
027D'  A7                559             AND   A                 ; TEST VALIDITY
027E'  CA 0147'          560             JP    Z,INVPAR          ; ERROR IF COUNT=0
0281'  81                561             ADD   A,C
0282'  FE 01             562             CP    1
0284'  DA 0147'          563             JP    C,INVPAR          ; ERROR IF B+C<1
0287'  FE 19             564             CP    25
0289'  D2 0147'          565             JP    NC,INVPAR         ; ERROR IF B+C>24
```

```
028C' CD 0292'          967             CALL  CLRSC0            ; DO SERVICE
028F' C3 0142'          968             JP    GOODRET           ; RETURN BC=0
                        969     ;
                        970     ;
0292'                   971     CLRSC0  PUSH  BC
0293' 3E 18             972             LD    A,SCRSZ           ; CONVERT TO INTERNAL FORMAT
0295' 91                973             SUB   C
0296' 47                974             LD    B,A
0297' 3A 0033'          975             LD    A,(CLINLEN)
029A' 3C                976             INC   A                 ; SET TO COL.0
029B' 4F                977             LD    C,A
029C' CD 053F'          978             CALL  LNDU              ; GET ADDRESS
029F' 50                979             LD    D,B
02A0' 59                980             LD    E,C               ; SAVE POSITION
02A1' C1                981             POP   BC                ; ORIG. BC
02A2' D5                982             PUSH  DE
02A3' 7D                983     CLRSC1  LD    A,L               ; GET # OF LINES THIS BLOCK
02A4' 07                984             RLCA
02A5' 07                985             RLCA
02A6' 07                986             RLCA
02A7' 4F                987             LD    C,A
02A8' 3E 08             988             LD    A,8
02AA' 91                989             SUB   C
02AB' B8                990             CP    B                 ; COMPARE TO TOTAL LINE COUNT
02AC' 38 30             991             JR    C,CLRSC7
02AE' 78                992     CLRSC2  LD    A,B               ; NO. OF LINES
02AF' 06 00             993             LD    B,0               ; REMAINING LINES
02B1' C5                994             PUSH  BC
02B2' 47                995     CLRSC3  LD    B,A
02B3' 0E 00             996             LD    C,0               ; SCAN COUNT
02B5' 78                997     CLRSC4  LD    A,B
02B6' C5                998             PUSH  BC
02B7' E6 07             999             AND   7
02B9' 0F                1000            RRCA
02BA' 0F                1001            RRCA
02BB' 0F                1002            RRCA
02BC' 4F                1003            LD    C,A               ; BC=32*NO. OF LINES
02BD' 06 00             1004            LD    B,0               ; (=0 IF 256 FOR 8 LINES)
02BF' 0D                1005            DEC   C                 ; ADJUST
02C0' C5                1006    CLRSC5  PUSH  BC
02C1' E5                1007            PUSH  HL
02C2' 54                1008            LD    D,H
02C3' 5D                1009            LD    E,L
02C4' 36 00             1010            LD    (HL),00           ; CLEAR OP
02C6' 13                1011            INC   DE
02C7' ED B0             1012            LDIR
02C9' E1                1013            POP   HL
02CA' C1                1014            POP   BC
02CB' 7C                1015            LD    A,H
02CC' CB 6F             1016            BIT   5,A
02CE' 20 05             1017            JR    NZ,CLRSC6
02D0' F6 20             1018            OR    20H
02D2' 67                1019            LD    H,A               ; DO OP2
02D3' 18 EB             1020            JR    CLRSC5
02D5  E6 DF             1021    CLRSC6  AND   0DFH              ; RESTORE OP1
02D7' 67                1022            LD    H,A
02D8' 24                1023            INC   H                 ; NEXT SCAN ROW
02D9' C1                1024            POP   BC                ; POP SCAN COUNT
02DA' 0D                1025            DEC   C
02DB' 20 D8             1026            JR    NZ,CLRSC4
02DD' C1                1027            POP   BC                ; TOTAL LINE COUNT
02DE' 78                1028            LD    A,B
02DF' A7                1029            AND   A
02E0' 28 05             1030            JR    Z,CLRSXT
02E2' 2E 00             1031            LD    L,C               ; HL=START OF NEXT BLOCK
02E4' C3 0243'          1032            JP    CLRSC1            ; B=REMAINING LINE COUNT
02E7' C1                1033    CLRSXT  POP   BC                ; POSITION
02E8' C3 0507'          1034            JP    UPDPOSN           ; RETURN VIA UPDATE AND STORE POSITION
02EB' 4F                1035    CLRSC7  LD    C,A               ; SAVE NO. LINES THIS BLOCK
02EC' 78                1036            LD    A,B
02ED' 91                1037            SUB   C                 ; ADJUST TOTAL LINE COUNT
02EE' 47                1038            LD    B,A
02EF' C5                1039            PUSH  BC                ; REMAINING LINE COUNT
02F0' 79                1040            LD    A,C               ; LINES THIS BLOCK
02F1' 18 BF             1041            JR    CLRSC3            ;
                        1042            SUBTTL ATTRIBUTE CONTROL
                        1043    ;
                        1044    ;
                        1045    ;
02F3'                   1046    SETAB0:                         ; HERE FROM BASIC ENTRY TO SET ATTRIBUTES
02F3' 3A 0011'          1047            LD    A,(ATTCTL)        ; GET CONTROL INFO.
                        1048    ;
02F6'                   1049    SETATT:                         ; ENTRY WITH ATTRIBUTE BYTE IN A
02F6' E6 07             1050            AND   7                 ; GET INK SELECTION
02F8' 4F                1051            LD    C,A               ; SAVE
02F9' 3A 0CC2           1052            LD    A,(VIOMOD)        ; TEST IF OP2 OPEN
02FC' A7                1053            AND   A
02FD' CA 0147'          1054            JP    Z,INVPAR          ; INVALID IF OP2 NOT OPEN
0300' 79                1055            LD    A,C
0301' 17                1056            RLA
0302' 17                1057            RLA
0303' 17                1058            RLA                     ; SHIFT TO H/W POSITION
0304' F5                1059            PUSH  AF                ; SAVE ROTATED VALUE
0305' F6 06             1060            OR    6                 ; SET H/W MODE
0307' 47                1061            LD    B,A               ; VALUE FOR H/W
0308' DB FF             1062            IN    A,(HREXPT)        ;
030A' E6 C0             1063            AND   0C0H              ; PRESERVE BITS 7 AND 6
030C' B0                1064            OR    B                 ; SET VIDEO MODE VALUE
030D' D3 FF             1065            OUT   (HREXPT),A        ; SET H/W
030F' F6 40             1066            OR    40H               ; UPDATE VIOMOD
0311' 32 0CC2           1067            LD    (VIOMOD),A        ;
0314' F1                1068            POP   AF                ; INK SEL.IN BITS 3-5
0315' 2F                1069            CPL                     ; GET COMPLEMENTARY COLOR
0316' B1                1070            OR    C                 ; COMBINE WITH INK
0317' 32 0039'          1071            LD    (ATTBYT),A        ; SAVE IN VARIABLE
031A' C3 0142'          1072            JP    GOODRET           ; RETURN BC=0
                        1073    ;
                        1074    ;
                        1075    ;
0310'                   1076    SETMB0:                         ; HERE FROM BASIC ENTRY TO SET MASK
0310' 3A 0015'          1077            LD    A,(MSKCTL)        ; GET CONTROL INFO.
                        1078    ;
0320'                   1079    SETMSK:                         ; ENTRY WITH MASK VALUE IN A
0320' 32 0038'          1080            LD    (MASK8),A         ; STORE MASK FOR APPLICATION TO FUTURE
0323' C3 0142'          1081            JP    GOODRET           ; DISPLAYS
                        1082    ;
```

```
                                685             SUBTTL "GET" ROUTINES
                                686     ;
                                687     ;
0326*                           688     GTCH00:                         ; HERE FROM BASIC ENTRY TO GET CHARACTER
                                689                                     ; (RETURNS CODE FOR CHARACTER AT
                                690                                     ;  POSITION SPECIFIED IN GETCTL)
0326* 08 48 0019*               691             LD    BC,(GETCTL)       ; GET CONTROL INFO.
                                692     ;
0329*                           693     GTCHAR:                         ; ENTRY WITH POSITION IN BC
0329* CD 0480*                   694             CALL TSTPAR             ; TEST PARAMETERS
032C* 3A 0038*                   695             LD    A,(CPOSIT)        ; SAVE CURRENT BIT POSITION
032F* F5                        696             PUSH  AF
0330* CD 049C*                   697             CALL  CON:FM            ; CONVERT TO INTERNAL FORMAT
0333* CD 050C*                   698             CALL  CA,CPOS           ; GET DP ADDRESS AND BIT POS.
0336* ED 58 002F*                699             LD    DE,(CHTBL)        ; CHAR.TABLE
033A* 06 60                     700             LD    B,96              ; NO. OF PRINTABLE CHARACTERS
033C* 3E 00                     701             LD    A,00H             ; SET ADJUSTMENT INDEX
033E* 32 003A*                  702     GTCH1:  LD    (GTINDX),A
0341* 14                        703             INC   D                 ; ADJUST TO START OF TABLE
0342* C5                        704     GTCH2:  PUSH  BC                ; SAVE COUNT
0343* E5                        705             PUSH  HL                ; SAVE ADRS. IN DP
0344* D5                        706             PUSH  DE                ; SAVE ADRS. IN CHAR.TABLE
0345* CD 0560*                  707             CALL  GTC20C            ; GET SCAN LINE FROM DP TO REG.B
                                708                                     ; BITS 7-2
0348* 78                        709     GTCH22: LD    A,B
0349* E6 FC                     710             AND   0FCH              ; 6 BITS TO A
034B* 47                        711             LD    B,A
034C* E0                        712             EX    DE,HL             ; CHAR.SET PTR. TO HL
034D* 7E                        713             LD    A,(HL)            ; PIXEL ROW TO A
034E* E6 FC                     714             AND   0FCH              ; MASK TO 6 BITS
0350* A8                        715             XOR   B                 ; TEST AGAINST DP
0351* 28 10                    716             JR    Z,GTCH3           ; MATCH
0353* F5                        717             PUSH  AF
0354* 3A 003A*                  718             LD    A,(GTINDX)        ; TEST IF LOOKING AT STD.GRAPHICS
0357* FE 90                     719             CP    90H
0359* 20 03                    720             JR    NZ,GTCH23
035B* F1                        721             POP   AF
035C* 18 38                    722             JR    GTCH4             ; DO NOT TEST INVERSE IF STD.GR.
035E* F1                        723     GTCH23: POP   AF
035F* FE FC                     724             CP    0FCH              ; TEST IF MATCH ON INVERSE
0361* 20 34                    725             JR    NZ,GTCH4          ; DOES NOT MATCH
0363* 4F                        726     GTCH3:  LD    C,A               ; C=B FOR MATCH  -4 FOR INVERSE
0364* 06 07                    727             LD    B,7               ; SCAN COUNT
0366* E0                        728     GTCH31: EX    DE,HL             ; DP ADRS.TO HL
0367* C5                        729             PUSH  BC        ; SAVE MATCH AND SCAN COUNT
0368* 24                        730             INC   H         ; NEXT SCAN IN DP
0369* 13                        731             INC   DE        ; NEXT ROW IN CHAR.SET
036A* CD 0560*                  732             CALL  GTC20C            ; NEXT SCAN ROW TO BC
036D* 78                        733             LD    A,B
036E* E6 FC                     734             AND   0FCH
0370* 47                        735             LD    B,A               ;
0371* E0                        736             EX    DE,HL             ; CHAR.SET PTR. TO HL
0372* 7E                        737             LD    A,(HL)            ; CHAR.SET ROW
0373* E6 FC                     738             AND   0FCH              ; MASK TO 6 BITS
0375* A8                        739             XOR   B                 ; TEST MATCH
0376* C1                        740             POP   BC                ; GET PREV.MATCH AND SCAN COUNT
0377* A9                        741             XOR   C
0378* 20 1F                    742             JR    NZ,GTCH4          ; NO MATCH-START AGAIN AT NEXT
037A* 10 EA                    743             DJNZ  GTCH31            ; DO NEXT SCAN
037C* 79                        744             LD    A,C               ; A=-4 IF MATCH ON INVERSE
037D* C6 03                    745             ADD   A,3               ; SET TO -1 IF MATCH ON INVERSE
037F* C1                        746             POP   BC                ; HERE IF MATCH- (SP)=PTR.TO CHAR.
0380* C1                        747             POP   BC                ; IN TABLE:(SP+2)=CHAR.IN DP:
0381* C1                        748             POP   BC                ; (SP+4)=CHAR.COUNT
0382* 4F                        749             LD    C,A               ; B=COUNT C=-1 IF MATCH ON INVERSE
0383* 3A 003A*                  750             LD    A,(GTINDX)        ; CONVERT MATCH TO CHAR.CODE
0386* 90                        751             SUB   B
0387* FE 20                    752             CP    20H               ; TEST IF MATCH ON SPACE
0389* 20 05                    753             JR    NZ,GTCH32         ; NO
038B* 0C                        754             INC   C                 ; TEST IF MATCH ON INVERSE
038C* 20 02                    755             JR    NZ,GTCH32
038E* 3E 0F                    756             LD    A,0FH             ; LOAD CODE FOR GRAPHICS BLOCK
0390* 4F                        757     GTCH32  LD    C,A               ; RETURN IN REG. C OF BC PAIR
0391* 06 00                    758             LD    B,0               ; AND IN REG. A
0393* F1                        759             POP   AF
0394* 32 0038*                  760             LD    (CPOSIT),A        ; RESTORE DPSIT
0397* 79                        761             LD    A,C               ; CHAR.CODE TO A
0398* C9                        762             RET
                                763     ;
0399* E1                        764     GTCH4:  POP   HL                ; PTR. TO CHR. SET
039A* 11 0008                  765             LD    DE,8              ; MOVE ON TO NEXT CHARACTER
039D* 19                        766             ADD   HL,DE
039E* 58                        767             LD    E,L
039F* 54                        768             LD    D,H
03A0* E1                        769             POP   HL                ; DP ADDRESS
03A1* C1                        770             POP   BC                ; CHAR. COUNT
03A2* 10 9E                    771             DJNZ  GTCH2             ; DECREMENT CHAR.COUNT AND LOOK AGAIN
                                772     ;
03A4* 3A 003A*                  773             LD    A,(GTINDX)        ; TEST IF DONE
03A7* FE 80                     774             CP    80H               ; TEST IF STD.CHR.SET
03A9* 20 0A                    775             JR    NZ,GTCH5          ; TRY GRAPHICS
03AB* ED 5B 0031*               776             LD    DE,(GRTBL)                ; GRAPHICS TABLE
03AF* 06 10                    777             LD    B,16              ; NO. OF ENTRIES
03B1* 3E 90                     778             LD    A,90H
03B3* 18 09                    779             JR    GTCH1
03B5* FE 90                     780     GTCH5:  CP    90H               ; TEST IF STD.GRAPHICS
03B7* 20 0C                    781             JR    NZ,GTCH6          ;
03B9* ED 5B 5C78                782             LD    DE,(USR)          ; TRY USER-DEFINED GRAPHICS
03BD* 13                        783             DEC   D                 ; ADJUST ADDRESS-100H
03BE* 06 1B                    784             LD    B,27              ; NO. OF ENTRIES
03C0* 3E A5                     785             LD    A,0A5H            ; INDEX ADJUST
03C2* C3 033F*                  786             JP    GTCH1
                                787
                                788     ;                               ; NO MATCH AT ALL
```

```
03C6' F1                    789 GTCM6   POP   AF                  ; RESTORE ORIG. OPBIT
03C7' 32 003E'              790         LD    (OPBIT),A
03CA' 48                    791         LD    C,B                 ; BC=0
03CB' AF                    792         XOR   A                   ; A=0
03CC' C9                    793         RET
                            794
                            795                               ; GET ATTRIBUTE BYTE
                            796                               ; NOTE THAT IN 80-COL.MODE ALL SCREEN POSITIONS
                            797                               ; HAVE THE SAME ATTRIBUTES, THEREFORE IT IS
                            798                               ; NOT NECESSARY TO USE THE 'GETCTL' POSITION
                            799                               ; PARAMETERS
                            800 ;
03CD'                       801 GETAB0:                       ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                            802 ;
03CD' 3A 0039'              803 GETATT: LD    A,(ATTBYT)      ; GET ATTRIBUTE BYTE
03D0' 4F                    804         LD    C,A             ; PUT IN BC
03D1' 06 00                 805         LD    B,0
03D3' C9                    806         RET
                            807 ;                             ; GET CURSOR POSITION
03D4'                       808 GETCB0:                       ; HERE FROM BASIC ENTRY
03D4'                       809 GETCUR:
03D4' ED 4B 0034'           810         LD    BC,(CURPOS)     ; GET INTERNAL POSITION
03D8' CD 04FC'              811         CALL  CONVPM          ; CONVERT TO USER FORMAT
03DB' 3A 0033'              812         LD    A,(LINLEN)      ; TEST IF END OF LINE
03DE' B9                    813         CP    C               ; IF =00 OR 05
03DF' 20 08                 814         JR    NZ,GETC1        ; NO
03E1' 0E 00                 815         LD    C,0             ; NEXT POSN.START OF NEXT LINE
03E3' 78                    816         LD    A,B
03E4' 3C                    817         INC   A               ; BUMP TO NEXT LINE
03E5' 47                    818         LD    B,A
03E6' FE 18                 819         CP    24              ; TEST IF OFF SCREEN
03E8' 38 02                 820         JR    C,GETC1         ; NO
03EA' 06 17                 821         LD    B,23            ; POSN. IS AT BOTTOM LINE
03EC' ED 43 0007'           822 GETC1   LD    (LINCOL),BC     ; STORE FOR BASIC PROGRAM
03F0' C9                    823         RET                   ; VALUES ALSO IN BC
                            824 ;
                            825         SUBTTL VIDEO MODE CONTROL
                            826 ;
                            827 ;
03F1'                       828 SYMOD0:                       ; HERE FROM BASIC ENTRY WITH MODE IN
                            829                               ; 'VIMCDE'
03F1' 3A 0024'              830         LD  A,(VIMCDE)
                            831 ;
03F4'                       832 SETMODE:                      ; ENTRY WITH MODE IN A
                            833 ;
03F4' A7                    834         AND  A
03F5' 28 4F                 835         JR   Z,SETMD1         ; TEST FOR VALIDITY
03F7' FE 08                 836         CP   8                ; 0-80 COL.MODE
03F9' C2 0147'              837         JP   NZ,INVPAR        ; INVALID PARAMETERS
                            838 ;                             ;
                            839 ;                             ;
03FC' 21 1701               840         LD   HL,SCINIT        ; INITIALIZE VARIABLES
03FF' 22 0028'              841         LD   (SCRCTL),HL      ; SCROLL CONTROL
0402' 3E 17                 842         LD   A,23             ;
0404' 32 0020'              843         LD   (BOTLN),A        ; BOTTOM LINE
0407' 3E 01                 844         LD   A,1              ;
0409' 32 002E'              845         LD   (SCRLCT),A       ; SCROLL COUNT
040C' 3E 50                 846         LD   A,80             ;
040E' 32 0033'              847         LD   (LINLEN),A       ; LINE LENGTH
0411' 3E 01                 848         LD   A,1              ;
0413' 32 0030'              849         LD   (MARGIN),A       ; MARGIN
0416' 21 1000               850         LD   HL,CLINIT        ;
0419' 22 000C'              851         LD   (CLRCTL),HL      ; CLEAR SCREEN CONTROL
041C' AF                    852         XOR  A
041D' 32 0030'              853         LD   (MASK0),A        ; MASK BYTE
0420' 32 0036'              854         LD   (STRGCT),A       ; WRSTRG REMAINING COUNT
0423' 32 003C'              855         LD   (STRGCT+1),A     ;
0426' 21 0478'              856         LD   HL,CHRSET        ; STD. CHAR. SET
0429' 22 002F'              857         LD   (CHTBL),HL       ;
042C' 21 0778'              858         LD   HL,GRPHST        ; STD. GRAPHICS SET
042F' 22 0031'              859         LD   (GRTBL),HL       ;
                            860 ;                             ;
                            861 ;                             ; INITIALIZE BASIC INPUTS
                            862 ;                             ;
0432' 32 0000'              863         LD   (DATA0),A        ; DATA BYTE
0435' 32 0007'              864         LD   (LINCCL),A       ; COLUMN
0438' 32 0050'              865         LD   (LINCCL+1),A     ; LINE
043B' 32 0010'              866         LD   (MSKCTL),A       ; MASK CONTROL
043E' 32 0019'              867         LD   (GETCTL),A       ; 'GET' COL.
0441' 32 001A'              868         LD   (GETCTL+1),A     ; 'GET' LINE
0444' 3E 06                 869         LD   A,6              ; SET H/W TO 64-COL.MODE
0446' A7                    870 SETMD1  LD   B,A              ; SAVE MODE
0447' 3A 9CC2               871         LD   A,(VIDMOD)
044A' A7                    872         AND  A                ; TEST CURRENT MODE
044B' 28 10                 873         JR   NZ,SETMD2
044D' 80                    874         OR   B                ; CURRENT MODE=0  TEST IF NEW MODE=0
044E' CA 0142'              875         JP   Z,GOODRET        ; RETURN BC=0
0451' 21 12C0               876         LD   HL,INSERTSZ
0454' 11 0040               877         LD   DE,MOVESZ
0457' 19                    878         ADD  HL,DE            ; HL=TOTAL ROOM NEEDED
0458' ED 5B 9C65            879         LD   DE,(STKEND)
045C' 19                    880         ADD  HL,DE            ; ADD MEMORY IN USE
045D' DA 04A7'              881         JP   C,EXTNRM         ; NOT ENOUGH MEMORY TO OPEN OPZ
0460' ED 5B 9C82            882         LD   DE,(RAMTOP)
0464' A7                    883         AND  A
0465' ED 52                 884         SBC  HL,DE
0467' D2 04A7'              885         JP   NC,EXTNRM        ; NOT ENOUGH MEMORY
046A' CD 04D2'              886 SETMD2  CALL ENBLEXT          ; ENABLE ROM EXT.
046D' 78                    887         LD   A,B              ; MODE TO A
046E' CD 04AC'              888         CALL GETVAL           ; H/W VALUE TO B; VIDMOD VALUE TO C
0471' C5                    889         PUSH BC               ; SAVE
0472' 78                    890         LD   A,B
0473' CD 0E88               891         CALL CHNGVID          ; CALL RTN. TO OPEN/CLOSE 2ND DP
0476' CD 04DF'              892         CALL ENBLHOME         ; REENABLE HOME BANK
0479' C1                    893         POP  BC               ; VALUE FOR VIDMOD
047A' 79                    894         LD   A,C
047B' 32 9CC2               895         LD   (VIDMOD),A; UPDATE VIDMOD
047E' CB 77                 896 UPDATE  BIT  6,A              ; UPDATE SYSTEM VARS. BASED ON MODE
0480' CA 0142'              897         JP   Z,GOODRET        ; DONE IF MODE 0
0483' 3E C1                 898 SXT4    LD   A,CC1H           ; 64/80-COL. MODE
0485' B0                    899         OR   B                ; STORE ATTRIBUTE BYTE BASED
0486' 2F                    900         CPL                   ; ON INK SELECTION
```

```
0487' CB 38                902             SRL   B
0489' CB 38                903             SRL   B
048B' CB 38                904             SRL   B
048D' B0                   905             OR    B
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 18                911             LD    B,18H             ; BC=HOME POSITION (LN.0/COL.0)
049B' 21 0000              912             LD    HL,0
049E' 22 5CF5              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   EXTNRM    LD    C,2               ; RETURN BC=2 FOR NO ROOM
04A9' C3 0144'             917             JP    ERRRET
                           918   !
                           919             SUBTTL INTERNAL SUB-RTNS.
                           920   !
                           921   !
                           922   !
                           923
                           924                                     ; INPUT:  VIDEO MODE IN A
                           925                                     ; OUTPUT: M/N SETTING IN B
                           926                                     ;         VIDMOD SETTING IN C
                           927                                     ; RETURNS NZ STATUS IF MODE INVALID
                           928                                     ;
                           929                                     ;  VALID VALUES:  INPUT     M/N     VIDMOD
                           930                                     ;                  -----     ---     ------
                           931                                     ;   **DISPLAY FILE ACTIVE AT SCREEN
                           932                                     ;
                           933                                     ;     OP1 ONLY       0        0        0
                           934                                     ;   OP1**& OP2       80       0        80
                           935                                     ;    OP1 & OP2**      1        1        81
                           936                                     ;  HIGH RES.GR.       2        2        2
                           937                                     ;   64/80-COLUMN    06 - 3E  06 - 3E  46 - 7E
04AC' 4F                   938   GETVAL    LD    C,A
04AD' CB 47                939             BIT   0,A
04AF' 20 16                940             JR    NZ,TST01
04B1' FE 80                941             CP    80H
04B3' 28 1A                942             JR    Z,SET0
04B5' CB 7F                943             BIT   7,A
04B7' C0                   944             RET   NZ
04B8' 47                   945             LD    B,A
04B9' FE 02                946             CP    2
04BB' C8                   947             RET   Z
04BC' E6 C6                948             AND   0C6H
04BE' EE 06                949             XOR   6
04C0' C0                   950             RET   NZ
04C1' 3E 40                951             LD    A,40H
04C3' B1                   952             OR    C
04C4' 4F                   953             LD    C,A
04C5' AF                   954             XOR   A
04C6' C9                   955             RET
04C7' FE 01                956   TST01     CP    1
04C9' C1                   957             RET   NZ
04CA' 06 01                958             LD    B,1
04CC' 0E 81                959             LD    C,81H
04CE' C9                   960             RET
04CF' AF                   961   SET0      XOR   A
04D0' 47                   962             LD    B,A
04D1' C9                   963             RET
                           964   !
                           965   !
                                                   ; ROUTINES TO ENABLE CHUNK 0 IN ROM EXT.
                           967                     ; FOR ACCESS TO CHNGVID ROUTINE
04D2' F5                   968   ENBLEXT   PUSH  AF
04D3' DB FF                969             IN    A,(MREXPT)
04D5' F6 80                970             OR    80H         ; SELECT ROM EXT.
04D7' D3 FF                971             OUT   (MREXPT),A
04D9' 3E 03                972             LD    A,3
04DB' D3 F4                973             OUT   (DRMSPT),A
04DD' F1                   974             POP   AF
04DE' C9                   975             RET
                           976   !
04DF' F5                   977   ENBLMOME  PUSH  AF
04E0' AF                   978             XOR   A
04E1' D3 F4                979             OUT   (DRMSPT),A
04E3' DB FF                980             IN    A,(MREXPT)
04E5' E6 3F                981             AND   03FH
04E7' D3 FF                982             OUT   (MREXPT),A
04E9' F1                   983             POP   AF
04EA' C9                   984             RET
                           985   !
                           986                                   ; TEST PARAMETER VALIDITY
                           987                                   ; IF NOT VALID, DISCARDS RETURN
                           988                                   ; AND EXITS VIA INVPAR
04EB' 3E 17                989   TSTPAR    LD    A,23            ; TEST LINE >23
04ED' B8                   990             CP    B
04EE' 30 04                991             JR    NC,TSTPR1
04F0' F1                   992   PARERR    POP   AF
04F1' C3 0147'             993             JP    INVPAR
04F4' 3A 0033'             994   TSTPR1    LD    A,(LINLEN)
04F7' 3D                   995             DEC   A
04F8' B9                   996             CP    C               ; TEST COL. >LINE LENGTH-1
04F9' 38 F9                997             JR    C,PARERR
04FB' C9                   998             RET
                           999   !
                          1000                                   ; CONVERT LINE/COL. FROM USER TO
                          1001                                   ; INTERNAL FORMAT AND VICE VERSA
                          1002                                   ; USER FORMAT: LINES 0-23
                          1003                                   ;              COL. 0-79(84)
                          1004                                   ; INTERNAL FORMAT:
                          1005                                   ;              LINES 24-1
                          1006                                   ;              COL. (86)81-2
                          1007                                   ;              (COL.1 END OF LINE)
                          1008   !
                          1009   !
04FC' 3A 0033'            1010   CONVPM    LD    A,(LINLEN)
```

```
04FF'  3C                  1011            INC     A
0500'  91                  1012            SUB     C
0501'  4F                  1013            LD      C,A             ; CONVERT
0502'  3E 10               1014            LD      A,SCRSZ
0504'  98                  1015            SUB     B
0505'  47                  1016            LD      B,A
0506'  C9                  1017            RET
                           1018    ;
                           1019    ;
                           1020    ;
0507'                      1021    UPDPOSN:                        ; ENTER HERE WITH DC=LINE/COL.IN
                           1022                                    ; INTERNAL FORMAT
0507'  CC 050C'            1023            CALL    CALCPOS         ; CALCULATE POSITION
050A'  18 44               1024            JR      STPOSN          ; STORE UPDATED POSITION AND RETURN
                           1025    ;
                           1026    ;                               ; ROUTINE TO CALCULATE POSITION IN DP
                           1027    ;                               ; UPDATES DIF POSITION IN VARIABLE DPDIF
                           1028    ;                               ; RETURNS DP ADDRESS IN HL.  PRESERVES
                           1029    ;                               ; LINE/COL.POSITION IN DC
                           1030    ;
050C'  CD 053F'            1031    CALCPOS CALL    LNBU            ; GET DISPLAY FILE ADRS.FOR LINE
050F'  3A 0033'            1032            LD      A,(LINLEN)      ; TEST WHICH DP
0512'  3C                  1033            INC     A
0513'  91                  1034            SUB     C
0514'  F5                  1035            PUSH    AF              ; SAVE COL.NO.(0-(LINLEN-1))
0515'  1F                  1036            LD      E,A
0516'  87                  1037            ADD     A,A
0517'  83                  1038            ADD     A,E             ; 3*(80-COL.POS.) / 4 = 64-COL.POS.
0518'  CB 3F               1039            SRL     A
051A'  CB 3F               1040            SRL     A
051C'  CB 47               1041            BIT     0,A
051E'  28 06               1042            JR      Z,CALCP1
0520'  F5                  1043            PUSH    AF              ; ODD COLS.ARE IN DP2
0521'  3E 20               1044            LD      A,20H
0523'  84                  1045            OR      H
0524'  67                  1046            LD      H,A
0525'  F1                  1047            POP     AF
0526'  A7                  1048    CALCP1  AND     A               ; CLEAR CARRY
0527'  1F                  1049            RRA                     ; COL/2 IS POSITION IN DISPLAY FILE
0528'  5F                  1050            LD      E,A
0529'  3A 0030'            1051            LD      A,(MARGIN)      ; GET ADJUSTMENT VALUE
052C'  83                  1052            ADD     A,E
052D'  5F                  1053            LD      E,A
052E'  16 00               1054            LD      D,0             ; DE CONTAINS OFFSET INTO LINE
0530'  19                  1055            ADD     HL,DE           ; HL CONTAINS DP ADDRESS
0531'  1E 07               1056            LD      E,7
0533'  F1                  1057            POP     AF              ; A=COL.POSN.FOR 80 COLS.
0534'  E6 03               1058            AND     3               ;
0536'  28 02               1059            JR      Z,CALCP2        ; STARTS AT BIT 7
0538'  3F                  1060            LD      E,A             ; STARTS AT BIT 1,3 OR 5
0539'  3D                  1061            DEC     A
053A'  83                  1062    CALCP2  ADD     A,E
053B'  32 0038'            1063            LD      (DPBIT),A
053E'  C9                  1064            RET                     ; CURPOS. HL=DP ADDRESS
                           1065    ;
                           1066    ;
053F'                      1067    LNBU:                           ; GET DISPLAY FILE ADDRESS
                           1068                                    ; FOR START OF LINE IN B
053F'  3E 18               1069            LD  A,SCRSZ
0541'  98                  1070            SUB B
0542'  57                  1071            LD  D,A
0543'  0F                  1072            RRCA
0544'  0F                  1073            RRCA
0545'  0F                  1074            RRCA
0546'  E6 E0               1075            AND 0E0H
0548'  6F                  1076            LD  L,A
0549'  7A                  1077            LD  A,D
054A'  E6 18               1078            AND 18H
054C'  F6 40               1079            OR  40H
054E'  67                  1080            LD  H,A
054F'  C9                  1081            RET
                           1082
0550'                      1083    STPOSN:                         ; STORE CURSOR POSITION
0550'  ED 43 0034'         1084            LD  (CURPOS),BC
0554'  22 0036'            1085            LD  (DPADRS),HL
0557'  C9                  1086            RET
                           1087    ;
0558'                      1088    LDPOSN:                         ; LOAD CURSOR POSITION
0558'  ED 4B 0034'         1089            LD  BC,(CURPOS)
055C'  2A 0036'            1090            LD  HL,(DPADRS)
055F'  C9                  1091            RET
                           1092    ;
                           1093    ;
                           1094    ;                               ; RTN. USED BY GTCHAR TO PUT
                           1095    ;                               ; 6 PIXELS FROM DP IN B REG.BITS 7-2
                           1096    ;                               ; FOR MATCHING AGAINST CHAR.SET
                           1097    ;
0560'  E5                  1098    GTC2DC  PUSH    HL              ; SAVE DP ADRS.
0561'  46                  1099            LD      B,(HL)          ; SCAN LINE FROM DP
0562'  7C                  1100            LD      A,H
0563'  EE 20               1101            XOR     20H
0565'  67                  1102            LD      H,A             ; TOGGLE TO ALTERNATE DP
0566'  CB 6F               1103            BIT     5,A             ; TEST WHICH DP
0568'  28 01               1104            JR      NZ,GTC2B1
056A'  23                  1105            INC     HL
056B'  4E                  1106    GTC2B1  LD      C,(HL)          ; DC CONTAINS CHAR.AT DPBIT
056C'  3A 0038'            1107            LD      A,(DPBIT)
056F'  D6 07               1108            SUB     7
0571'  ED 44               1109            NEG                     ; A=NO.OF BITS OFFSET IN DC
0573'  28 07               1110            JR      Z,GTC2B2        ; STARTS AT BIT 7
0575'  CB 11               1111    GTCSHFT RL      C
0577'  CB 10               1112            RL      B               ; SHIFT 6 PIXELS FOR CHARACTER
0579'  3D                  1113            DEC     A               ; TO B, BITS 7-2
057A'  20 F9               1114            JR      NZ,GTCSHFT
057C'  E1                  1115    GTC2B2  POP     HL              ;
057D'  C9                  1116            RET                     ; CHAR.IN B, BITS 7-2
                           1117    ;
                           1118
                           1119            INCLUDE CHSTBB
                           1120            SUBTTL CHAR.SET FOR 40/80 COL.MODES
```

```
                        1121
                        1122    ;
                        1123    ; Dot patterns for characters on the TV screen!
                        1124    ;       character with code x is at offsets 8x (top line) to 8x+7 (bottom line)
                        1125    ;       0 = white, 1 = black, no margins between character spaces either
                        1126    ;               vertically, or horizontally
                        1127
0578' 00                1128    CHRST   defb 00000000b          ; code 20       space
0579' 00                1129            defb 00000000b
057A' 00                1130            defb 00000000b
057B' 00                1131            defb 00000000b
057C' 00                1132            defb 00000000b
057D' 00                1133            defb 00000000b
057E' 00                1134            defb 00000000b
057F' 00                1135            defb 00000000b
                        1136
0580' 00                1137            defb 00000000b          ; code 21       !
0581' 10                1138            defb 00010000b
0582' 10                1139            defb 00010000b
0583' 10                1140            defb 00010000b
0584' 10                1141            defb 00010000b
0585' 00                1142            defb 00000000b
0586' 10                1143            defb 00010000b
0587' 00                1144            defb 00000000b
                        1145
0588' 00                1146            defb 00000000b          ; code 22       "
0589' 48                1147            defb 01001000b
058A' 48                1148            defb 01001000b
058B' 00                1149            defb 00000000b
058C' 00                1150            defb 00000000b
058D' 00                1151            defb 00000000b
058E' 00                1152            defb 00000000b
058F' 00                1153            defb 00000000b
                        1154
0590' 00                1155            defb 00000000b          ; code 23       #
0591' 28                1156            defb 00101000b
0592' 7C                1157            defb 01111100b
0593' 28                1158            defb 00101000b
0594' 28                1159            defb 00101000b
0595' 7C                1160            defb 01111100b
0596' 28                1161            defb 00101000b
0597' 00                1162            defb 00000000b
                        1163
0598' 00                1164            defb 00000000b          ; code 24       $
0599' 10                1165            defb 00010000b
059A' 7C                1166            defb 01111100b
059B' 90                1167            defb 01011000b
059C' 7C                1168            defb 01111100b
059D' 14                1169            defb 00101000b
059E' 7C                1170            defb 01111100b
059F' 10                1171            defb 00010000b
                        1172
05A0' 00                1173            defb 00000000b          ; code 25       %
05A1' 44                1174            defb 01100100b
05A2' 68                1175            defb 01101000b
05A3' 10                1176            defb 00010000b
05A4' 10                1177            defb 00010000b
05A5' 2C                1178            defb 00110100b
05A6' 4C                1179            defb 01001100b
05A7' 00                1180            defb 00000000b
                        1181
05A8' 00                1182            defb 00000000b          ; code 26       &
05A9' 18                1183            defb 00011000b
05AA' 28                1184            defb 00101000b
05AB' 10                1185            defb 00010000b
05AC' 2C                1186            defb 00101100b
05AD' 48                1187            defb 01001000b
05AE' 34                1188            defb 00110100b
05AF' 00                1189            defb 00000000b
                        1190
05B0' 00                1191            defb 00000000b          ; code 27       '
05B1' 10                1192            defb 00010000b
05B2' 20                1193            defb 00100000b
05B3' 00                1194            defb 00000000b
05B4' 00                1195            defb 00000000b
05B5' 00                1196            defb 00000000b
05B6' 00                1197            defb 00000000b
05B7' 00                1198            defb 00000000b
                        1199
05B8' 00                1200            defb 00000000b          ; code 28       (
05B9' 08                1201            defb 00001000b
05BA' 10                1202            defb 00010000b
05BB' 10                1203            defb 00010000b
05BC' 10                1204            defb 00010000b
05BD' 10                1205            defb 00010000b
05BE' 08                1206            defb 00001000b
05BF' 00                1207            defb 00000000b
                        1208
05C0' 00                1209            defb 00000000b          ; code 29       )
05C1' 20                1210            defb 00100000b
05C2' 10                1211            defb 00010000b
05C3' 10                1212            defb 00010000b
05C4' 10                1213            defb 00010000b
05C5' 10                1214            defb 00010000b
05C6' 20                1215            defb 00100000b
05C7' 00                1216            defb 00000000b
                        1217
05C8' 00                1218            defb 00000000b          ; code 2A       *
05C9' 00                1219            defb 00000000b
05CA' 28                1220            defb 00101000b
05CB' 10                1221            defb 00010000b
05CC' 7C                1222            defb 01111100b
05CD' 10                1223            defb 00010000b
05CE' 28                1224            defb 00101000b
05CF' 00                1225            defb 00000000b
                        1226
05D0' 00                1227            defb 00000000b          ; code 2B       +
05D1' 00                1228            defb 00000000b
05D2' 10                1229            defb 00010000b
05D3' 10                1230            defb 00010000b
05D4' 7C                1231            defb 01111100b
```

```
0580' 18          1232        defb 000110000b
053C' 18          1233        defb 000110000b
0583' 00          1234        defb 000000000b
                  1235
058E' 00          1236        defb 000C00000b     ; code 2C     ,
058F' 00          1237        defb 000000000b
0590' 00          1238        defb 000C00000b
0591' 00          1239        defb 000000000b
0592' 00          1240        defb 000C00000b
0593' 18          1241        defb 000110000b
0594' 18          1242        defb 000110000b
0595' 20          1243        defb 001000000b
                  1244
0596' 00          1245        defb 000000000b     ; code 2D     -
0597' 00          1246        defb 000C00000b
0598' 00          1247        defb 000000000b
0599' 00          1248        defb 000C00000b
059A' 7C          1249        defb 011111100b
059B' 00          1250        defb 000000000b
059C' 00          1251        defb 000000000b
059D' 00          1252        defb 000000000b
                  1253
059E' 00          1254        defb 000C00000b     ; code 2E     .
059F' 00          1255        defb 000000000b
05A0' 00          1256        defb 000000000b
05A1' 00          1257        defb 000000000b
05A2' 00          1258        defb 000C00000b
05A3' 3C          1259        defb 001110000b
05A4' 33          1260        defb 001100000b
05A5' 00          1261        defb 000C00000b
                  1262
05A6' 00          1263        defb 000C00000b     ; code 2F     /
05A7' 00          1264        defb 000000000b
05A8' 04          1265        defb 00C001000b
05A9' 08          1266        defb 000010000b
05AA' 10          1267        defb 000100000b
05AB' 20          1268        defb 001000000b
05AC' 40          1269        defb 010000000b
05AD' 00          1270        defb 000000000b
                  1271
05AE' 00          1272        defb 000000000b     ; code 30     0
05AF' 38          1273        defb 001110000b
0600' 4C          1274        defb 010011000b
0601' 04          1275        defb 010101000b
0602' 34          1276        defb 010101000b
0603' 64          1277        defb 011001000b
0604' 38          1278        defb 001110000b
0605' 00          1279        defb 000000000b
                  1280
0606' 00          1281        defb 000000000b     ; code 31     1
0607' 38          1282        defb 001110000b
0608' 08          1283        defb 000100000b
0609' 18          1284        defb 000110000b
060A' 18          1285        defb 000110000b
060B' 18          1286        defb 000110000b
060C' 7C          1287        defb 011111100b
060D' 00          1288        defb 000000000b
                  1289
060E' 00          1290        defb 000000000b     ; code 32     2
060F' 38          1291        defb 001110000b
0610' 44          1292        defb 010001000b
0611' 04          1293        defb 000001000b
0612' 38          1294        defb 001110000b
0613' 40          1295        defb 010000000b
0614' 7C          1296        defb 011111100b
0615' 00          1297        defb 000000000b
                  1298
0616' 00          1299        defb 000000000b     ; code 33     3
0617' 38          1300        defb 001110000b
0618' 44          1301        defb 010001000b
0619' 18          1302        defb 000110000b
061A' 04          1303        defb 000001000b
061B' 44          1304        defb 010001000b
061C' 38          1305        defb 001110000b
061D' 00          1306        defb 000000000b
                  1307
061E' 00          1308        defb 000C00000b     ; code 34     4
061F' 08          1309        defb 000210000b
0620' 18          1310        defb 000110000b
0621' 28          1311        defb 001C10000b
0622' 48          1312        defb 010010000b
0623' 7C          1313        defb 011111100b
0624' 08          1314        defb 000010000b
0625' 00          1315        defb 000000000b
                  1316
0626' 00          1317        defb 000000000b     ; code 35     5
0627' 7C          1318        defb 011111100b
0628' 40          1319        defb 010000000b
0629' 78          1320        defb 011110000b
062A' 04          1321        defb 000001000b
062B' 44          1322        defb 010001000b
062C' 38          1323        defb 001110000b
062D' 00          1324        defb 000C00000b
                  1325
062E' 00          1326        defb 000000000b     ; code 36     6
062F' 38          1327        defb 001110000b
0630' 40          1328        defb 010000000b
0631' 78          1329        defb 011110000b
0632' 44          1330        defb 010C01000b
0633' 44          1331        defb 010CC1000b
0634' 38          1332        defb 001110000b
0635' 00          1333        defb 000000000b
                  1334
0636' 00          1335        defb 000000000b     ; code 37     7
```

```
0637' 7C          1336          defb 01111100b
0638' 04          1337          defb 00000100b
0639' 08          1338          defb 00001000b
063A' 10          1339          defb 00010000b
063B' 20          1340          defb 00100000b
063C' 20          1341          defb 00100000b
063D' 00          1342          defb 00000000b
                  1343
063E' 00          1344          defb 00000000b        ; code 38     8
063F' 38          1345          defb 00111000b
0640' 44          1346          defb 01000100b
0641' 38          1347          defb 00111000b
0642' 44          1348          defb 01000100b
0643' 44          1349          defb 01000100b
0644' 38          1350          defb 00111000b
0645' 00          1351          defb 00000000b
                  1352
0646' 00          1353          defb 00000000b        ; code 39     9
0647' 38          1354          defb 00111000b
0648' 44          1355          defb 01000100b
0649' 44          1356          defb 01000100b
064A' 3C          1357          defb 00111100b
064B' 04          1358          defb 00000100b
064C' 38          1359          defb 00111000b
064D' 00          1360          defb 00000000b
                  1361
064E' 00          1362          defb 00000000b        ; code 3A     :
064F' 00          1363          defb 00000000b
0650' 00          1364          defb 00000000b
0651' 10          1365          defb 00010000b
0652' 00          1366          defb 00000000b
0653' 00          1367          defb 00000000b
0654' 10          1368          defb 00010000b
0655' 00          1369          defb 00000000b
                  1370
0656' 00          1371          defb 00000000b        ; code 3B     ;
0657' 00          1372          defb 00000000b
0658' 10          1373          defb 00010000b
0659' 00          1374          defb 00000000b
065A' 00          1375          defb 00010000b
065B' 10          1376          defb 00010000b
065C' 10          1377          defb 00010000b
065D' 20          1378          defb 00100000b
                  1379
065E' 00          1380          defb 00000000b        ; code 3C     <
065F' 00          1381          defb 00000000b
0660' 08          1382          defb 00001000b
0661' 10          1383          defb 00010000b
0662' 20          1384          defb 00100000b
0663' 10          1385          defb 00010000b
0664' 08          1386          defb 00001000b
0665' 00          1387          defb 00000000b
                  1388
0666' 00          1389          defb 00000000b        ; code 3D     =
0667' 00          1390          defb 00000000b
0668' 00          1391          defb 00000000b
0669' 7C          1392          defb 01111100b
066A' 00          1393          defb 00000000b
066B' 7C          1394          defb 01111100b
066C' 00          1395          defb 00000000b
066D' 00          1396          defb 00000000b
                  1397
066E' 00          1398          defb 00000000b        ; code 3E     >
066F' 00          1399          defb 00000000b
0670' 20          1400          defb 00100000b
0671' 10          1401          defb 00010000b
0672' 08          1402          defb 00001000b
0673' 10          1403          defb 00010000b
0674' 20          1404          defb 00100000b
0675' 00          1405          defb 00000000b
                  1406
0676' 00          1407          defb 00000000b        ; code 3F     ?
0677' 38          1408          defb 00111000b
0678' 44          1409          defb 01000100b
0679' 08          1410          defb 00001000b
067A' 10          1411          defb 00010000b
067B' 00          1412          defb 00000000b
067C' 10          1413          defb 00010000b
067D' 00          1414          defb 00000000b
                  1415
067E' 00          1416          defb 00000000b        ; code 40     @
067F' 38          1417          defb 00111000b
0680' 44          1418          defb 01000100b
0681' 5C          1419          defb 01011100b
0682' 5C          1420          defb 01011100b
0683' 40          1421          defb 01000000b
0684' 3C          1422          defb 00111100b
0685' 00          1423          defb 00000000b
                  1424
0686' 00          1425          defb 00000000b        ; code 41     A
0687' 38          1426          defb 00111000b
0688' 44          1427          defb 01000100b
0689' 44          1428          defb 01000100b
068A' 7C          1429          defb 01111100b
068B' 44          1430          defb 01000100b
068C' 44          1431          defb 01000100b
068D' 00          1432          defb 00000000b
                  1433
068E' 00          1434          defb 00000000b        ; code 42     B
068F' 78          1435          defb 01111000b
0690' 44          1436          defb 01000100b
0691' 78          1437          defb 01111000b
0692' 44          1438          defb 01000100b
0693' 44          1439          defb 01000100b
```

```
0694' 78                1440          defb 01111000b
0695' 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' 5C                1483          defb 01011100b
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' 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
```

```
06C2' 44            1492          defb 01100010b
06C3' 44            1493          defb 01100010b
06C4' 44            1494          defb 01100010b
06C5' 00            1495          defb 00000000b
                    1496
06C6' 00            1497          defb 00000000b          ; code 43     C
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 44     D
06CF' 04            1507          defb 00001100b
06D0' 04            1508          defb 00001100b
06D1' 04            1509          defb 00001100b
06D2' 44            1510          defb 01001100b
06D3' 44            1511          defb 01001100b
06D4' 3C            1512          defb 00111100b
06D5' 00            1513          defb 00000000b
                    1514
06D6' 00            1515          defb 00000000b          ; code 45     E
06D7' 40            1516          defb 01001000b
06D8' 98            1517          defb 01011000b
06D9' 00            1518          defb 01101000b
06DA' 98            1519          defb 01011000b
06DB' 40            1520          defb 01001000b
06DC' 44            1521          defb 01000100b
06DD' 00            1522          defb 00000000b
                    1523
06DE' 00            1524          defb 00000000b          ; code 46     F
06DF' 46            1525          defb 01000300b
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 47     G
06E7' 44            1534          defb 01001100b
06E8' 4C            1535          defb 01101100b
06E9' 54            1536          defb 01101100b
06EA' 44            1537          defb 01101100b
06EB' 44            1538          defb 01001100b
06EC' 44            1539          defb 01001100b
06ED' 00            1540          defb 00000000b
                    1541
06EE' C0            1542          defb 00000000b          ; code 48     H
06EF' 44            1543          defb 01001100b
06F0' 44            1544          defb 01101100b
06F1' 54            1545          defb 01101100b
06F2' 54            1546          defb 01101100b
06F3' 4C            1547          defb 01001100b
06F4' 44            1548          defb 01001100b
06F5' 00            1549          defb 00000000b
                    1550
06F6' 00            1551          defb 00000000b          ; code 49     I
06F7' 38            1552          defb 00111000b
06F8' 44            1553          defb 01001100b
06F9' 44            1554          defb 01001100b
06FA' 44            1555          defb 01001100b
06FB' 44            1556          defb 01001100b
06FC' 38            1557          defb 00111000b
06FD' 00            1558          defb 00000000b
                    1559
06FE' 00            1560          defb 00000000b          ; code 50     P
06FF' 78            1561          defb 01111100b
0700' 44            1562          defb 01001100b
0701' 44            1563          defb 01001100b
0702' 78            1564          defb 01111000b
0703' 40            1565          defb 01001000b
0704' 40            1566          defb 01001000b
0705' 00            1567          defb 00000000b
                    1568
0706' 00            1569          defb 00000000b          ; code 51     Q
0707' 38            1570          defb 00111000b
0708' 44            1571          defb 01001100b
0709' 44            1572          defb 01001100b
070A' 44            1573          defb 01001100b
070B' 54            1574          defb 01101100b
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 01001100b
0711' 44            1581          defb 01001100b
0712' 78            1582          defb 01111000b
0713' 48            1583          defb 01001000b
0714' 44            1584          defb 01001100b
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 01001100b
071C' 38            1593          defb 00111000b
071D' 00            1594          defb 00000000b
                    1595
```

```
0718' 00          1596          defb 00000000b          ; code 94      T
0719' 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 95      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 96      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 97      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 98      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 99      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 9A      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 9B      [
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 9C      \
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 9D      ]
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 9E      ^
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 9F      _
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
                      1703
077E' 00              1704        defb 00000000b          ; code 60       pound sign
077F' 18              1705        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
0785' 00              1711        defb 00000000b
                      1712
0786' 00              1713        defb 00000000b          ; code 61       a
0787' 00              1714        defb 00000000b
0788' 38              1715        defb 00111000b
0789' 04              1716        defb 00000100b
078A' 3C              1717        defb 00111100b
078B' 44              1718        defb 01000100b
078C' 3C              1719        defb 00111100b
078D' 00              1720        defb 00000000b
                      1721
078E' 00              1722        defb 00000000b          ; code 62       b
078F' 20              1723        defb 00100000b
0790' 20              1724        defb 00100000b
0791' 38              1725        defb 00111000b
0792' 24              1726        defb 00100100b
0793' 24              1727        defb 00100100b
0794' 38              1728        defb 00111000b
0795' 00              1729        defb 00000000b
                      1730
0796' 00              1731        defb 00000000b          ; code 63       c
0797' 00              1732        defb 00000000b
0798' 1C              1733        defb 00011100b
0799' 20              1734        defb 00100000b
079A' 20              1735        defb 00100000b
079B' 20              1736        defb 00100000b
079C' 1C              1737        defb 00011100b
079D' 00              1738        defb 00000000b
                      1739
079E' 00              1740        defb 00000000b          ; code 64       d
079F' 08              1741        defb 00001000b
07A0' 08              1742        defb 00001000b
07A1' 38              1743        defb 00111000b
07A2' 48              1744        defb 01001000b
07A3' 48              1745        defb 01001000b
07A4' 38              1746        defb 00111000b
07A5' 00              1747        defb 00000000b
                      1748
07A6' 00              1749        defb 00000000b          ; code 65       e
07A7' 00              1750        defb 00000000b
07A8' 38              1751        defb 00111000b
                      1752        defb 01000100b
07A9' 44              1753        defb 01111000b
07AA' 78              1754        defb 01000000b
07AB' 40              1755        defb 00111100b
07AC' 3C              1756        defb 00000000b
07AD' 00              1757
                      1758        defb 00000000b          ; code 66       f
07AE' 00              1759        defb 00011000b
07AF' 18              1760        defb 00101000b
0780' 28              1761        defb 01110000b
0781' 70              1762        defb 00100000b
0782' 20              1763        defb 00100000b
0783' 20              1764        defb 00100000b
0784' 20              1765        defb 00000000b
0785' 00              1766
                      1767        defb 00000000b          ; code 67       g
0786' 00              1768        defb 00000000b
0787' 00              1769        defb 00111000b
0788' 38              1770        defb 01001000b
0789' 48              1771        defb 01001000b
078A' 48              1772        defb 00111000b
078B' 38              1773        defb 00001000b
078C' 08              1774        defb 00110000b
078D' 30              1775
                      1776        defb 00000000b          ; code 68       h
078E' 00              1777        defb 01000000b
078F' 40              1778        defb 01000000b
07C0' 40              1779        defb 01110000b
07C1' 70              1780        defb 01001000b
07C2' 48              1781        defb 01001000b
07C3' 48              1782        defb 01001000b
07C4' 48              1783        defb 00000000b
07C5' 00              1784
                      1785        defb 00020000b          ; code 69       i
07C6' 00              1786        defb 00010000b
07C7' 10              1787        defb 00000000b
07C8' 00              1788        defb 00110000b
07C9' 30              1789        defb 00010000b
07CA' 10              1790        defb 00010000b
07CB' 10              1791        defb 00111000b
07CC' 38              1792        defb 00000000b
07CD' 00              1793
                      1794        defb 00020000b          ; code 6A       j
07CE' 00              1795        defb 00001000b
07CF' 08              1796        defb 00000000b
07D0' 00              1797        defb 00001000b
07D1' 08              1798        defb 00001000b
07D2' 08              1799        defb 00001000b
07D3' 08              1800        defb 01001000b
07D4' 48              1801        defb 00110000b
07D5' 30              1802
                      1803        defb 00020000b          ; code 6B       k
07D6' 00
```

```
07C7* 28          1804          defb 00111000b
07D8* 28          1805          defb 00111000b
07D9* 38          1806          defb 00111000b
07DA* 38          1807          defb 00111000b
07DB* 28          1808          defb 00101000b
07DC* 24          1809          defb 00100100b
07DD* 00          1810          defb 00000000b
                  1811
07DE* 00          1812          defb 00000000b        ; code 6C     l
07DF* 20          1813          defb 00100000b
07E0* 20          1814          defb 00100000b
07E1* 20          1815          defb 00100000b
07E2* 20          1816          defb 00100000b
07E3* 20          1817          defb 00100000b
07E4* 18          1818          defb 00011000b
07E5* 00          1819          defb 00000000b
                  1820
07E6* 00          1821          defb 00000000b        ; code 6D     m
07E7* 00          1822          defb 00000000b
07E8* 28          1823          defb 00101000b
07E9* 54          1824          defb 01010100b
07EA* 54          1825          defb 01010100b
07EB* 54          1826          defb 01010100b
07EC* 54          1827          defb 01010100b
07ED* 00          1828          defb 00000000b
                  1829
07EE* 00          1830          defb 00000000b        ; code 6E     n
07EF* 00          1831          defb 00000000b
07F0* 78          1832          defb 01111000b
07F1* 48          1833          defb 01001000b
07F2* 48          1834          defb 01001000b
07F3* 48          1835          defb 01001000b
07F4* 48          1836          defb 01001000b
07F5* 00          1837          defb 00000000b
                  1838
07F6* 00          1839          defb 00000000b        ; code 6F     o
07F7* 00          1840          defb 00000000b
07F8* 38          1841          defb 00111000b
07F9* 44          1842          defb 01000100b
07FA* 44          1843          defb 01000100b
07FB* 44          1844          defb 01000100b
07FC* 38          1845          defb 00111000b
07FD* 00          1846          defb 00000000b
                  1847
07FE* 00          1848          defb 00000000b        ; code 70     p
07FF* 00          1849          defb 00000000b
0800* 78          1850          defb 01111000b
0801* 48          1851          defb 01001000b
0802* 48          1852          defb 01001000b
0803* 78          1853          defb 01111000b
0804* 40          1854          defb 01000000b
0805* 40          1855          defb 01000000b
                  1856
0806* 00          1857          defb 00000000b        ; code 71     q
0807* 00          1858          defb 00000000b
0808* 38          1859          defb 00111000b
0809* 48          1860          defb 01001000b
080A* 48          1861          defb 01001000b
080B* 38          1862          defb 00111000b
080C* 08          1863          defb 00001000b
080D* 0C          1864          defb 00001100b
                  1865
080E* 00          1866          defb 00000000b        ; code 72     r
080F* 00          1867          defb 00000000b
0810* 18          1868          defb 00011000b
0811* 20          1869          defb 00100000b
0812* 20          1870          defb 00100000b
0813* 20          1871          defb 00100000b
0814* 20          1872          defb 00100000b
0815* 00          1873          defb 00000000b
                  1874
0816* 00          1875          defb 00000000b        ; code 73     s
0817* 00          1876          defb 00000000b
0818* 38          1877          defb 00111000b
0819* 40          1878          defb 01000000b
081A* 38          1879          defb 00111000b
081B* 08          1880          defb 00001000b
081C* 78          1881          defb 01111000b
081D* 00          1882          defb 00000000b
                  1883
081E* 00          1884          defb 00000000b        ; code 74     t
081F* 20          1885          defb 00100000b
0820* 78          1886          defb 01111000b
0821* 20          1887          defb 00100000b
0822* 20          1888          defb 00100000b
0823* 20          1889          defb 00100000b
0824* 18          1890          defb 00011000b
0825* 00          1891          defb 00000000b
                  1892
0826* 00          1893          defb 00000000b        ; code 75     u
0827* 00          1894          defb 00000000b
0828* 48          1895          defb 01001000b
0829* 48          1896          defb 01001000b
082A* 48          1897          defb 01001000b
082B* 48          1898          defb 01001000b
082C* 38          1899          defb 00111000b
082D* 00          1900          defb 00000000b
                  1901
082E* 00          1902          defb 00000000b        ; code 76     v
082F* 00          1903          defb 00000000b
0830* 44          1904          defb 01000100b
0831* 44          1905          defb 01000100b
0832* 28          1906          defb 00101000b
0833* 28          1907          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
088E' EC                2011            defb 11101000b          ; code 82       graphics
```

```
088F' E0            2012          defb 11100000b
0890' E0            2013          defb 11100000b
0891' E0            2014          defb 11100000b
0892' 00            2015          defb 00000000b
0893' 00            2016          defb 00000000b
0894' 00            2017          defb 00000000b
0895' 00            2018          defb 00000000b
                    2019
0896' FC            2020          defb 11111100b        ; code 83       graphics
0897' FC            2021          defb 11111100b
0898' FC            2022          defb 11111100b
0899' FC            2023          defb 11111100b
089A' 00            2024          defb 00000000b
089B' 00            2025          defb 00000000b
089C' 00            2026          defb 00000000b
089D' 00            2027          defb 00000000b
                    2028
                    2029  ;
089E' 00            2030          defb 00000000b        ; code 84       graphics
089F' 00            2031          defb 00000000b
08A0' 00            2032          defb 00000000b
08A1' 00            2033          defb 00000000b
08A2' 1C            2034          defb 00011100b
08A3' 1C            2035          defb 00011100b
08A4' 1C            2036          defb 00011100b
08A5' 1C            2037          defb 00011100b
                    2038  ;
08A6' 1C            2039          defb 00011100b        ; code 85       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        ; code 86       graphics
08AF' E0            2050          defb 11100000b
08B0' E0            2051          defb 11100000b
08B1' E0            2052          defb 11100000b
08B2' 1C            2053          defb 00011100b
08B3' 1C            2054          defb 00011100b
08B4' 1C            2055          defb 00011100b
08B5' 1C            2056          defb 00011100b
                    2057  ;
08B6' FC            2058          defb 11111100b        ;code 87        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
08BC' 1C            2064          defb 00011100b
08BD' 1C            2065          defb 00011100b
                    2066  ;
08BE' E0            2067          defb 11100000b        ; code 88       graphics
08BF' E0            2068          defb 11100000b
08C0' E0            2069          defb 11100000b
08C1' E0            2070          defb 11100000b
08C2' E0            2071          defb 11100000b
08C3' E0            2072          defb 11100000b
08C4' E0            2073          defb 11100000b
08C5' E0            2074          defb 11100000b
                    2075  ;
08C6' 1C            2076          defb 00011100b        ; code 89       graphics
08C7' 1C            2077          defb 00011100b
08C8' 1C            2078          defb 00011100b
08C9' 1C            2079          defb 00011100b
08CA' E0            2080          defb 11100000b
08CB' E0            2081          defb 11100000b
08CC' E0            2082          defb 11100000b
08CD' E0            2083          defb 11100000b
                    2084  ;
08CE' E0            2085          defb 11100000b        ; code 8A       graphics
08CF' E0            2086          defb 11100000b
08D0' E0            2087          defb 11100000b
08D1' E0            2088          defb 11100000b
08D2' E0            2089          defb 11100000b
08D3' E0            2090          defb 11100000b
08D4' E0            2091          defb 11100000b
08D5' E0            2092          defb 11100000b
                    2093  ;
08D6' FC            2094          defb 11111100b        ; code 8B       graphics
08D7' FC            2095          defb 11111100b
08D8' FC            2096          defb 11111100b
08D9' FC            2097          defb 11111100b
08DA' E0            2098          defb 11100000b
08DB' E0            2099          defb 11100000b
08DC' E0            2100          defb 11100000b
08DD' E0            2101          defb 11100000b
                    2102  ;
08DE' 00            2103          defb 00000000b        ; code 8C       graphics
08DF' 00            2104          defb 00000000b
08E0' 00            2105          defb 00000000b
08E1' 00            2106          defb 00000000b
08E2' FC            2107          defb 11111100b
08E3' FC            2108          defb 11111100b
08E4' FC            2109          defb 11111100b
08E5' FC            2110          defb 11111100b
                    2111  ;
08E6' 1C            2112          defb 00011100b        ; code 8D       graphics
08E7' 1C            2113          defb 00011100b
08E8' 1C            2114          defb 00011100b
08E9' 1C            2115          defb 00011100b
```

```
08EA'  FC                  2116          defb 11111100b
08E9'  FC                  2117          defb 11111100b
08EC'  FC                  2118          defb 11111100b
08ED'  FC                  2119          defb 11111100b
                           2120  ;
08EE'  E0                  2121          defb 11100000b          ;code 8E        graphics
08EF'  E0                  2122          defb 11100000b
08F0'  E0                  2123          defb 11100000b
08F1'  E0                  2124          defb 11100000b
08F2'  FC                  2125          defb 11111100b
08F3'  FC                  2126          defb 11111100b
08F4'  FC                  2127          defb 11111100b
08F5'  FC                  2128          defb 11111100b
                           2129  ;
08F6'  FC                  2130          defb 11111100b          ;code 8F        graphics
08F7'  FC                  2131          defb 11111100b
08F8'  FC                  2132          defb 11111100b
08F9'  FC                  2133          defb 11111100b
08FA'  FC                  2134          defb 11111100b
08FB'  FC                  2135          defb 11111100b
08FC'  FC                  2136          defb 11111100b
08FD'  FC                  2137          defb 11111100b
                           2138        END
Additional input after END statement ignored
```

```
A       Reserved        ATTBYT  0039            ATTCTL  0011            B       Reserved        BOTLN   0020
C       Reserved        CALCP0  053C            CALCP1  0526            CALCP2  053A            CHNGVI  0E8E
CHRSET  047E            CHRST   0378            CHTBL   002F            CLINIT  1000            CLRCTL  000C
CLRSB0  0278            CLRSC0  003E IN         CLRSCN  027C IN         CLRSC0  0292            CLRSC1  0243
CLRSC2  02AE            CLRSC3  0282            CLRSC4  0285            CLRSC5  02C0            CLRSC6  02D5
CLRSC7  02E8            CLRSXT  02E7            CONVPM  04FC            CODROS  9C7D            CURPOS  0034
D       Reserved        DATAB   0000            DESTT   F7C0            DPADRS  0036            DPBIT   003E
DRMSPT  00F4            DRIVES  6940            E       Reserved        ENBLEX  0402            ENBLMC  040F
ERRRET  0144            EXTNRM  0447            FIX     97C0            FIXTBL  1000            GETAB0  03C0
GETATB  001E IN         GETATT  03C0 IN         GETC80  0304            GETCTL  0019            GETCUB  2021 IN
GETCUR  0304 IN         GETC1   03EC            GETVAL  044C            GOODRE  0142            GRALN   0240
GRPMST  077E            GRPST   0878            GRTBL   0031            GTCHAR  032A IN         GTCHB0  0326
GTCHR0  001B IN         GTCH1   033F            GTCH2   0343            GTCH22  0349            GTCH23  035F
GTCH3   0364            GTCH31  0367            GTCH32  0391            GTCH4   039A            GTCH5   03B6
GTCH6   03C6            GTCSHF  0379            GTC2B1  0368            GTC2B2  057C
GTINDX  003A            H       Reserved        GTC2BC  0960            INSDBL  0231            INSERT  12C0
INVPAR  0147            L       Reserved        HREXPT  00FF            LINCCL  0007            LINLEN  0033
LN9U    053F            LOOP    0205            LOOPB   0228            LOOP1   0210            M       Reserved
MARGIN  0030            MASKB   0038            MCVESI  0840            MSKCTL  0015            NEXT1   00C2
NEXT2   00EA            NEXT3   013A            NOINVE  0117            NORM1   00C9            NORM2   00F1
NORM3   0111            NOSTRG  0186            NOXOR   00CF            NXTSC   0226            PARAM6  5CC3
PARBPR  04F9            PRAMT   5C84            PSW     Reserved        RAMTOP  5C82            RECLEN  1720
SCINIT  1701            SCRCTL  0028            SCRL0   002A IN         SCRL80  01CA            SCRLCT  002E
SCRLXT  0237            SCRL0   01E4            SCRGLL  01CE IN         SCRSZ   0018            SETAB0  02F3
SETATB  0012 IN         SETATT  02F6 IN         SETC00  018A            SETCUB  0009 IN         SETCUR  018E IN
SETM00  031D            SETM0B  0025 IN         SETM01  0446            SETM02  046A            SETM00  03F6 IN
SETM5B  0016 IN         SETM5K  0320 IN         SET0    04CF            SP      Reserved        STBLK   0248
STBLK0  024C            STBLK1  0240            STBLK2  0264            STKEND  5C65            STM080  03F1
STPOSN  0550            STRGCT  0038            SXT4    0483            TESTAD  01F3            TSTPAR  0488
TSTPR1  04F4            TSTO1   04C7            TVPULQ  0148            TVPUL1  0167            UDG     5C7B
UPDATE  047E            UPJPOS  0507            VARS    5C4B            VIDMOD  5CC2            VIMODE  0024
WRCHAR  0042 IN         WRCHR8  0061 IN         WRCHXT  013F            WRCH0   003F            WRCH11  0041
WRCH12  0069            WRCH13  0060            WRCH14  0083            WRCH19  0084            WRCH2   008A
WRCH3   0095            WRCH5   0098            WRCH6   00AF            WRCH7   0124            WRSTR8  0004 IN
WRSTRG  0194 IN         WRSTR0  0171            WRSTR1  01JC            WRSTR2  018C            WRST43  0198
WRSTX1  01A7            WRSTX2  0183
```

No errors detected

Cross reference listing (M80P version 4.7)

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

ATTBYT    1348    6728     903     9068
ATTCTL     888     648    9078
BOTLN     1188     322    8448
CALCP1    1042   1048#
CALCP2    1059   1062#
CALCP0     698    1023    1031#
CHNGVI     42#     892
CHRSET     39#     126     957
CHRST      39    1128#
CHTBL     1268     173     699     8588
CLINIT     388     851
CLRCTL     84#     555    8528
CLRSB0     87     554#
CLRSC0     507     567    5718
CLRSC1    983#     632
CLRSC2    592#
CLRSC3    595#     641
CLRSC4    597#     626
CLRSC5    606#     620
CLRSC6     617     621#
CLRSC7     591     635#
CLRSC8      30      87#
CLRSCN      27     557#
CLRSXT     630     632#
CONVPM     415     697     811    1010#
CODROS      51#    913#
CURPOS    129#     810   10848   1089
DATAB      76#     147    8648
DESTT      59#      60
DPADRS    131#   10858   1090
DPBIT     141#     224     240     269     299    3018    3048
                   693    7608    7908   10638   1107
DRMSPT     46#    973#    979#
DRIVES     56#      57      58
ENBLEX    887    968#
ENBLMC    893    977#
ERRRET   3138     317     334     917
EXTNRM    882     886    9168
FIX        60#
FIXTBL     62#
```

```
GETA80    105     801#
GETAT8     31     105#
GETATT     28     903#
GETC1     814     820     8228
GETC80    106     808#
GETCTL    102#    691     8681    8698
GETCUB     31     104#
GETCUR     28     809#
GETVAL    889     938#
GOSORE   3128#    417     439     968     673     682     876
                  898     915
GRBLK     461     509#
GRPNST     40#    127     859
GRPST      40    1972#
GRTBL     127#    167     776     8608
GTC2B1   1104    1104#
GTC2B2   1110    1115#
GTC2BC    707     732    1098#
GTCH1     702#    779     786
GTCH2     704#    771
GTCH22    709#
GTCH23    720     723#
GTCH3     716     726#
GTCH31    728#    743
GTCH32    753     755    7578
GTCH4     722     725     742    7648
GTCH5     775     780#
GTCH6     781     789#
GTCHAR     28     693#
GTCHB0    104     688#
GTCHRB     31     104#
GTCSHP   11118   1114
GTINDX    136#    7029    718     750     773
HREXPT     45#    443     6668    969     9718    980     9828
INSOBL    462#
INSERT    574     877
INVPAR    155     157     316     432     435     437     968
                  364     566     655     838     993
LOPOSN    150    1088#
LINCOL     80#    411     8218    8658    8668
LINLEN    128#    182     445     575     812     8688    968
                  994    1010    1032
LNBU      448     578    1031    1067#
LOOP      465#    915
LOOP0     467#    494
LOOP1     474#    488
MARGIN    140#    8508   1051
MASKB     132#    196     6818    8548
MOVESZ     58#     59     878
MSKCTL     98#    678     8678
NEXT1     230#    233
NEXT2     256#    259
NEXT3     275#    278
NOINVB    268     285#
NORR1     228     234#
NORR2     254     260#
NORR3     273     279#
NOSTRG    353     401#
NOXOR     223     240#
NXTSC     485     489#
PARAM9     55#    346
PAREER    992#    997
PRAMT      53#
RAMTOP     52#    883
RECLEN     43#    357
SCINIT     37#    841
SCRCTL    111#    336     425     8428
SCRL0     338     438     4418
SCRL8      30     114#
SCRLB0    114     423#
SCRLCT    123#    326     3308    3358    8468
SCRLXT    498     501#    546
SCRCLL     27     427#
SCRSZ      36#    319     442     572    1014    1069
SET0      942     961#
SETAB0     97     647#
SETATB     30      97#
SETATT     27     650#
SETCB0     83     409#
SETCUB     30      83#
SETCUR     27     413#
SETMB0    101     677#
SETMD1    836     871#
SETMD2    874     887#
SETMDB     33     110#
SETMOD     33     833#
SETMSB     30     101#
SETMSK     27     680#
STBLK     453     517#
STBLK0    520#    540
STBLK1    521#    534
STBLK2    531     535#
STKEN0     49#    880
STMOB0    110     829#
STPOSN    311    1024    1083#
STRGCT    138#    3958    8558    8568
SXT4     8998
TESTAD    451#    500     547
TSTD1     940     956#
TSTPAR    414     694     989#
TSTPR1    991     994#
TVPULL    328     335#
TVPULS    191     319#
UDG        50#    163     782
UPDATE    897#
UPDPOS    324     416     634     914    1021#
VARS       48#    350
VIOMCO     54#    653     6688    872     8968
VIMODE    107#    831
```

```
WRCH0      77     1490
WRCH11    162     1670
WRCH12    160     1700
WRCH13    166     169      1710
WRCH14    183     1880
WRCH15    187     1890
WRCH2     1940
WRCH3     199     2010
WRCH5     2060    295
WRCH6     217     2190
WRCH7     2960
WRCHAR     27     1490     377
WRCHCB     30      778
WRCHIT    302     309      3110
WRSTR0     78     3460
WRSTR1    3510    360
WRSTR2    355     3610
WRSTR3    3620    393
WRSTR8     30      780
WRSTRG     27     3740     388
WRSTX1    380     3910
WRSTX2    3980    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/57344).

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 Nuaber 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) 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 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 (0-41)

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

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

- -

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
CMTBL	(Internal to Module)
GRTBL	(Internal to Module)
LINLEN	28H
CURPOS	1829H
DFADDR	4000H
GTINDX	80H
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	1800H
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 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 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.

```
                                1           NAME ADL - ASC 003 40-COL.MODE SUPPORT
                                2   ;
                                3   ;
                                4   ;
                                5   ;
                                6   ;
                                7   ;
                                8   ;
                                9   ;
                               10   ;               ********************************
                               11   ;               *                              *
                               12   ;               *          TIMEX               *
                               13   ;               * APPLICATION DEVELOPMENT LIBRARY *
                               14   ;               *                              *
                               15   ;               * NAME: 40-COL.MODE SUPPORT     *
                               16   ;               *                              *
                               17   ;               * ASC NO:  003                  *
                               18   ;               * VERSION: 001                  *
                               19   ;               * AUTHOR:  C. CORCORAN          *
                               20   ;               * DATE:    12/19/83             *
                               21   ;               ********************************
                               22           SUBTTL DEFINITIONS
                               23   ;
                               24   ;
                               25   ;                          ****DEFINITIONS****
                               26   ;
                               27           INTERN  WRCHAR, WRSTRG, SETCUR, CLRSCN, SCRCLL
                               28           INTERN  GTCHAR, GETATT, GETCUR
                               29   ;
                               30           INTERN  WRCHRB, WRSTRB, SETCUB, CLRSCB, SCRLB
                               31           INTERN  GTCHRB, GETATB, GETCUB
                               32   ;
                               33           INTERN  INIT40,INIT48
                               34   ;
                               35   ;
0018                           36   SCRSZ           EQU     24      ; 24 LINES
1701                           37   SCINIT          EQU     1701H   ; SCROLL CONTROL INITIALIZATION
1800                           38   CLINIT          EQU     1800H   ; CLEAR SCREEN CTL. INIT.
4429'                          39   CHRSET          EQU     ((CHRST)-100H)  ;CHAR.TABLE-100H
0729'                          40   GRPHST          EQU     ((GRPST)-100H) ; STD.GRAPHICS TABLE
                               41   ;
1720                           42   RECLEN          EQU     1720H   ; ROUTINE IN ROME ROM
                               43   ;
5C48                           44   VARS            EQU     5C48H   ; SYSTEM VARIABLES
5C78                           45   UDG             EQU     5C78H   ;
5C7A                           46   CHADDS          EQU     5C7AH   ;
5C8D                           47   ATTR_P          EQU     5C8DH   ;
5C8F                           48   MASK_P          EQU     5C8FH   ;
5C91                           49   P_FLAG          EQU     5C91H   ;
5CC2                           50   VIDADD          EQU     5CC2H   ;
5CC3                           51   PARAMS          EQU     5CC3H   ;  (Location 23747)
                               52
                               53           SUBTTL INPUTS AND ENTRIES
                               54   ;
                               55   ;
                               56   ;               **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                               57   ;
                               58   ;                                                       WITH
                               59   ;                                                    MODULE LOADED
                               60   ;                                                    AT 57344(E000H)
                               61   ;                                                    DECIMAL ADDRESS
                               62   ;                                                    ---------------
0000' 20                       63   DATAB           DEFB    20H     ; CHAR.CODE FROM BASIC          57344
0001' C3 0040'                 64   WRCHRB:         JP      WRCH0   ; ENTRY TO WRITE CHARACTER      57345
0304' C3 0169'                 65   WRSTRB:         JP      WRSTR0  ; ENTRY TO WRITE STRING         57348
0007' 0000                     66   LINCOL          DEFW    0       ; SET CURSOR PARAMETERS
                               67                                   ;   LINCOL=COL. (0-39)         57351
                               68                                   ;   LINCOL+1=LINE (0-23)       57352
0009' C3 0192'                 69   SETCUB:         JP      SITC80  ; ENTRY TO SET CURSOR POSN.     57353
000C' 1800                     70   CLRCTL          DEFW    1800H   ; CLEAR SCREEN CONTROLS
                               71                                   ;   CLRCTL=STARTING LINE NO.    57356
                               72                                   ;   CLRCTL+1=LINE COUNT        57357
000E' C3 024F'                 73   CLRSCB:         JP      CLRS80  ; ENTRY TO CLEAR SCREEN         57358
0011' 0000                     74   SPARE0          DEFW    0       ; NOT USED (ALIGNMENT ONLY)
0013' 0000                     75   SPARE1          DEFW    0       ;
0015' 0000                     76   SPARE2          DEFW    0       ;
0017' 0000                     77   SPARE3          DEFW    0       ;
0019' 0000                     78   GETCTL          DEFW    0       ; 'GET' CONTROL FROM BASIC      57369
                               79                                   ; FORMAT IS AS FOR LINCOL
001B' C3 02D7'                 80   GTCHRB:         JP      GTCH80  ; ENTRY TO GET CHARACTER        57371
001E' C3 037E'                 81   GETATB:         JP      GETA80  ; ENTRY TO GET ATTRIBUTE        57374
0021' C3 039C'                 82   GETCUB:         JP      GETC80  ; ENTRY TO GET CURSOR POSN.     57377
0024' 00                       83   SPARE4          DEFB    0       ; NOT USED                      57380
0025' C3 0389'                 84   INIT48          JP      INIT40  ; ENTRY TO INITIALIZE 40-COL.   57381
0028' 1701                     85   SCRCTL          DEFW    1701H   ; SCROLL CONTROL
                               86                                   ;   SCRCTL=STARTING LINE NO.    57384
                               87                                   ;   SCRCTL+1=LINE COUNT        57385
002A' C3 01A2'                 88   SCRLB:          JP      SCRL80  ; ENTRY TO SCROLL               57386
                               89           SUBTTL  OTHER VARIABLES
                               90   ;
002D' 17                       91   BOTLN           DEFB    17H     ; BOTTOM LINE (LINE AFTER       57389
                               92                                   ; WHICH SCREEN IS CONSIDERED
                               93                                   ; FULL.  AUTOMATIC SCROLL
                               94                                   ; WILL BE DONE IF SCRLCT
                               95                                   ; DOES NOT DECREMENT TO 0.
002E' 01                       96   SCRLCT          DEFB    1       ; SCROLL COUNT - 1 PLUS NO.     57390
                               97                                   ; OF TIMES AUTOMATIC SCROLL
                               98                                   ; WILL BE DONE.
002F' 0429'                    99   CHTBL           DEFW    CHRSET  ; CHARACTER TABLE               57391
0031' 0729'                   100   GRTBL           DEFW    GRPHST  ; GRAPHICS TABLE                57393
0033' 20                      101   LINLEN          DEFB     20H    ; LINE LENGTH (40/42)           57395
0034' 1829                    102   CURPOS          DEFW    1829H   ; CURRENT POSITION              57396
                              103   ;                             ; (INTERNAL FORMAT)
0036' 4000                    104   DFADRS          DEFW    4000H   ; CURRENT DISPLAY FILE ADDRS.   57398
0038' 00                      105   MASK0           DEFB    0       ; MASK TO BE APPLIED TO         57400
                              106   ;                             ; DISPLAY CHARACTERS
0039' 00                      107   ATTRVT          DEFB    0       ; CURRENT ATTRIBUTES            57401
                              108   ;                             ;
003A' 00                      109   GTINDX          DEFB    00H     ; INDEX FOR ADJUSTING           57402
```

```
                        114  ;
0038'  0000             115  STRGCT          DEFW    0       ; REMAINING COUNT FROM WRSTRG
                        116                                  ; WHEN 'SCREEN PULL'            57403
0030'  01               117  MARGIN          DEFB    1       ; MARGIN ADJUST (0/1)           57405
0038'  07               118  DPBIT           DEFB    7       ; BIT POSN.IN LINE(7,1,3,5)     57406
003F'  00               119  ATTMSK          DEFB    0       ; ATTRIBUTE MASK                57407
                        121          SUBTTL WRITE CHARACTER
                        122  ;
0040'                   123  WRCH0:                          ; HERE FROM BASIC ENTRY TO DISPLAY THE
                        124                                  ; CHARACTER WHOSE CODE IS IN "DATAB"
0040'  3A 0000'         125          LD   A,(DATAB)
                        126  ;
0043'                   127  WRCHAR:                         ; ENTRY WITH CODE IN A
0043'  CD 04FD'         128          CALL  LDATTR            ; GET ATTRIBUTE AND MASK CONTROLS
                        129                                  ;
                        130                                  ; ENTRY TO USE ATTRIBUTE VALUES IN
                        131                                  ; INTERNAL VARIABLES WITHOUT RELOADING
0046'  CD 0481'         132  WRCH01: CALL  LDPOSN           ; LOAD REGISTERS FOR CURRENT POSITION
                        133                                  ;   BC=CURSOR POSITION FROM 'CURPOS'
                        134                                  ;   HL=DISPLAY FILE ADDRESS FROM 'DPADRS
                        135                                  ;   A=CODE FOR CHAR. TO BE DISPLAYED
0049'  FE 20            136          CP    20H               ; TEST VALID RANGE
004B'  DA 013F'         137          JP    C,INVPAR          ; < 20H
004E'  FE A5            138          CP    0A5H              ;
0050'  D2 013F'         139          JP    NC,INVPAR           > 0A4H
0053'  C5               140          PUSH  BC                ; SAVE CURSOR POSITION
0054'  FE 80            141          CP    80H               ; TEST IF ASCII PRINTABLE (20H-7FH)
0056'  38 15            142          JR    C,WRCH12          ; YES
0058'  FE 90            143          CP    90H               ; TEST IF STD.GRAPHICS(80-8FH)
005A'  38 09            144          JR    C,WRCH11
005C'  ED 4B 9C7B       145          LD    BC,(UDG)          ; USER -DEFINED GRAPHICS(90-A4H)
0060'  05               146          DEC   B                 ; ADJUST ADDRESS-100H
0061'  D6 70            147          SUB   70H               ; ADJUST FOR INDEX INTO TABLE
0063'  18 0C            148          JR    WRCH13
0065'  ED 4B 0031'      149  WRCH11  LD    BC,(GRTBL)        ; ADDRESS OF GRAPHICS TABLE
0069'  D6 60            150          SUB   60H               ; ADJUST FOR INDEX INTO TABLE
006B'  18 04            151          JR    WRCH13
006D'  ED 4B 002F'      152  WRCH12  LD    BC,(CHTBL)        ; CHARACTER TABLE
0071'  E8               153  WRCH13  EX    DE,HL             ; DE=POSN.IN DISPLAY FILE
0072'  26 00            154          LD    H,0
0074'  6F               155          LD    L,A               ; CODE IS INDEX INTO TABLE
0075'  29               156          ADD   HL,HL
0076'  29               157          ADD   HL,HL
0077'  29               158          ADD   HL,HL
0078'  09               159          ADD   HL,BC             ; HL=PIXEL PATTERN IN TABLE
0079'  C1               160          POP   BC
007A'  E8               161          EX    DE,HL             ; DE=PIXEL PATTERN IN TABLE/HL=DP
007B'  79               162          LD    A,C               ; TEST IF END OF LINE
007C'  3D               163          DEC   A
007D'  3A 0033'         164          LD    A,(LINLEN)
0080'  28 05            165          JR    NZ,WRCH14
0082'  3C               166          INC   A                 ; START OF NEW LINE
0083'  05               167          DEC   B                 ; BUMP LINE NO.
0084'  4F               168          LD    C,A               ; LINLEN+1=START OF LINE
0085'  18 01            169          JR    WRCH15
0087'  3C               170  WRCH14  INC   A
0088'  B9               171  WRCH15  CP    C                 ; TEST IF START OF LINE
0089'  D5               172          PUSH  DE
008A'  CC 0163'         173          CALL  Z,TVPULQ          ; TEST IF PAST "BOTTOM" LINE
008D'  D1               174          POP   DE
                        175                                  ; HERE TO PUT CHARACTER IN DISPLAY FILE
008E'  C5               176  WRCH2   PUSH  BC                ; CURSOR POSITION
008F'  E5               177          PUSH  HL                ; DP ADDRESS
0090'  3A 0030'         178          LD    A,(MASK0)         ; GET OVER AND INVERSE CONTROLS
0093'  06 FF            179          LD    B,-1
0095'  1F               180          RRA
0096'  30 01            181          JR    C,WRCH3           ; IF XORING NEW INTO OLD
0098'  04               182          INC   B                 ; B=-1 IF XORING  =0 IF NOT
0099'  1F               183  WRCH3   RRA
009A'  1F               184          RRA
009B'  9F               185          SBC   A,A
009C'  4F               186          LD    C,A               ; C=-1 FOR INVERSE  =0 IF NOT
009D'  3E 08            187          LD    A,8               ; SCAN COUNT
009F'  08               188  WRCH5   EX    AF,AF'            ; SAVE SCAN COUNT IN ALTERNATE
00A0'  C5               189          PUSH  BC                ; SAVE MASK VALUES
00A1'  D5               190          PUSH  DE                ; SAVE CHAR.TABLE ADRS
00A2'  DD 21 0000       191          LD    IX,0              ; SAVE PTR.TO STACK
00A6'  DD 39            192          ADD   IX,SP
00A8'  56               193          LD    D,(HL)            ; READ SCAN LINE FROM DP
00A9'  E5               194          PUSH  HL                ; SAVE ADDRESS
00AA'  23               195          INC   HL                ; NEXT BYTE
00AB'  5E               196  WRCH6   LD    E,(HL)            ; READ SCAN LINE FROM NEXT BYTE
00AC'  EB               197          EX    DE,HL             ; SCANS TO HL. LAST DP PCS.IN DE
00AD'  78               198          LD    A,B               ; TEST IF "OVER" IS ACTIVE
00AE'  A7               199          AND   A
00AF'  20 1A            200          JR    NZ,NOXOR          ; YES-RETAIN OLD DATA
00B1'  3A 003E'         201          LD    A,(DPBIT)         ; NO- CLEAR OLD DATA AT CHAR.POSN.
00B4'  D6 07            202          SUB   7                 ;
00B6'  ED 44            203          NEG
00B8'  01 03FF          204          LD    BC,03FFH          ; MASK
00BB'  28 08            205          JR    Z,NORR1           ; NO NEED TO ADJUST(CHAR.STARTS AT BIT 7)
00BD'  37               206          SCF
00BE'  CB 18            207  NEXT1   RR    B                 ; SHIFT MASK 'A' TIMES
00C0'  CB 19            208          RR    C                 ; (6 0'S START AT BIT 1 OF B, BIT 3 OF C.
00C2'  3D               209          DEC   A                 ;  OR BIT 9 OF B)
00C3'  20 F9            210          JR    NZ,NEXT1
00C5'  7C               211  NORR1   LD    A,H               ; SCAN LINE FROM DP
00C6'  A0               212          AND   B
00C7'  67               213          LD    H,A               ; RETAIN ADJACENT CHAR.PIXELS
00C8'  7D               214          LD    A,L               ; CLEAR CURRENT CHAR.PIXELS
00C9'  A1               215          AND   C
00CA'  6F               216          LD    L,A               ; HL=OLD BIT ROW
00CB'  3A 003E'         217  NOXOR   LD    A,(DPBIT)         ; GET STARTING BIT POSN.
00CE'  D6 07            218          SUB   7
00D0'  ED 44            219          NEG
00D2'  E5               220          PUSH  HL                ; SAVE SCANS
00D3'  DD 6E 00         221          LD    L,(IX)            ; PTR. TO CHAR.SET
00D6'  DD 66 01         222          LD    H,(IX+1)
00D9'  46               223          LD    B,(HL)            ; SCAN LINE FROM CHAR.SET TO BC
00DA'  08 00            224          LD    C,0               ; (PIXELS IN B, BITS 7-2)
```

```
00DC'  F5                  229          PUSH  AF              ; SAVE A AND FLAGS
00DD'  78                  226          LD    A,B             ; PIXELS FROM CHAR.SET
00DE'  E6 FC               227          AND   0FCH            ; LIMIT TO 6 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 08               231          JR    Z,NORR2         ; NO SHIFT NEEDED
00E5'  A7                  232          AND   A               ; CLEAR CARRY
00E6'  CB 18               233  NEXT2   RR    B               ; SHIFT "A" TIMES
00E8'  CB 19               234          RR    C
00EA'  3C                  239          DEC   A
00EB'  20 F9               236          JR    NZ,NEXT2
00ED'  7C                  237  NORR2   LD    A,H             ; SCAN FROM DP
00EE'  A8                  238          XOR   B               ; INSERT/COMBINE NEW CHAR.
00EF'  67                  239          LD    H,A
00F0'  7D                  240          LD    A,L
00F1'  A9                  241          XOR   C
00F2'  6F                  242          LD    L,A             ; DP XOR CH.SET->HL
00F3'  CC 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,(OPBIT)
00FC'  D6 07               247          SUB   7
00FE'  ED 44               248          NEG
0100'  01 FC00             249          LD    BC,0FC00H       ; MASK TO INVERT
0103'  28 08               250          JR    Z,NORR3
0105'  A7                  251          AND   A
0106'  CB 18               252  NEXT3   RR    B               ; SHIFT MASK "A" TIMES
0108'  CB 19               253          RR    C
010A'  3C                  254          DEC   A
010B'  20 F9               255          JR    NZ,NEXT3
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'  EB                  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'  24                  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,NRCH3
0120'  25                  273  NRCH7   DEC   H               ; LAST CHAR.SCAN
0121'  CD 04A9'            274          CALL  UPDATT          ; UPDATE ATTRIBUTE BYTE
0124'  E1                  275          POP   HL              ; STARTING ADRS.
0125'  C1                  276          POP   BC              ; CURSOR POSITION
0126'  0D                  277          DEC   C               ; ADJUST CURSOR POSITION
0127'  3A 003E'            278          LD    A,(OPBIT)       ; ADJUST BIT POSITION
012A'  D6 06               279          SUB   6
012C'  32 003E'            280          LD    (OPBIT),A       ; STORE UPDATED POSITION
012F'  30 06               281          JR    NC,NRCHNT       ; NEXT CHAR.IN SAME BYTE
0131'  C6 08               282          ADD   A,8             ; NEXT CHAR.IN NEXT BYTE
0133'  32 003E'            283          LD    (OPBIT),A       ; STORE ADJUSTED VALUE
0136'  23                  284          INC   HL              ; ADJUST DP ADRS.
0137'  CD 0499'            285  NRCHNT  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 F9               291          JR    ERRRET
                           292  ;
0143'  3E 18               293  TVFULG  LD    A,SCRSZ         ; 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 0492'            298          JP    NC,UPDPOSN      ; ON SCREEN - RETURN TO WRITE CHAR.
                           299                                ;  VIA UPDATE AND STORE POSITION
014E'  3A 002E'            300          LD    A,(SCRLCT)      ; TEST SCROLL COUNT
0151'  3D                  301          DEC   A
0152'  28 08               302          JR    NZ,TVFUL1       ; DO AUTOMATIC SCROLL USING SCRCTL
0154'  3E 01               303          LD    A,1             ; REINITIALIZE TO 1
0156'  32 002E'            304          LD    (SCRLCT),A
0159'  C1                  305          POP   BC
015A'  C1                  306          POP   BC              ; DISCARD RETURN TO WRITE CHAR.
015B'  0E 03               307          LD    C,3             ; RETURN BC=3 FOR SCREEN FULL
015D'  18 DD               308          JR    ERRRET
015F'  32 002E'            309  TVFUL1  LD    (SCRLCT),A      ; UPDATE VARIABLE
0162'  ED 4B 0028'         310          LD    BC,(SCRCTL)     ; GET SCROLL CONTROLS (STARTING LINE
                           311                                ;   AND LINE COUNT)
0166'  C3 018C'            312          JP    SCRL0           ; RETURN TO WRITE CHAR. VIA SCROLL
                           314          SUBTTL WRITE STRING
                           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 040E'            319  WRSTRG  CALL  GETSTRG        ; GET STRING VAR.ADRS. TO HL
016C'  C8                  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 04FD'            327  WRSTRG  CALL  LDATTR          ; GET ATTRIBUTE AND MASK CONTROLS
0170'  7E                  328  WRSTLP  LD    A,(HL)          ; GET CODE
0171'  E5                  329          PUSH  HL              ; SAVE ADRS. AND
0172'  C5                  330          PUSH  BC              ; COUNT
0173'  CC 0046'            331          CALL  WRC=01          ; WRITE "A"
0176'  79                  332          LD    A,C             ; TEST IF GOOD
0177'  B8                  333          OR    B
0178'  20 09               334          JR    NZ,WRSTX1       ; EXIT IF INVALID CODE OR
                           335                                ; IF SCREEN FULL
017A'  C1                  336  WRSTR3  POP   BC              ; COUNT
```

```
0178'  81                  337           POP   HL              ; ADDRESS
017C'  23                  338           INC   HL              ; NEXT CHAR.
017D'  08                  339           DEC   BC              ; ADJUST COUNT
017E'  78                  340           LD    A,B             ; TEST IF DONE
017F'  B1                  341           OR    C
0180'  20 EE               342           JR    NZ,WRSTLP       ; WRITE NEXT CHAR.
0182'  C9                  343           RET                   ; RETURN BC=0
                           344   ;
0183'  79                  345   WRSTX1  LD    A,C             ; SAVE ERROR
0184'  FE 01               346           CP    1               ; TEST IF UNRECOGNIZED CODE
0186'  28 F2               347           JR    Z,WRSTR3        ; CONTINUE
0188'  E1                  348           POP   HL              ; REMAINING COUNT TO HL
0189'  22 6038'            349           LD    (STRGCT),HL     ; STORE REMAINING COUNT
018C'  E1                  350           POP   HL              ; ADDRESS TO HL
018D'  0E 03               351           LD    C,3             ; RETURN BC=3 FOR SCREEN FULL
018F'  06 00               352           LD    B,0             ;
0191'  C9                  353           RET
                           354   ;
                           355
                           356   ;
                           357   ;
                           358   ;
                           360           SUBTTL POSITION CONTROL
                           361   ;
0192'                      362   SETCB0: ;                     ; HERE FROM BASIC ENTRY.  PARAMETERS
                           363   ;                             ; IN LINCOL
0192'  ED 4B 0007'         364           LD    BC,(LINCOL)
                           365   ;
0196'                      366   SETCUR: ;                     ; ENTRY WITH LINE/COL. IN BC REG.
0196'  CD 0436'            367           CALL  TSTPAR          ; TEST PARAMETERS FOR VALIDITY
0199'  CD 0447'            368           CALL  CONVPM          ; CONVERT TO INTERNAL FORMAT
019C'  CD 0452'            369           CALL  UPDPOSN         ; CALCULATE AND STORE POSITION
019F'  C3 013A'            370           JP    GOODRET         ; RETURN BC=0
                           371   ;
                           373           SUBTTL SCROLL
                           374   ;
                           375   ;
01A2'                      376   SCRLB0: ;                     ; HERE FROM BASIC ENTRY TO SCROLL
                           377   ;                             ; SCREEN
01A2'  ED 4B 0028'         378           LD    BC,(SCRCTL)     ; GET CONTROL INFO.
                           379   ;
01A6'                      380   SCRCLL: ;                     ; HERE WITH CONTROLS IN BC
                           381   ;                             ;  B=NO. OF LINES
                           382   ;                             ;  C=STARTING LINE NO.
01A6'  78                  383           LD    A,B             ; TEST VALIDITY
01A7'  A7                  384           AND   A
01A8'  CA 013F'            385           JP    Z,INVPAR        ; ERROR IF COUNT=0
01AB'  81                  386           ADD   A,C
01AC'  FE 01               387           CP    1
01AE'  DA 013F'            388           JP    C,INVPAR        ; ERROR IF B+C<1
01B1'  FE 19               389           CP    25
01B3'  D2 013F'            390           JP    NC,INVPAR       ; ERROR IF B+C>24
01B6'  CD 01BC'            391           CALL  SCRL0           ; DO SCROLL
01B9'  C3 013A'            392           JP    GOODRET         ; RETURN BC=0
                           393   ;
01BC'  C5                  394   SCRL0   PUSH  BC
01BD'  3E 18               395           LD    A,SCRSZ         ; GET STARTING LINE IN INTERNAL FORMAT
01BF'  91                  396           SUB   C
01C0'  47                  397           LD    B,A
01C1'  3A 0033'            398           LD    A,(LINLEN)
01C4'  3C                  399           INC   A               ; COL. 0
01C5'  4F                  400           LD    C,A
01C6'  CD 0478'            401           CALL  LNBU
01C9'  C1                  402           POP   BC
01CA'  E5                  403           PUSH  HL              ; SAVE STARTING ADRS.
01CB'  C5                  404           PUSH  BC              ; SAVE ORIG.BC FOR EXIT
01CC'  7D                  405   TESTAC  LD    A,L             ; TEST IF AT START OF BLOCK
01CD'  A7                  406           AND   A
01CE'  28 5C               407           JR    Z,STBLK         ; YES
01D0'  07                  408           RLCA
01D1'  07                  409           RLCA
01D2'  07                  410           RLCA
01D3'  4F                  411           LD    C,A             ; GET NO. OF LINES THIS BLOCK
01D4'  3E 08               412           LD    A,8
01D6'  91                  413           SUB   C
01D7'  B8                  414           CP    B               ; TEST AGAINST TOTAL LINES
01D8'  38 4A               415           JR    C,GRBLK         ; TOTAL GREATER THAN THIS BLOCK
01DA'  78                  416   INSDBLK LD    A,B
01DB'  06 00               417           LD    B,0             ; REMAINING COUNT
01DD'  C5                  418           PUSH  BC
01DE'  47                  419   LOOP    LD    B,A             ; B=LINES THIS BLOCK
01DF'  0E 08               420           LD    C,8             ; C=SCAN COUNT
01E1'  78                  421   LOOP0   LD    A,B
01E2'  C5                  422           PUSH  BC              ; SAVE SCAN COUNT
01E3'  0F                  423           RRCA
01E4'  0F                  424           RRCA
01E5'  0F                  425           RRCA                  ; CALCULATE NO. OF BYTES
01E6'  4F                  426           LD    C,A
01E7'  06 00               427           LD    B,0             ; BC=32** OF LINES
01E9'  E8                  428   LOOP1   EX    DE,HL
01EA'  21 FFE0             429           LD    HL,-20H         ; GET ADRS. OF PREV. LINE
01ED'  19                  430           ADD   HL,DE
01EE'  EB                  431           EX    DE,HL           ; TO DE
01EF'  E5                  432           PUSH  HL              ; SAVE OF ADRS.
01F0'  ED B0               433           LDIR                  ; DO MOVE
01F2'  E1                  434           POP   HL              ; GET ADRS.
01F3'  24                  435           INC   H               ; NEXT SCAN LINE
01F4'  C1                  436           POP   BC              ; SCAN COUNT
01F5'  0D                  437           DEC   C
01F6'  20 E9               438           JR    NZ,LOOP0        ; MORE SCANS
01F8'  C1                  439           POP   BC              ; RESIDUAL LINE COUNT
01F9'  78                  440           LD    A,B
01FA'  A7                  441           AND   A
01FB'  28 04               442           JR    Z,SCRLXT        ; DONE
01FD'  2E 00               443           LD    L,0             ; START OF NEXT BLOCK
01FF'  18 C8               444           JR    TESTAD          ; DO REMAINING LINE(S)
0201'  C1                  445   SCRLXT  POP   BC              ; ORIG.BC
0202'  D1                  446           POP   DE              ; STARTING ADRS.
0203'  C5                  447           PUSH  BC              ; SAVE ORIG.BC
0204'  68                  448           LD    L,B             ; NO. OF LINES SCROLLED
0205'  26 00               449           LD    H,0             ;
0207'  29                  450           ADD   HL,HL           ; 32*NO. OF LINES=
0208'  29                  451           ADD   HL,HL           ; NO OF ATTRIBUTE BYTES
0209'  29                  452           ADD   HL,HL           ; TO BE SCROLLED
020A'  29                  453           ADD   HL,HL           ;
```

```
0208*  29                    454             ADD   HL,HL             !
020C*  44                    455             LD    B,H              ! COUNT TO BC
020D*  4D                    456             LD    C,L              !
020E*  62                    457             LD    H,D              ! ADRS. TO HL
020F*  6B                    458             LD    L,E              !
0210*  CD 04E0*              459             CALL  CALCATT          ! ATTRIBUTE ADRS. TO HL
0213*  EB                    460             EX    DE,HL            !
0214*  21 FFE0               461             LD    HL,-20H          ! ADRS. OF PREV.ATTR.BYTE
0217*  19                    462             ADD   HL,DE            !
0218*  EB                    463             EX    DE,HL            ! PREV.BYTE DE
0219*  ED B0                 464             LDIR                   ! SCROLL ATTRIBUTES
021B*  C1                    465             POP   BC               ! SWAG. BC
021C*  79                    466             LD    A,C              ! STARTING LINE
021D*  80                    467             ADD   A,B              ! ADD NO. OF LINES MOVED
021E*  3D                    468             DEC   A                ! LAST LINE SCROLLED
021F*  4F                    469             LD    C,A              !
0220*  06 01                 470             LD    B,1              ! CLEAR 1 LINE
0222*  18 45                 471             JR    CLRSC0           ! CLEAR VACATED LINE AND RETURN
                             472     !
0224*  4F                    473     SRBLK   LD    C,A              !
0225*  78                    474             LD    A,B              !
0226*  91                    475             SUB   C                ! ADJUST TOTAL LINE COUNT BY NO.
0227*  47                    476             LD    B,A              ! OF LINES THIS BLOCK
0228*  C5                    477             PUSH  BC               !
0229*  79                    478             LD    A,C              ! LOAD A NO. OF LINES THIS BLOCK
022A*  18 02                 479             JR    LBBP             !
                             480     !
022C*  05                    481     STBLK   DEC   B                ! ADJUST TOTAL LINE COUNT-1
022D*  C5                    482             PUSH  BC               !
022E*  0E 00                 483             LD    C,0              ! SCAN COUNT
0230*  C5                    484     STBLK0  PUSH  BC               ! SAVE SCAN COUNT
0231*  EB                    485     STBLK1  EX    DE,HL            !
0232*  21 F880               486             LD    HL,-780H         !
0235*  19                    487             ADD   HL,DE            !
0236*  EB                    488             EX    DE,HL            ! ADRS. OF PREV. LINE
0237*  01 0020               489             LD    BC,32            ! BYTE COUNT
023A*  E5                    490             PUSH  HL               !
023B*  ED B0                 491             LDIR                   ! BB MOVE
023D*  E1                    492             POP   HL               !
023E*  24                    493             INC   H                ! TO NEXT SCAN LINE
023F*  C1                    494             POP   BC               !
0240*  0C                    495             DEC   C                ! TEST IF MORE SCANS
0241*  20 E8                 496             JR    NZ,STBLK0        !
0243*  C1                    497             POP   BC               ! REMAINING LINE COUNT
0244*  78                    498             LD    A,B              !
0245*  A7                    499             AND   A                !
0246*  28 09                 500             JR    Z,SCRLRT         !
0248*  11 F820               501             LD    DE,-7E0H         ! ADJUST TO LINE 1
024B*  19                    502             ADD   HL,DE            !
024C*  C3 01CC*              503             JP    TESTAD           ! BC=LINE COUNT
                             504                                    ! HL=DP ADDRESS
                             505     .
                             507             SUBTTL CLEAR SCREEN
                             508     !
                             509     !
024F*                        510     CLRSB01                        ! HERE FROM BASIC ENTRY TO CLEAR SCREEN
0250*  ED 4B 000C*           511             LD    BC,(CCLRCTL)     ! GET CONTROL INFO.
                             512     !
0253*                        513     CLRSCN1                        ! ENTRY WITH BC=LINE COUNT AND
                             514                                    ! STARTING LINE NO. FOR CLEAR
0253*  78                    515             LD    A,B              !
0254*  A7                    516             AND   A                ! TEST VALIDITY
0255*  CA 013D*              517             JP    Z,INVPAR         ! ERROR IF COUNT=0
0258*  81                    518             ADD   A,C              !
0259*  FE 01                 519             CP    1                !
025B*  DA 013D*              520             JP    C,INVPAR         ! ERROR IF B+CC1
025E*  FE 19                 521             CP    25               !
0260*  D2 013D*              522             JP    NC,INVPAR        ! ERROR IF B+C>24
0263*  CD 0269*              523             CALL  CLRSC0           ! DO SERVICE
0266*  C3 013A*              524             JP    GOCCRET          ! RETURN BC=0
                             525     !
                             526     !
0269*  CD 04FB*              527     CLRSC0  CALL  LOATYB           ! GET ATTRIBUTE CTL
026C*  C5                    528             PUSH  BC               ! ORIGINAL BC
026D*  3E 18                 529             LD    A,SCRSZ          ! CONVERT TO INTERNAL FORMAT
026F*  91                    530             SUB   C                !
0270*  47                    531             LD    B,A              !
0271*  3A 0033*              532             LD    A,(LINLEN)       !
0274*  3C                    533             INC   A                ! SET TO CCL.0
0275*  4F                    534             LD    C,A              !
0276*  CD 0478*              535             CALL  LNOU             ! GET ADDRESS
0279*  50                    536             LD    D,B              !
027A*  59                    537             LD    E,C              ! SAVE POSITION
027B*  C1                    538             POP   BC               ! ORIG. BC
027C*  D5                    539             PUSH  DE               ! SAVE CURSOR POSITION
027D*  78                    540     CLRSC1  LD    A,L              ! GET # OF LINES THIS BLOCK
027E*  07                    541             RLCA                   !
027F*  07                    542             RLCA                   !
0280*  07                    543             RLCA                   !
0281*  4F                    544             LD    C,A              !
0282*  3E 00                 545             LD    A,0              !
0284*  91                    546             SUB   C                !
0285*  B8                    547             CP    B                ! COMPARE TO TOTAL LINE COUNT
0286*  38 47                 548             JR    C,CLRSC7         !
0288*  78                    549     CLRSC2  LD    A,B              ! NO. OF LINES
0289*  06 00                 550             LD    B,0              ! REMAINING LINES
028B*  C5                    551             PUSH  BC               !
028C*  47                    552     CLRSC3  LD    B,A              !
028D*  0E 00                 553             LD    C,0              ! SCAN COUNT
028F*  78                    554     CLRSC4  LD    A,B              !
0290*  C5                    555             PUSH  BC               ! SAVE LINE CNT./SCAN CNT.
0291*  E6 07                 556             AND   7                !
0293*  0F                    557             RRCA                   !
0294*  0F                    558             RRCA                   !
0295*  0F                    559             RRCA                   !
0296*  4F                    560             LD    C,A              ! BC=32*NO. OF LINES
0297*  06 00                 561             LD    B,0              ! (=0 IF 256 FOR 8 LINES)
0299*  0D                    562             DEC   C                ! ADJUST
029A*  E5                    563     CLRSC5  PUSH  HL
```

```
0298'  54                    564          LD    D,H
029C'  5D                    565          LD    E,L
029D'  36 00                 566          LD    (HL),00          ; CLEAR DF
029F'  13                    567          INC   DE
02A0'  ED B0                 568          LDIR
02A2'  E1                    569          POP   HL
02A3'  24                    570          INC   H                ; NEXT SCAN ROW
02A4'  C1                    571          POP   BC               ; POP LINE/SCAN COUNT
02A5'  0C                    572          DEC   C                ; DECR. SCAN COUNT
02A6'  20 E7                 573          JR    NZ,CLRSC4        ; NEXT SCAN
02A8'  E5                    574          PUSH  HL               ; SAVE NEXT DF ADRS.
02A9'  25                    575          DEC   H                ; LAST SCAN OF CHAR.
02AA'  CD 04ED'              576          CALL  CALCATT          ; GET ADRS. IN ATTRIBUTE FILE
02AD'  78                    577          LD    A,B              ; NO. OF LINES THIS BLOCK
02AE'  E6 07                 578          AND   7
02B0'  0F                    579          RRCA
02B1'  0F                    580          RRCA
02B2'  0F                    581          RRCA
02B3'  4F                    582          LD    C,A              ; BC=32*NO. OF LINES
02B4'  06 00                 583          LD    B,0
02B6'  0D                    584          DEC   C                ; ADJUST
02B7'  54                    585          LD    D,H
02B8'  5D                    586          LD    E,L
02B9'  3A 0039'              587          LD    A,(ATTBYT)       ; ATTRIBUTE VALUE
02BC'  77                    588          LD    (HL),A           ; UPDATE ATTRIBUTE FILE
02BD'  13                    589          INC   DE               ;
02BE'  ED B0                 590          LDIR                   ;
02C0'  E1                    591          POP   HL               ; DF ADRS.(LAST SCAN+100H)
02C1'  C1                    592          POP   BC               ; TOTAL LINE COUNT
02C2'  78                    593          LD    A,B
02C3'  A7                    594          AND   A
02C4'  28 05                 595          JR    Z,CLRSXT
02C6'  2E 00                 596          LD    L,0              ; HL=START OF NEXT BLOCK
02C8'  C3 027D'              597          JP    CLRSC1           ; B=REMAINING LINE COUNT
02CB'  C1                    598  CLRSXT  POP   BC               ; CURSOR POSITION
02CC'  C3 0452'              599          JP    UPDPOSN          ; RETURN VIA UPDATE AND STORE POSITION
02CF'  4F                    600  CLRSC7  LD    C,A              ; SAVE NO. LINES THIS BLOCK
02D0'  78                    601          LD    A,B
02D1'  91                    602          SUB   C                ; ADJUST TOTAL LINE COUNT
02D2'  47                    603          LD    B,A
02D3'  C5                    604          PUSH  BC               ; REMAINING LINE COUNT
02D4'  79                    605          LD    A,C              ; LINES THIS BLOCK
02D5'  18 B5                 606          JR    CLRSC3           ;
                             607  ;
                             609          SUBTTL 'GET' ROUTINES
                             610  ;
                             611  ;
02D7'                        612  GTCH00:                        ; HERE FROM BASIC ENTRY TO GET CHARACTER
                             613                                 ; (RETURNS CODE FOR CHARACTER AT
                             614                                 ;  POSITION SPECIFIED IN GETCTL)
02D7'  ED 4B 0019'           615          LD    BC,(GETCTL)      ; GET CONTROL INFO.
                             616  ;
02DB'                        617  GTCHAR:                        ; ENTRY WITH POSITION IN BC
02DB'  CD 0436'              618          CALL  TSTPAR           ; TEST PARAMETERS
02DE'  3A 003E'              619          LD    A,(CPBIT)        ; SAVE CURRENT BIT POSITION
02E1'  F5                    620          PUSH  AF
02E2'  CD 0447'              621          CALL  CONVPM           ; CONVERT TO INTERNAL FORMAT
02E5'  CD 0457'              622          CALL  CALCPOS          ; GET DF ADDRESS AND BIT POS.
02E8'  ED 5B 002F'           623          LD    DE,(CHTBL)       ; CHAR.TABLE
02EC'  06 60                 624          LD    B,96             ; NO. OF PRINTABLE CHARACTERS
02EE'  3E 00                 625          LD    A,0CH            ; SET ADJUSTMENT INDEX
02F0'  32 0034'              626  GTCH1   LD    (GTINDX),A
02F3'  14                    627          INC   D                ; ADJUST TO START OF TABLE
02F4'  C5                    628  GTCH2   PUSH  BC               ; SAVE COUNT
02F5'  E5                    629          PUSH  HL               ; SAVE ADRS. IN DF
02F6'  D5                    630          PUSH  DE               ; SAVE ADRS. IN CHAR.TABLE
02F7'  CD 0513'              631          CALL  GTC28C           ; GET SCAN LINE FROM DF TO REG.B
                             632                                 ; BITS 7-2
02FA'  78                    633  GTCH22  LD    A,B
02FB'  E6 FC                 634          AND   0FCH             ; 6 BITS TO A
02FD'  47                    635          LD    B,A              ; DF CHAR.SCAN ROW IN B
02FE'  EB                    636          EX    DE,HL            ; CHAR.SET PTR. TO HL
02FF'  7E                    637          LD    A,(HL)           ; PIXEL ROW FROM CHAR.SET
0300'  E6 FC                 638          AND   0FCH             ; MASK TO 6 BITS
0302'  A8                    639          XOR   B                ; TEST AGAINS DF IMAGE
0303'  28 10                 640          JR    Z,GTCH3          ; MATCH
0305'  F5                    641          PUSH  AF
0306'  3A 003A'              642          LD    A,(GTINDX)       ; TEST IF LOOKING AT STD.GRAPHICS
0309'  FE 90                 643          CP    90H
030B'  20 03                 644          JR    NZ,GTCH23
030D'  F1                    645          POP   AF
030E'  18 38                 646          JR    GTCH4            ; DO NOT TEST INVERSE IF STD.GR.
0310'  F1                    647  GTCH23  POP   AF
0311'  FE FC                 648          CP    0FCH             ; TEST IF INVERSE
0313'  20 36                 649          JR    NZ,GTCH4         ; DOES NOT MATCH
0315'  4F                    650  GTCH3   LD    C,A              ; C=0 FOR MATCH  -4 FOR INVERSE
0316'  06 07                 651          LD    B,7              ; SCAN COUNT
0318'  EB                    652  GTCH31  EX    DE,HL            ; DF ADRS.TO HL
0319'  C5                    653          PUSH  BC       ; SAVE MATCH AND SCAN COUNT
031A'  24                    654          INC   H        ; NEXT SCAN IN DF
031B'  13                    655          INC   DE       ; NEXT ROW IN CHAR.SET
031C'  CD 0513'              656          CALL  GTC28C           ; NEXT SCAN ROW TO BC
031F'  78                    657          LD    A,B
0320'  E6 FC                 658          AND   0FCH
0322'  47                    659          LD    B,A
0323'  EB                    660          EX    DE,HL            ; CHAR.SET PTR. TO HL
0324'  7E                    661          LD    A,(HL)           ; PIXEL ROW TO A
0325'  E6 FC                 662          AND   0FCH
0327'  A8                    663          XOR   B                ; TEST MATCH
0328'  C1                    664          POP   BC               ; GET PREV.MATCH AND SCAN COUNT
0329'  A9                    665          XOR   C
032A'  20 1F                 666          JR    NZ,GTCH4         ; NO MATCH-START AGAIN AT NEXT
032C'  10 EA                 667          DJNZ  GTCH31           ; DO NEXT SCAN
032E'  79                    668          LD    A,C              ; A=-4 IF MATCH ON INVERSE
032F'  C6 03                 669          ADD   A,3              ; SET TO -1 IF MATCH ON INVERSE
0331'  C1                    670          POP   BC               ; HERE IF MATCH- (SP)=PTR.TO CHAR.
0332'  C1                    671          POP   BC               ; IN TABLE1(SP+2)=CHAR.IN DF;
0333'  C1                    672          POP   BC               ; (SP+4)=CHAR.COUNT
0334'  4F                    673          LD    C,A              ; B=COUNT C=-1 IF MATCH ON INVERSE
0335'  3A 003A'              674          LD    A,(GTINDX)       ; CONVERT MATCH TO CHAR.CODE
0338'  90                    675          SUB   B
0339'  FE 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    NZ,GTCH32
```

```
0340'  3E 0F              680          LD    A,0FH             ; LOAD CODE FOR GRAPHICS BLOCK
0342'  4F                 681  GTCH32  LD    C,A               ; RETURN IN REG. C OF BC PAIR
0343'  06 00              682          LD    B,0               ; AND IN REG. A
0345'  F1                 683          POP   AF
0346'  32 003E'           684          LD    (OPBIT),A         ; RESTORE OPBIT
0349'  79                 685          LD    A,C               ; CHAR.CODE TO A
034A'  C9                 686          RET
                          687  ;
034B'  E1                 688  GTCH4   POP   HL                ; PTR. TO CHR. SET
034C'  11 0008            689          LD    DE,8              ; MOVE ON TO NEXT CHARACTER
034F'  19                 690          ADD   HL,DE
0350'  5D                 691          LD    E,L
0351'  54                 692          LD    D,H
0352'  E1                 693          POP   HL                ; DP ADDRESS
0353'  C1                 694          POP   BC                ; CHAR. COUNT
0354'  10 9E              695          DJNZ  GTCH2             ; DECREMENT CHAR.COUNT AND LOOK AGAIN
                          696  ;
0356'  3A 003A'           697          LD    A,(GTINDX)        ; TEST IF DONE
0359'  FE 80              698          CP    80H               ; TEST IF STD.CHR.SET
035B'  20 0A              699          JR    NZ,GTCH5          ; TRY GRAPHICS
035D'  ED 5B 0031'        700          LD    DE,(GRTBL)        ; GRAPHICS TABLE
0361'  06 10              701          LD    B,16              ; NO. OF ENTRIES
0363'  3E 90              702          LD    A,90H
0365'  18 09              703          JR    GTCH1
0367'  FE 90              704  GTCH5   CP    90H               ; TEST IF STD.GRAPHICS
0369'  20 0C              705          JR    NZ,GTCH6          ;
036B'  ED 5B 9C78         706          LD    DE,(UDG)          ; TRY USER-DEFINED GRAPHICS
036F'  1B                 707          DEC   D                 ; ADJUST ADDRESS-100H
0370'  06 15              708          LD    B,21              ; NO. OF ENTRIES
0372'  3E A5              709          LD    A,0A5H            ; INDEX ADJUST
0374'  C3 02F0'           710          JP    GTCH1
                          711
                          712  ;                              ; NO MATCH AT ALL
0377'  F1                 713  GTCH6   POP   AF                ; RESTORE ORIG. OPBIT
0378'  32 003E'           714          LD    (OPBIT),A
037B'  48                 715          LD    C,B               ; BC=0
037C'  AF                 716          XOR   A                 ; A=0
037D'  C9                 717          RET
                          718
                          719                                  ; GET ATTRIBUTE BYTE
                          720  ;
                          721  ;                               ; HERE FROM BASIC ENTRY
037E'  ED 4B 0019'        722  GETAB0: LD    BC,(GETCTL)
                          723  ;                               ; HERE WITH LINE/COL. IN BC
0382'                     724  GETATT:
0382'  CD 0436'           725          CALL  TSTPAR            ; TEST PARAMETERS
0385'  3A 003E'           726          LD    A,(OPBIT)         ; SAVE CURRENT BIT POSITION
0388'  F5                 727          PUSH  AF
0389'  CD 0447'           728          CALL  CONVPM            ; CONVERT TO INTERNAL FORMAT
038C'  CD 0457'           729          CALL  CALCPOS           ; CALCULATE DP POSITION
038F'  CD 0460'           730          CALL  CALCATT           ; CALCULATE ATTRIBUTE POSITION
0392'  7E                 731          LD    A,(HL)            ; ATTRIBUTE BYTE TO A
0393'  4F                 732          LD    C,A               ; PASS BACK IN BC
0394'  06 00              733          LD    B,0
0396'  F1                 734          POP   AF                ; RESTORE (OPBIT)
0397'  32 003E'           735          LD    (OPBIT),A ;
039A'  79                 736          LD    A,C               ; ATTRIBUTE ALSO IN A
039B'  C9                 737          RET
                          738  ;
                          739  ;
                          740  ;                               ; GET CURSOR POSITION
039C'                     741  GETC80: ; HERE FROM BASIC ENTRY
039C'                     742  GETCUR:
039C'  ED 4B 0034'        743          LD    BC,(CURPOS)       ; GET INTERNAL POSITION
03A0'  CD 0447'           744          CALL  CONVPM            ; CONVERT TO USER FORMAT
03A3'  3A 0033'           745          LD    A,(LINLEN)        ; TEST IF END OF LINE
03A6'  B9                 746          CP    C                 ; IF =40 OR 42
03A7'  20 08              747          JR    NZ,GETC1          ; NO
03A9'  3E 00              748          LD    C,0               ; NEXT POSN.START OF NEXT LINE
03AB'  78                 749          LD    A,B
03AC'  3C                 750          INC   A                 ; BUMP TO NEXT LINE
03AD'  47                 751          LD    B,A
03AE'  FE 18              752          CP    24                ; TEST IF OFF SCREEN
03B0'  38 02              753          JR    C,GETC1           ;   NO
03B2'  06 17              754          LD    B,23              ; POSN. IS AT BOTTOM LINE
03B4'  ED 43 0007'        755  GETC1   LD    (LINCOL),BC       ; STORE FOR BASIC PROGRAM
03B8'  C9                 756          RET                     ; VALUES ALSO IN BC
                          757  ;
                          759          SUBTTL INITIALIZE 40-COL.MODE
                          760  ;
                          761  ;
                          762  ;
0389'                     763  INIT40:
0389'  21 1701            764          LD  HL,SCINIT           ; INITIALIZE VARIABLES
038C'  22 0020'           765          LD  (SCRCTL),HL         ; SCROLL CONTROL
038F'  3E 17              766          LD  A,23                ;
03C1'  32 002D'           767          LD  (BOTLN),A           ; BOTTOM LINE
03C4'  3E 01              768          LD  A,1                 ;
03C6'  32 002E'           769          LD  (SCRLCT),A          ; SCROLL COUNT
03C9'  3E 28              770          LD  A,40                ;
03CB'  32 0033'           771          LD  (LINLEN),A          ; LINE LENGTH
03CE'  3E 01              772          LD  A,1                 ;
03D0'  32 0030'           773          LD  (MARGIN),A          ; MARGIN
03D3'  21 1800            774          LD  HL,CLINIT           ;
03D6'  22 000C'           775          LD  (CLRCTL),HL         ; CLEAR SCREEN CONTROL
03D9'  AF                 776          XOR A
03DA'  32 0038'           777          LD   (STRGCT),A         ; STRING COUNT
03DD'  32 003C'           778          LD   (STRGCT+1),A       ;
03E0'  21 0429'           779          LD   HL,CHRSET          ; STD. 40-COL.CHAR.SET
03E3'  22 002F'           780          LD   (CHTBL),HL         ;
03E6'  21 0729'           781          LD   HL,GRPHST          ; STD. 40-COL.GRAPHICS SET
03E9'  22 0031'           782          LD   (GRTBL),HL         ;
                          783  ;                               ; INITIALIZE BASIC INPUTS
                          784  ;
03EC'  32 0000'           785          LD   (DATAB),A          ; DATA BYTE
03EF'  32 0007'           786          LD   (LINCOL),A         ; COLUMN
03F2'  32 0008'           787          LD   (LINCOL+1),A       ; LINE
03F5'  32 0019'           788          LD   (GETCTL),A         ; 'GET' COLUMN
03F8'  32 001A'           789          LD   (GETCTL+1),A       ; 'GET' LINE
                          790  ;
```

```
03F8' 21 0000           791         LD   HL,0              !
03FB' 22 5C70           792         LD   (CCORDS),HL      ! PLOT POSITION
03FE' 3A 0033'          793         LD   A,(LINLEN)       !
0401' 3C                794         INC  A                ! HOME POSITION TO BC
0402' 4F                795         LD   C,A
0403' 06 18             796         LD   B,SCRSZ          ! = LINE 0/COL.0
0405' CC 0452'          797         CALL UPDPOSN          ! CALCULATE AND STORE HOME POSITION
0408' C3 013A'          798         JP   GOODRET          ! RETURN BC=0
                        799 !
                        800 !
                        801 !
                        803         SUBTTL INTERNAL SUB-RTNS.
                        804 !
                        805 !
0408' 3A 5CC3           806 GETSTRG LD   A,(PARAM3)       ! GET STRING ID
040B' E6 1F             807         AND  1FH              ! MASK OFF UPPER BITS
040D' F6 40             808         OR   40H              ! STRING VARIABLE IDENTIFIER
040F' 57                809         LD   D,A              ! SAVE IN D
0410' 2A 5C4B           810         LD   HL,(VARS)        ! FIND STRING
0413' 7E                811 GTSTR1  LD   A,(HL)           !
0414' E6 7F             812         AND  07FH             ! TEST IF END OF VARS AREA
0416' 28 13             813         JR   Z,NOSTRG         ! NO FIND - RETURN BC=2
0418' BA                814         CP   D                ! TEST IF MATCH
0419' 28 08             815         JR   Z,GTSTR2         ! FOUND STRING
041B' 05                816         PUSH DE               ! SAVE STRING ID
041C' CD 1720           817         CALL RECLEN           ! RETURNS ADRS. OF NEXT VAR. IN DE
041F' EB                818         EX   DE,HL            ! ADRS. TO HL
0420' D1                819         POP  DE               ! RESTORE STRING ID
0421' 18 F0             820         JR   GTSTR1           ! LOOK AGAIN
0423' 23                821 GTSTR2  INC  HL               ! GET LENGTH
0424' 4E                822         LD   C,(HL)
0425' 23                823         INC  HL
0426' 46                824         LD   B,(HL)
0427' 23                825         INC  HL               ! FIRST TEXT CODE
0428' 78                826         LD   A,B              ! TEST IF NULL STRING
0429' B1                827         OR   C
042A' C9                828         RET                   ! RETURN (Z IF NULL STRING/NZ IF NOT)
                        829 !
042B' 0E 02             830 NOSTRG  LD    C,2
042D' 06 00             831         LD    B,0
042F' C9                832         RET                   ! (Z AND BC=2 FOR NO FIND)
                        833 !
                        834 !
                        835
                        836 !
                        837 !
                        838                               ! TEST PARAMETER VALIDITY
                        839                               ! IF NOT VALID, DISCARDS RETURN
                        840                               ! AND EXITS VIA INVPAR
0430' 3E 17             841 TSTPAR  LD   A,23              ! TEST LINE >23
0432' B8                842         CP   B
0433' 30 04             843         JR   NC,TSTPR1
0435' F1                844 PARERR  POP  AF
0436' C3 013F'          845         JP   INVPAR
0439' 3A 0033'          846 TSTPR1  LD   A,(LINLEN)
043C' 3D                847         DEC  A
043D' B9                848         CP   C          ! TEST COL. >LINE LENGTH-1
043E' 38 F5             849         JR   C,PARERR
0440' C9                850         RET
                        851 !
                        852                               ! CONVERT LINE/COL. FROM USER TO
                        853                               ! INTERNAL FORMAT AND VICE VERSA
                        854                               ! USER FORMAT: LINES 0-23
                        855                               !              COL. 0-39(+1)
                        856                               ! INTERNAL FORMAT:
                        857                               !              LINES 24-1
                        858                               !              COL. (+3)+1-2
                        859                               !              (COL.1 ENC OF LINE)
                        860 !
                        861 !
0441' 3A 0033'          862 CONVFM  LD    A,(LINLEN)
0444' 3C                863         INC   A
0445' 91                864         SUB   C
0446' 4F                865         LD    C,A             ! CONVERT
0447' 3E 18             866         LD    A,SCRSZ
0449' 90                867         SUB   B
044A' 47                868         LD    B,A
044B' C9                869         RET
                        870 !
                        871 !
                        872 !
044C'                   873 UPDPOSN1                      ! ENTER HERE WITH BC=LINE/COL.IN
                        874                               ! INTERNAL FORMAT
044C' CD 0457'          875         CALL   CALCPOS        ! CALCULATE POSITION
044F' 18 42             876         JR     STPOSN         ! STORE UPDATED POSITION AND RETURN
                        877 !
                        878 !                             ! ROUTINE TO CALCULATE POSITION IN DF
                        879 !                             ! UPDATES BIT POSITION IN VARIABLE DFBIT
                        880 !                             ! RETURNS DF ADDRESS IN HL.  PRESERVES
                        881 !                             ! LINE/COL.POSITION IN BC
                        882 !
0451' CD 047E'          883 CALCPOS CALL   LNBU           ! GET DISPLAY FILE ADRS.FOR LINE
0454' 3A 0033'          884         LD     A,(LINLEN)
0457' 3C                885         INC    A
0458' 91                886         SUB    C              ! GET COL. NO.
0459' F5                887         PUSH   AF             ! SAVE COL. NO.(0-(LINLEN-1))
045A' 5F                888         LD     E,A
045B' 87                889         ADD    A,A
045C' 83                890         ADD    A,E            ! 3 @<40-COL.POS> /4 = 32-COL.POS
045D' CB 3F             891         SRL    A
045F' CB 3F             892         SRL    A
0461' 5F                893         LD     E,A            !
0462' 3A 0030'          894         LD     A,(MARGIN)     ! GET ADJUSTMENT VALUE
0465' 83                895         ADD    A,E
0466' 5F                896         LD     E,A
0467' 16 00             897         LD     D,0            ! DE CONTAINS OFFSET INTO LINE
0469' 19                898         ADD    HL,DE          ! HL CONTAINS DF ADDRESS
046A' 1E 07             899         LD     E,7
046C' F1                900         POP    AF             ! A=COL.POSN.FOR 40 COLS.
046D' E6 03             901         AND    3              !
046F' 28 02             902         JR     Z,CALCP2       ! STARTS AT BIT 7
0471' 5F                903         LD     E,A            ! STARTS AT BIT 1,3 OR 5
0472' 3D                904         DEC    A
0473' 83                905 CALCP2  ADD    A,E
0474' 32 0038'          906         LD     (DFBIT),A
```

```
0478* C9                    987             RET                     ; BC=POS. HL=CP ADDRESS
                            988     ;
                            989     ;
0479*                       990     GNDU:                           ; GET DISPLAY FILE ADDRESS
                            991                                     ; FOR START OF LINE IN B
0479* 3E 10                 992             LD  A,SCRSZ
047B* 90                    993             SUB B
047C* 57                    994             LD  D,A
047D* 0F                    995             RRCA
047E* 0F                    996             RRCA
047F* 0F                    997             RRCA
0480* E6 E0                 998             AND 0E0H
0482* 6F                    999             LD  L,A
0483* 7A                   1000             LD  A,D
0484* E6 1F                1001             AND 1FH
0486* F6 40                1002             OR  40H
0488* 67                   1003             LD  H,A
0489* 3A 5CC2              1004             LD  A,(VIDMOD)          ; TEST IF USING DP1
048C* CB 47                1005             BIT 0,A
048E* C8                   1006             RET Z
048F* 3E 20                1007             LD  A,20H               ; USING DP2
0491* B4                   1008             OR  H
0492* 67                   1009             LD  H,A
0493* C9                   1010             RET
                           1011
0494*                      1012     STPOSN:                         ; STORE CURSOR POSITION
0494* ED 43 0036*          1013             LD  (CURPOS),BC
0498* 22 0036*            1014             LD  (DPADRS),HL
049B* C9                   1015             RET
                           1016     ;
049C*                      1017     LDPOSN:                         ; LOAD CURSOR POSITION
049C* ED 4B 0036*          1018             LD  BC,(CURPOS)
04A0* 2A 0036*            1019             LD  HL,(DPADRS)
04A3* C9                   1020             RET
                           1021     ;
                           1022     ;
04A4*                      1023     UPDATT: ;                       ; UPDATE ATTRIBUTE BYTE FOR CHARACTER
                           1024     ;                               ; JUST PRINTED
                           1025     ;                               ; HL = ANY SCAN OF CHAR.IN DP
                           1026     ;                               ;
                           1027     ;                               ; IF CHARACTER SPANS ATTRIBUTE BYTE
                           1028     ;                               ; BOUNDARY, BOTH ATTRIBUTE BYTES WILL
                           1029     ;                               ; BE ADJUSTED
                           1030     ;
                           1031     ;
04A4* CD 04E0*             1032             CALL CALCATT            ; ADDRS. OF ATTRIBUTE BYTE TO HL
04A7* 3A 003E*             1033             LD  A,(DPBIT)           ; SAVE BIT POSITION
04AA* F5                   1034             PUSH AF                 ;
04AB* 3A 0039*             1035             LD  A,(ATTBYT)          ; CURRENT ATTRIBUTE BYTE VALUE
04AE* 57                   1036             LD  D,A                 ; TO D
04AF* 3A 003F*             1037             LD  A,(ATTMSK)          ; MASK TO D
04B2* 57                   1038             LD  D,A
04B3* 3A 0038*             1039             LD  A,(PASKB)           ; FLAGS
04B6* 47                   1040             LD  B,A                 ; TO B
04B7* 7E                   1041     UPDAT0  LD  A,(HL)              ; BYTE FROM ATTRIBUTE FILE
04B8* A8                   1042             XOR B
04B9* A2                   1043             AND D
04BA* A8                   1044             XOR B                   ; NEW ATTR.FOR 0; OLD FOR 1
04BB* CB 60                1045             BIT 4,B                 ; SET INK COMPLEMENT OF PAPER?
04BD* 28 08                1046             JR  Z,UPDAT1            ;
04BF* E6 F8                1047             AND 0F8H                ; SET INK TO BLACK
04C1* CB 6F                1048             BIT 5,A                 ; IS PAPER 4-7
04C3* 28 02                1049             JR  NZ,UPDAT1           ; YES - USE BLACK INK
04C5* F6 07                1050             OR  7                   ; NO  - USE WHITE INK
04C7* CB 70                1051     UPDAT1  BIT 6,B                 ; SET PAPER COMPLEMENT OF INK?
04C9* 28 08                1052             JR  Z,UPDAT2            ;
04CB* E6 C7                1053             AND 0C7H                ; SET PAPER TO BLACK
04CD* CB 57                1054             BIT 2,A                 ; IS INK 4-7
04CF* 28 02                1055             JR  NZ,UPDAT2           ; YES - USE BLACK PAPER
04D1* F6 38                1056             OR  38H                 ; NO  - USE WHITE PAPER
04D3* 77                   1057     UPDAT2  LD  (HL),A              ; TO ATTRIBUTE FILE
04D4* 3A 003E*             1058             LD  A,(DPBIT)           ; TEST BIT POSITION
04D7* FE 05                1059             CP  5
04D9* 38 08                1060             JR  NC,UPDAT3           ; CHAR.IN SINGLE BYTE IF STARTS
                           1061     ;                               ; IN BIT 7 OR 5
04DB* 23                   1062             INC HL                  ; CHAR.SPANS 2 BYTES IF STARTS
04DC* 3E 07                1063             LD  A,7                 ; IN BIT 3 OR 1
04DE* 32 003E*             1064             LD  (DPBIT),A           ; TEMPORARILY SET TO 7
04E1* 18 D4                1065             JR  UPDAT0              ; UPDATE NEXT BYTE
                           1066                                     ; (DPBIT=7 WILL FORCE EXIT AND
                           1067                                     ;  ORIGINAL VALUE OF DPBIT WILL
                           1068                                     ;  BE RESTORED)
                           1069     ;
04E3* F1                   1070     UPDAT3  POP AF                  ; ORIGINAL DPBIT
04E4* 32 003E*             1071             LD  (DPBIT),A           ; RESTORE
04E7* C9                   1072             RET
                           1073     ;
                           1074
                           1075     ;                               ; SUB-RTN. TO CALCULATE ADRS. OF
                           1076     ;                               ; ATTRIBUTE BYTE FROM ADRS. OF
                           1077     ;                               ; ANY SCAN ROW IN DP
                           1078     ;
                           1079     ;                               ; USED BY UPDATT AND GETATT
                           1080     ;
04E8* 7C                   1081     CALCATT LD  A,H                 ; UPPER BYTE OF ADDRESS
04E9* 0F                   1082             RRCA
04EA* 0F                   1083             RRCA
04EB* 0F                   1084             RRCA
04EC* E6 03                1085             AND 03H
04EE* F6 58                1086             OR  58H                 ; ADRS. OF ATTRIBUTE BYTE
04F0* CB 6C                1087             BIT 5,H                 ; TEST WHICH DP
04F2* 28 02                1088             JR  Z,CALCA1            ; DP1
04F4* F6 20                1089             OR  20H                 ; DP2
04F6* 67                   1090     CALCA1  LD  H,A                 ; ADRS. OF ATTRIBUTE
04F7* C9                   1091             RET
                           1092     ;
04F8*                      1093     LDATTR: ;                       ; LOAD INTERNAL ATTRIBUTE VARIABLES
                           1094     ;                               ; FROM SYSTEM VARIABLES
                           1095     ;
04F8* F5                   1096             PUSH AF                 ; SAVE A
04F9* 3A 5C8D              1097             LD  A,(ATTR_P)          ;
04FC* 32 0039*             1098             LD  (ATTBYT),A          ;
04FF* 3A 5C8E              1099             LD  A,(MASK_P)          ;
0502* 32 003F*             1100             LD  (ATTMSK),A          ;
0505* 3A 5C91              1101             LD  A,(P_FLAG)
0508* 0F                   1102             RRCA                    ; SHIFT OOO BITS TO OVEN
```

```
050E'  32 0038'          1023            LD   (MASKB),A          ;
0511'  F1                1024            POP  AF
0512'  C9                1025            RET
                         1026    ;
                         1027    ;                               ; RTN. USED BY GTCHAR TO PUT
                         1028    ;                               ; 6 PIXELS FROM DP IN B REG.BITS 7-2
                         1029    ;                               ; FOR MATCHING AGAINST CHAR.SET
                         1030    ;
0513'  E5                1031    GTC28C  PUSH HL                 ; SAVE DP ADRS.
0514'  46                1032            LD   B,(HL)             ; SCAN LINE FROM DP
0515'  23                1033            INC  HL                 ; NEXT BYTE
0516'  4E                1034    GTC281  LD   C,(HL)             ; BC CONTAINS CHAR.AT DPBIT
0517'  3A 003E'          1035            LD   A,(DPBIT)
051A'  D6 07             1036            SUB  7
051C'  ED 44             1037            NEG                     ; A=NO.OF BITS OFFSET IN BC
051E'  28 07             1038            JR   Z,GTC282           ; STARTS AT BIT 7
0520'  CB 11             1039    GTCSHFT RL   C
0522'  CB 10             1040            RL   B                  ; SHIFT 6 PIXELS FOR CHARACTER
0524'  3D                1041            DEC  A                  ; TO B, BITS 7-2
0525'  20 F9             1042            JR   NZ,GTCSHFT
0527'  E1               1043    GTC282  POP  HL                 ;
0528'  C9               1044            RET                     ; CHAR.IN B, BITS 7-2
                         1045    ;
                         1046
                         1047            INCLUDE CHST80
                         1048            SUBTTL CHAR.SET FOR 40/80 COL.MODES
                         1049
                         1050    ;
                         1051    ; Dot patterns for characters on the TV screen:
                         1052    ;       character with code x is at offsets 8x (top line) to 8x+7 (bottom line)
                         1053    ;       0 = white, 1 = black, no margins between character spaces either
                         1054    ;               vertically or horizontally
                         1055
0529'  00                1056    CHRST   defb 00000000b          ; code 20       space
052A'  00                1057            defb 00000000b
052B'  00                1058            defb 00000000b
052C'  00                1059            defb 00000000b
052D'  00                1060            defb 00000000b
052E'  00                1061            defb 00000000b
052F'  00                1062            defb 00000000b
```

See APPENDIX C-2 for Character Set

```
ADL - ASC 003 40-COL.MODE SUPPORT       CRI80/11 version 10.36.14       16-Mar-84  12:43:23
CHAR.SET FOR 40/80 COL.MODES    CHST80.SRC

A       Reserved        ATTBYT  0039            ATTMSK  003F            ATTR8P  5C8D            B       Reserved
BOTLN   0020            C       Reserved        CALCAT  04E0            CALCA1  04FB            CALCP0  0457
CALCP2  0479            CHRSET  0429            CHRST   0529            CHTBL   002F            CLINIT  1800
CLRCTL  000C            CLRSB0  024F            CLRSC0  900E IN          CLRSCN  0253 IN         CLRSC0  0269
CLRSC1  027D            CLRSC2  0288            CLRSC3  928C            CLRSC4  028F            CLRSC5  029A
CLRSC7  02CF            CLRSXT  02C8            CONVPM  0447            COORDS  5C7D            CURPOS  0034
D       Reserved        DATAB   0000            DPADRS  0036            DPBIT   003E            E       Reserved
ERREET  013C            GETAB0  037E            GETATB  C01E IN          GETATT  0382 IN         GETC80  039C
GETCTL  0019            GETCUB  0021 IN         GETCUR  039C IN         GETC1   0384            GETSTR  0408
GOODRE  013A            GRBLK   0224            GRPMST  0729            GRPST   0829            GRTBL   0031
GTCHAR  020B IN         GTCHB0  0207            GTCHRB  0918 IN         GTCH1   02F0            GTCH2   02F4
GTCH22  02FA            GTCH23  0310            GTCH3   0315            GTCH31  0318            GTCH32  0342
GTCH4   0348            GTCH5   0367            GTCH6   0377            GTCSHF  0920            GTC28C  0513
GTC281  0516            GTC282  0527            GTINDX  003A            GTSTR1  0419            GTSTR2  0429
H       Reserved        INIT48  0025 IN         INIT40  03B9 IN         INSOBL  010A            INVPAR  013F
L       Reserved        LDATTR  04F0            LOPOSN  04A1            LINCOL  0007            LINLEN  0033
LNBU    047E            LOOP    010E            LOOP0   01E1            LOOP1   01E9            M       Reserved
MARGIN  0030            MASKB   0038            MASK8P  5C8E            NEXT1   008E            NEXT2   00E6
NEXT3   0136            NOINVE  0113            NORR1   02C5            NORR2   00E0            NORR3   0100
NOSTRG  0431            NOXOR   00CB            PARAM8  5CC3            PARERR  0438            PSW     Reserved
P8PLAG  5C91            RECLEN  1720            SCINIT  1701            SCRCTL  0028            SCRLB   002A IN
SCRLB0  01A2            SCRLCT  002E            SCRLXT  0201            SCRLO   019C            SCRCLL  01A6 IN
SCRSZ   0018            SETCB0  0192            SETCUB  0009 IN         SETCUR  0196 IN         SP      Reserved
SPARE0  0011            SPARE1  0013            SPARE2  0015            SPARE3  0017            SPARE4  0024
STBLK   022C            STBLK0  0230            STBLK1  0231            STPOSN  0499            STRGCT  003B
TESTAD  01CC            TSTPAR  0436            TSTPR1  043F            TVPULC  0143            TVPUL1  015F
UDG     5C7B            UPDATT  04A9            UPDAT0  04BC            UPDAT1  04CC            UPDAT2  04D8
UPDAT3  04E8            UPOPOS  0452            VARS    5C4B            VIDMOD  5CC2            WRCHAR  0043 IN
WRCHRB  0001 IN         WRCHXT  0137            WRCH0   0340            WRCH01  0046            WRCH11  0065
WRCH12  006D            WRCH13  0071            WRCH14  0087            WRCH15  0088            WRCH2   008E
WRCH3   0099            WRCH5   009F            WRCH6   03AB            WRCH7   0120            WRSTLP  0170
WRSTRB  0004 IN         WRSTRG  0160 IN         WRSTR0  0169            WRSTR3  017A            WRSTX1  0183

    No errors detected
```

```
Cross reference listing (XREF version 4.7)

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

ATTBYT    1116    587     955     10188
ATTMSK    1190    957    10206
ATTR_P     480   1017
BOTLN      950    296     7678
CALCA1    1008   1010#
CALCAT     459    576     730     952    1001#
CALCP2     902    905#
CALCP0     622    729     875     883#
CHRSET     400    103     779
CHRST       40   1056#
CHTBL     1030    152     623     780#
CLINIT     390    774
CLRCTL     730    511     775#
CLRSB0      76    510#
CLRSC0     471    523     527#
CLRSC1    5400    597
CLRSC2    5490
CLRSC3    5520    606
CLRSC4    5540    573
CLRSC5    5630
CLRSC7     948    600#
```

```
CLRSCB      31      760
CLRSCN      28     5130
CLRSET     595     5980
CONVPN     368      621     728     744     8628
COORDS     470     7928
CURPOS    1060      743    9338     938
DATAB      660      125    7856
DPADRS    1080     9340     939
DPBIT     1180      201     217     246     278    2808    2838
                    619    6848    7148     726    7358    9068
                    953     978    9848    9918    1035
ERRRET    2870      291     308
GETAB0      84      722#
GETATB      32      840
GETATT      29     7240
GETC1      747      753    7598
GETCB0      85     7410
GETCTL      818      615     722    7688    7898
GETCUB      32      858
GETCUR      29     7428
GETSTR      319     8068
GOODRE     2860      378     392     524     798
GRBLK       415     4738
GRPHST      410      104     781
GRPST        61    19208
GRTBL      1048      149     700    7028
GTC2B1    10348
GTC2B2     1038    10438
GTC2BC      631      656    10318
GTCH1      6268      703     718
GTCH2      6288      695
GTCH22     6338
GTCH23      644     6478
GTCH3       648     6508
GTCH31     6528      667
GTCH32      677      679    6818
GTCH4       646      649     666    6888
GTCH5       699     7048
GTCH6       705     7138
GTCHAR       29     6178
GTCHB0       83     6128
GTCHRB       32      838
GTCSHP    10398     1042
GTINDX     1138     6268     642     674     697
GTSTR1     9118      920
GTSTR2      915     9218
INIT40       34       87    7638
INIT48       34      878
INSOBL     4168
INVPAR      137      139     2988     389     388     398     517
                    528      522     845
LOATTR      128      327     527    10138
LOPOSN      132     9378
LINCOL      698      364    7558    7848    7878
LINLEN     1058      164     398     532     745    7718     793
                    846      862     884
LNBU        401      535     883    9108
LOOP       4198      479
LOOP0      4218      438
LOOP1      4288
MARGIN     1178     7738     894
MASAB      1098      178     959    10238
MASK_P      498     1019
NEXT1      2078      210
NEXT2      2338      236
NEXT3      2528      255
NOINVE      245     2628
NORM1       205     2118
NORM2       231     2378
NORM3       250     2568
NOSTRG      813      838#
NOXOR       200     2178
PARAMB       528      806
PARERR      8448      849
P_FLAG       508     1021
RECLEN       438      817
SCINIT       388      764
SCRCTL       888      510     378    7658
SCRLO        312      391    3948
SCRLB         31      918
SCRLB0        91     3768
SCRLCT      1008      308    3048    3098    7698
SCRLXT       442     4458     508
SCROLL        28     3808
SCRSZ         378      293     398     529     798     868     918
SETCB0        72     3628
SETCUB        31      728
SETCUR        28     3668
SPARE0        778
SPARE1        788
SPARE2        798
SPARE3        808
SPARE4        868
STBLK       407     4818
STBLK0     4848      498
STBLK1     4858
STPOSN      285      878      328
STRGCT      1158     3498    7778    7788
TESTAD      4058      444     503
```

TSTPAR	367	618	725	841#	
TSTPR1	843	846#			
TVPUL1	302	309#			
TVPULQ	173	293#			
UDG	46#	145	706		
UPDAT0	961#	985			
UPDAT1	966	969	971#		
UPDAT2	972	975	977#		
UPDAT3	980	990#			
UPDATT	274	963#			
UPDPOS	298	369	599	797	873#
VARS	45#	817			
VIDMOD	51#	924			
WRCH0	67	123#			
WRCH01	132#	331			
WRCH11	144	149#			
WRCH12	142	152#			
WRCH13	148	151	153#		
WRCH16	165	170#			
WRCH15	169	171#			
WRCH2	176#				
WRCH3	181	183#			
WRCH5	188#	272			
WRCH6	196#				
WRCH7	273#				
WRCHAR	28	127#			
WRCHAB	31	67#			
WRCHXT	281	285#			
WRSTLP	328#	342			
WRSTR0	68	319#			
WRSTR3	336#	347			
WRSTR8	31	68#			
WRSTRG	28	327#			
WRSTX1	334	345#			

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 RAM 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 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).

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.

- -

Application 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 OS RAM routines at the top of memory. The second display file is cleared to zeroes. The effected 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 (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) 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 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: 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.

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)

From 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)

From 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 0 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 (WRSTRE).
ATTMSK	1	Working Byte - (Copy of MASK P)

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

Variable Name	Value
SCRCTL	1701H
BOTLN	17H
SCRLCT	1H
CHTBL	3000H
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 these initialized by SETMOD when the second display file is first opened:

Variable Name	Size	Value
* DATAB	1	0H
* LINCCL	2	0H
* CLRCTL	2	1800H
* 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.

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 (15M) and SCRCTL+1 be set to 21 (15M) 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.

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 video 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  CRI80/11 version 10.36.14       16-Mar-84  12:47:30
VERSION LEVEL CONTROL   ADDS.SRC

                                1           NAME ADL - ASC 004 DUAL SCREEN MODE SUPPORT
                                2   ;
                                3   ;
                                4   ;
                                5   ;
                                6   ;
                                7   ;
                                8   ;
                                9   ;
                               10   ;               ************************************
                               11   ;               *                                  *
                               12   ;               *              TIMEX               *
                               13   ;               * APPLICATION DEVELOPMENT LIBRARY *
                               14   ;               *                                  *
                               15   ;               * NAME: DUAL SCREEN MODE SUPPORT  *
                               16   ;               *                                  *
                               17   ;               * ASC NO: 004                     *
                               18   ;               * VERSION: 001                    *
                               19   ;               * AUTHOR:  C. CORCORAN            *
                               20   ;               * DATE:    1/06/84                *
                               21   ;               ************************************
                               23           SUBTTL VERSION LEVEL CONTROL
                               24   ;
                               25   ;
                               26   ;       VERSION         DATE            COMMENTS
                               27   ;       -------         ----            --------
                               28   ;
                               29   ;         001           1/6/84          ORIGINAL
                               30   ;
                               32           SUBTTL DEFINITIONS
                               33   ;
                               34   ;
                               35   ;
                               36   ;                       ****DEFINITIONS****
                               37   ;
                               38           INTERN  WRCHAR,WRSTRG,SETCUR,CLRSCN,SCROLL
                               39           INTERN  GTCHAR,GETATT,GETCUR
                               40           INTERN  WRCHRB,WRSTRB,SETCUB,CLRSC9,SCRLB
                               41           INTERN  GTCHRB,GETATB,GETCUB
                               42           INTERN  SETMODE,SETMOB
                               43           INTERN  COPYSC,COPYSB
                               44           INTERN  EXCHSC,EXCHSB   .
                               45   ;
=0018                          46   SCRSZ           EQU     24      ; 24 LINES
=1701                          47   SCINIT          EQU     1701H   ; SCROLL CONTROL INITIALIZATION
=1800                          48   CLINIT          EQU     1800H   ; CLEAR SCREEN CTL. INIT.
=3C00                          49   CHRSET          EQU     3C00H   ; ROM CHAR.TABLE-100H
=05E8'                         50   GRPHST          EQU     ((GRPST)-100H) ;STD.GRAPHICS CHARS.
                               51   ;
=0E8E                          52   CHNGVID         EQU     0E8EH   ; ROUTINE IN ROM EXTENSION
=1720                          53   RECLEN          EQU     1720H   ; ROUTINE IN HOME ROM
=30F9                          54   ININT           EQU     30F9H   ; ROUTINE IN HOME ROM
=3193                          55   PP2A            EQU     3193H   ; ROUTINE IN HOME ROM
                               56   ;
=00FF                          57   HREXPT          EQU     0FFH    ; HOME ROM EXT.SELECT PORT (BIT 7)
=00F4                          58   DKHSPT          EQU     0F4H    ; DOCK HORIZONTAL SELECT PORT
                               59   ;
=5C48                          60   VARS            EQU     5C48H   ; SYSTEM VARIABLES
=5C50                          61   CH_ADD          EQU     5C50H   ;
=5C65                          62   STKEND          EQU     5C65H   ;
=5C7B                          63   UDG             EQU     5C7BH   ;
=5C7D                          64   COORDS          EQU     5C7DH   ;
=5C8D                          65   ATTR_P          EQU     5C8DH   ;
=5C8E                          66   MASK_P          EQU     5C8EH   ;
=5C91                          67   P_FLAG          EQU     5C91H   ;
=5CB2                          68   RAMTOP          EQU     5CB2H   ;
=5CB4                          69   PRAMT           EQU     5CB4H   ;
=5CC2                          70   VIDMOD          EQU     5CC2H   ;
=5CC3                          71   PARAMS          EQU     5CC3H   ; (LOCATION 23747)
=6840                          72   DRIVES          EQU     6840H   ;
=12C0                          73   INSERTSZ        EQU     7800H-DRIVES
=0840                          74   MOVESZ          EQU     DRIVES-6000H
=F7C0                          75   DEST7           EQU     0FFFFH-MOVESZ+1
=97C0                          76   PIX             EQU     DEST7-6000H
                               77
=1000                          78   PIXTBL          EQU     1000H
                               79
                               80
```

```
                                82          SUBTTL INPUTS AND ENTRIES
                                83  ;
                                84  ;
                                85  ;
                                86  ;
                                87  ;
                                88  ;
                                89  ;               **PARAMETER INPUTS AND ENTRY POINTS FROM BASIC**
                                90  ;                                                                  WITH
                                91  ;                                                               MODULE LOADED
                                92  ;                                                               AT 57344(E000H)
                                93  ;                                                               DECIMAL ADDRESS
                                94  ;                                                               ---------------
                                95  ;
0000' 20                        96  DATAB           DEFB    20H     ; DATA BYTE FROM BASIC          57344
0001' C3 0046'                  97  WRCHRB:         JP      WRCH0   ; ENTRY TO WRITE CHARACTER      57345
0004' C3 00EE'                  98  WRSTRB:         JP      WRSTR0  ; ENTRY TO WRITE STRING         57348
                                99  ;                               ; (see PARAMS above)
0007' 0000                     100  LINCOL          DEFW    0       ; SET CURSOR PARAMETERS
                               101                                 ;   LINCOL=COL. (0-31)          57351
                               102                                 ;   LINCOL+1=LINE (0-23)        57352
0009' C3 0117'                 103  SETCUB:         JP      SETC80  ; ENTRY TO SET CURSOR POSN.     57353
000C' 1800                     104  CLRCTL          DEFW    1800H   ; CLEAR SCREEN CONTROLS
                               105                                 ;   CLRCTL=STARTING LINE NO.    57356
                               106                                 ;   CLRCTL+1=LINE COUNT         57357
000E' C3 0104'                 107  CLRSCB:         JP      CLRS80  ; ENTRY TO CLEAR SCREEN         57358
0011' 0000                     108  SPARE0          DEFW    0       ; NOT USED (ALIGNMENT ONLY)
0013' 0000                     109  SPARE1          DEFW    0       ;
0015' 0000                     110  SPARE2          DEFW    0       ;
0017' 0000                     111  SPARE3          DEFW    0       ;
0019' 0000                     112  GETCTL          DEFW    0       ; "GET" CONTROL FROM BASIC      57369
                               113                                 ; FORMAT IS AS FOR LINCOL
001B' C3 025C'                 114  GTCHRB:         JP      GTCH80  ; ENTRY TO GET CHARACTER        57371
001E' C3 020C'                 115  GETATB:         JP      GETAB0  ; ENTRY TO GET ATTRIBUTE        57374
0021' C3 02F1'                 116  GETCUB:         JP      GETC80  ; ENTRY TO GET CURSOR POSN.     57377
0024' 00                       117  VIMCDE          DEFB    0       ; VIDEO MODE CONTROL            57380
0025' C3 0305'                 118  SETMDB:         JP      STMD80  ; ENTRY TO SET VIDEO MODE       57381
0028' 1701                     119  SCRCTL          DEFW    1701H   ; SCROLL CONTROL
                               120                                 ;   SCRCTL=STARTING LINE NO.    57384
                               121                                 ;   SCRCTL+1=LINE COUNT         57385
002A' C3 0127'                 122  SCRLB:          JP      SCRL80  ; ENTRY TO SCROLL               57386
                               123  ;
                               124  ;COPYSB                        ; ENTRY TO COPY FROM DP         57408
                               125  ;                              ; (SEE NEXT PAGE)
                               126  ;EXCHSB                        ; ENTRY TO EXCHANGE SCREEN      57411
                               127  ;                              ; (SEE NEXT PAGE)
                               128  ;
                               129          SUBTTL  OTHER VARIABLES
                               130  ;
                               131  ;
                               132  ;
                               133  ;
002D' 17                       134  BOTLN           DEFB    17H     ; BOTTOM LINE (LINE AFTER       57389
                               135                                 ; WHICH SCREEN IS CONSIDERED
                               136                                 ; FULL.  AUTOMATIC SCROLL
                               137                                 ; WILL BE DONE IF SCRLCT
                               138                                 ; DOES NOT DECREMENT TO 0.
002E' 01                       139  SCRLCT          DEFB    1       ; SCROLL COUNT - 1 PLUS NO.     57390
                               140                                 ; OF TIMES AUTOMATIC SCROLL
                               141                                 ; WILL BE DONE.
                               142  ;
                               143  ;
                               144  ;
                               145  ;
002F' 3C00                     146  CHTBL           DEFW    CHRSET  ; ADDRESS OF CHARACTER TABLE    57391
                               147                                 ; (DEFAULTS TO TS2068 SYSTEM
                               148                                 ;  ROM)
0031' 0580'                    149  GRTBL           DEFW    GRPHST  ; ADDRESS OF STD.GRAPHICS TBL.  57393
0033' 20                       150  LINLEN          DEFB    20H     ; LINE LENGTH (32)              57395
0034' 1821                     151  CURPOS          DEFW    1821H   ; CURRENT POSITION (INTERNAL     57396
                               152                                 ;  FORMAT)
0036' 4000                     153  DFADRS          DEFW    4000H   ; CURRENT DISPLAY FILE ADDRS.   57398
0038' 00                       154  MASKB           DEFB    0       ; MASK TO BE APPLIED TO         57400
                               155                                 ; DISPLAY CHARACTERS
0039' 00                       156  ATTRVT          DEFB    0H      ; CURRENT ATTRIBUTES            57401
                               157  ;
003A' 80                       158  GTINDX          DEFB    80H     ; INDEX FOR ADJUSTING           57402
                               159                                 ; CH.CODE IN GTCHAR
003B' 0000                     160  STRGCT          DEFW    0       ; REMAINING COUNT FROM WRSTRG
                               161  ;                               ; WHEN "SCREEN FULL"            57403
003D' 0000                     162  SPARE4          DEFW    0       ;                               57405
003F' 00                       163  ATTMSK          DEFB    0       ; ATTRIBUTE MASK                57407
                               164  ;
0040' C3 030A'                 165  COPYSB          JP      COPYS0  ; BASIC ENTRY TO COPY DP        57408
0043' C3 04F5'                 166  EXCHSB          JP      EXCHS0  ; BASIC ENTRY TO EXCHANGE DP    57411
                               167          SUBTTL  WRITE CHARACTER
                               168  ;
                               169  ;
0046'                          170  WRCH0:                          ; HERE FROM BASIC ENTRY TO DISPLAY THE
                               171                                  ; CHARACTER WHOSE CODE IS IN "DATAB"
0046' 3A 0000'                 172          LD      A,(DATAB)
                               173  ;
0049'                          174  WRCHAR:                         ; ENTRY WITH CODE IN A
0049' CD 06A2'                 175          CALL    LDATTR          ; GET ATTRIBUTE AND MASK CONTROLS
                               176  ;
                               177  ;                               ; ENTRY TO USE ATTRIBUTE VALUES IN
                               178  ;                               ; INTERNAL VARIABLES WITHOUT RELOADING
004C' CD 0650'                 179  WRCH01: CALL    LDPOSN          ; LOAD REGISTERS FOR CURRENT POSITION
                               180                                  ;   BC=CURSOR POSITION FROM "CURPOS"
                               181                                  ;   HL=DISPLAY FILE ADDRESS FROM "DFADRS
                               182                                  ;   A=CODE FOR CHAR. TO BE DISPLAYED
004F' FE 20                    183          CP      20H             ; TEST VALID RANGE
0051' DA 00C4'                 184          JP      C,INVPAR        ; < 20H
0054' FE A0                    185          CP      0A0H            ;
0056' D2 00C4'                 186          JP      NC,INVPAR       ; > A0H
0059' C5                       187          PUSH    BC              ; SAVE CURSOR POSITION
005A' FE 80                    188          CP      80H             ; TEST IF ASCII PRINTABLE (20H-7FH)
005C' 38 15                    189          JR      C,WRCH12        ; YES
005E' FE 90                    190          CP      90H             ; TEST IF STD.GRAPHICS(80H-8FH)
0060' 38 09                    191          JR      C,WRCH11
0062' 0E 48 9C78               192          LD      BC,(UDG)        ; USER-DEFINED GRAPHICS(90H-A4H)
```

```
0066'  05                      195             DEC   B               ; ADJUST ADDRESS-100H
0067'  D6 70                   196             SUB   70H             ; ADJUST FOR INDEX INTO TABLE
0069'  18 0C                   197             JR    WRCH13
0069'  ED 4B 0031'             198   WRCH11    LD    BC,(CGRTBL)     ; ADDRESS OF GRAPHICS TABLE
006F'  D6 40                   199             SUB   40H             ; ADJUST FOR INDEX INTO TABLE
0071'  18 04                   200             JR    WRCH13
0073'  ED 4B 002F'             201   WRCH12    LD    BC,(CHRTBL)     ; CHARACTER TABLE
0077'  EB                      202   WRCH13    EX    DE,HL           ; DE=POSN.IN DISPLAY FILE
0078'  26 00                   203             LD    H,0
0078'  6F                      204             LD    L,A             ; CODE IS INDEX INTO TABLE
0078'  29                      205             ADD   HL,HL
007C'  29                      206             ADD   HL,HL
007D'  29                      207             ADD   HL,HL
007E'  09                      208             ADD   HL,BC           ; HL PIXEL PATTERN IN TABLE
007F'  C1                      209             POP   BC
0080'  EB                      210             EX    DE,HL
0081'  79                      211             LD    A,C             ; TEST IF END OF LINE
0082'  3D                      212             DEC   A
0083'  3A 0033'                213             LD    A,(LINLEN)
0086'  20 05                   214             JR    NZ,WRCH14
0088'  3C                      215             INC   A               ; START OF NEW LINE
0089'  05                      216             DEC   B               ; BUMP LINE NO.
008A'  4F                      217             LD    C,A             ; LINLEN+1=START OF LINE
008B'  18 01                   218             JR    WRCH15
008D'  3C                      219   WRCH14    INC   A
008E'  B9                      220   WRCH15    CP    C               ; TEST IF START OF LINE
008F'  D5                      221             PUSH  DE
0090'  CC 00C8'                222             CALL  Z,TVFULG        ; TEST IF OFF SCREEN
0093'  D1                      223             POP   DE
                               224                                   ; HERE TO PUT CHARACTER IN DISPLAY FILE
0094'  C5                      225   WRCH2     PUSH  BC              ; CURSOR POSITION
0095'  E5                      226             PUSH  HL              ; DISPLAY FILE ADRS.
0096'  3A 0038'                227             LD    A,(MASK8)
0099'  06 FF                   228             LD    B,-1
009B'  1F                      229             RRA
009C'  38 01                   230             JR    C,WRCH3         ; IF XORING NEW INTO OLD
009E'  04                      231             INC   B               ; B=-1 IF XORING  =0 IF NOT
009F'  1F                      232   WRCH3     RRA
00A0'  1F                      233             RRA
00A1'  9F                      234             SBC   A,A
00A2'  4F                      235             LD    C,A             ; C=-1 FOR INVERSE  =0 IF NOT
00A3'  3E 08                   236             LD    A,8             ; SCAN COUNT
00A5'  A7                      237             AND   A
00A6'  EB                      238             EX    DE,HL
00A7'  08                      239   WRCH5     EX    AF,AF'
00A8'  1A                      240             LD    A,(DE)          ; OLD BIT PGM
00A9'  A0                      241             AND   B
00AA'  AE                      242             XOR   (HL)
00AB'  A9                      243             XOR   C               ; INVERT IF C=-1
00AC'  12                      244             LD    (DE),A          ; UPDATE DISPLAY FILE
00AD'  08                      245             EX    AF,AF'
00AE'  14                      246             INC   D               ; NEXT SCAN
00AF'  23                      247             INC   HL              ; NEXT BYTE IN CHAR.FILE
00B0'  3C                      248             DEC   A               ; TEST IF MORE SCANS
00B1'  20 F4                   249             JR    NZ,WRCH5
00B3'  15                      250   WRCH7     DEC   D               ; ADJUST DP ADRS.
00B4'  EB                      251             EX    DE,HL           ; DP ADRS. TO HL
00B5'  CD 0665'                252             CALL  UPDATT          ; UPDATE ATTRIBUTE BYTE
00B8'  E1                      253             POP   HL              ; STARTING ADDRESS
00B9'  C1                      254             POP   BC              ; CURSOR POSITION
00BA'  0C                      255             DEC   C               ; ADJUST CURSOR POSITION
00BB'  23                      256             INC   HL              ; ADJUST DP ADDRESS
00BC'  CD 0655'                257   WRCHRT    CALL  SYPOSN          ; STORE NEW POSITION
00BF'  0E 00                   258   GCJDRET   LD    C,0             ; RETURN BC=0
00C1'  06 00                   259   ERRRET    LD    B,0             ; ENTER HERE WITH C NON-ZERO
00C3'  C9                      260             RET
                               261   ;
00C4'  0E 01                   262   INVPAR    LD    C,1             ; RETURN BC=1 FOR INVALID PARAMETERS
00C6'  18 F9                   263             JR    ERRRET
                               264   ;
00C8'  3E 18                   265   TVFULG    LD    A,SCRSZ         ; TEST IF NEW LINE IS OFF SCREEN
00CA'  90                      266             SUB   B               ; A=LINE NO.
00CB'  57                      267             LD    D,A             ; SAVE IN D
00CC'  3A 0020'                268             LD    A,(BOTLN) ; GET BOTTOM LINE
00CF'  BA                      269             CP    D
00D0'  D2 0628'                270             JP    NC,UPDPOSN      ; ON SCREEN - RETURN TO WRITE CHAR.
                               271                                   ;  VIA UPDATE AND STORE POSITION
00D3'  3A 0026'                272             LD    A,(SCRLCT)      ; TEST SCROLL COUNT
00D6'  3D                      273             DEC   A
00D7'  20 08                   274             JR    NZ,TVFUL1       ; DO AUTOMATIC SCROLL USING SCRCTL
00D9'  3E 01                   275             LD    A,1             ; REINITIALIZE TO 1
00DB'  32 0026'                276             LD    (SCRLCT),A
00DE'  C1                      277             POP   BC
00DF'  C1                      278             POP   BC              ; DISCARD RETURN TO WRITE CHAR.
00E0'  0E 03                   279             LD    C,3             ; RETURN BC=3 FOR SCREEN FULL
00E2'  18 DD                   280             JR    ERRRET
00E4'  32 0026'                281   TVFUL1    LD    (SCRLCT),A      ; UPDATE VARIABLE
00E7'  ED 4B 0026'             282             LD    BC,(SCRCTL)     ; GET SCROLL CONTROLS (STARTING LINE
                               283                                   ;   AND LINE COUNT)
00EB'  C3 0141'                284             JP    SCRL0           ; RETURN TO WRITE CHAR. VIA SCROLL
                               285   ;
                               287             SUBTTL WRITE STRING
                               288   ;                               ; HERE FROM BASIC ENTRY TO
                               289   ;                               ; WRITE CHARACTER STRING TO SCREEN.
                               290   ;                               ; STRING IDENTIFIER (CH.CODE) IN
                               291   ;                               ; SYSTEM VARIABLE PARAM4 AT 5CC3H
00EE'  CD 0688'                292   WRSTR0    CALL  GETSTRG         ; GET STRING VAR.ADRS. TO HL
00F1'  C8                      293             RET   Z               ; STRING NOT FOUND OR NULL STRING
                               294   ;                               ; (BC CONTAINS STATUS)
                               295   ;
                               296   ;                               ; ENTRY FROM MACHINE CODE WITH
                               297   ;                               ; ADDRESS OF CHAR.CODE 'STRING'
                               298   ;                               ; IN HL AND LENGTH IN BC
                               299   ;
00F2'  CD 06A2'                300   WRSTRG1   CALL  LDATTR          ; GET ATTRIBUTE AND MASK CONTROLS
00F5'  7E                      301   WRSTLP    LD    A,(HL)          ; GET CODE
00F6'  E5                      302             PUSH  HL              ; SAVE ADDRESS
00F7'  C5                      303             PUSH  BC              ; AND COUNT
00F8'  CD 004C'                304             CALL  WRCH01          ; WRITE 'A'
00FB'  79                      305             LD    A,C             ; TEST IF GOOD
00FC'  B0                      306             OR    B
00FD'  20 09                   307             JR    NZ,WRSTX1       ; EXIT IF INVALID CODE OR
                               308   ;                               ; IF SCREEN FULL
00FF'  C1                      309   WRSTR3    POP   BC              ; COUNT
```

```
0100'  E1                    310          POP   HL              ; ADDRESS
0101'  23                    311          INC   HL              ; NEXT CHAR.
0102'  08                    312          DEC   BC              ; ADJUST COUNT
0103'  78                    313          LD    A,B             ; TEST IF DONE
0104'  B1                    314          OR    C
0105'  20 EB                 315          JR    NZ,WRSTLP       ; WRITE NEXT CHAR.
0107'  C9                    316          RET                   ; RETURN BC=0
                             317  ;
0108'  79                    318  WRSTX1  LD    A,C             ; SAVE ERROR
0109'  FE 01                 319          CP    1               ; TEST IF UNRECOGNIZED CODE
010B'  28 F2                 320          JR    Z,WRSTR3        ; CONTINUE
010D'  E1                    321          POP   HL              ; REMAINING COUNT TO HL
010E'  22 0039'              322          LD    (ST4GCT),HL     ; STORE REMAINING COUNT
0111'  E1                    323          POP   HL              ; ADDRESS TO HL
0112'  0E 03                 324          LD    C,3             ; RETURN BC=3
0114'  06 00                 325  WRSTX2  LD    B,0
0116'  C9                    326          RET
                             327  ;
POSITION CONTROL   IODS.SRC

                             329          SUBTTL POSITION CONTROL
                             330  ;
0117'                        331  SETC00:                       ; HERE FROM BASIC ENTRY.  PARAMETERS
                             332                                ; IN LINCCL
0117'  ED 4B 0007'           333          LD    BC,(LINCCL)
                             334  ;
0118'                        335  SETCUR:                       ; ENTRY WITH LINE/COL. IN BC REG.
0118'  CC 040C'              336          CALL   TSTPAR         ; TEST PARAMETERS FOR VALIDITY
011E'  CC 0410'              337          CALL   CONVFP         ; CONVERT TO INTERNAL FORMAT
0121'  CD 0429'              338          CALL   UPDPOSN        ; CALCULATE AND STORE POSITION
0124'  C3 008F'              339          JP     GOODRET        ; RETURN BC=0
                             340  ;
                             342          SUBTTL SCROLL
                             343  ;
                             344  ;
                             345  ;
0127'                        346  SCRL00:                       ; HERE FROM BASIC ENTRY TO SCROLL
                             347  ;                             ; SCREEN
0127'  ED 4B 0028'           348          LD    BC,(SCRCTL)     ; GET CONTROL INFO.
                             349  ;
0128'                        350  SCROLL:                       ; HERE WITH CONTROLS IN BC
                             351                                ;  B=NO. OF LINES
                             352                                ;  C=STARTING LINE NO.
0128'  78                    353          LD    A,B             ; TEST VALIDITY
012C'  A7                    354          AND   A               ;
012D'  CA 00C4'              355          JP    Z,INVPAR        ; ERROR IF COUNT=0
0130'  81                    356          ADD   A,C
0131'  FE 01                 357          CP    1
0133'  DA 00C4'              358          JP    C,INVPAR        ; ERROR IF B+C<1
0136'  FE 19                 359          CP    25
0138'  D2 00C4'              360          JP    NC,INVPAR       ; ERROR IF B+C>24
013B'  CD 0141'              361          CALL  SCRL0           ; DO SCROLL
013E'  C3 008F'              362          JP    GOODRET         ; RETURN BC=0
                             363  ;
0141'  C5                    364  SCRL0   PUSH  BC
0142'  3E 18                 365          LD    A,SCRSZ         ; GET STARTING LINE IN INTERNAL FORMAT
0144'  91                    366          SUB   C
0145'  47                    367          LD    B,A
0146'  3A 0033'              368          LD    A,(LINLEN)
0149'  3C                    369          INC   A               ; COL. 0
014A'  4F                    370          LD    C,A
014B'  CD 063A'              371          CALL  LNBU
014E'  C1                    372          POP   BC
014F'  E5                    373          PUSH  HL              ; SAVE STARTING ADRS.
0150'  C5                    374          PUSH  BC              ; SAVE ORIG.BC FOR EXIT
0151'  78                    375  TESTA0  LD    A,L             ; TEST IF AT START OF BLOCK
0152'  A7                    376          AND   A
0153'  28 3C                 377          JR    Z,STBLK         ; YES
0155   07                    378          RLCA
0156'  07                    379          RLCA
0157'  07                    380          RLCA
0158'  4F                    381          LD    C,A             ; GET NO. OF LINES THIS BLOCK
0159'  3E 08                 382          LD    A,8
015B'  91                    383          SUB   C
015C'  B8                    384          CP    B               ; TEST AGAINST TOTAL LINES
015D'  38 4A                 385          JR    C,GRBLK         ; TOTAL GREATER THAN THIS BLOCK
015F'  78                    386  INSOBLK LD    A,B
0160'  06 00                 387          LD    B,0             ; REMAINING COUNT
0162'  C5                    388          PUSH  BC
0163'  47                    389  LOOP    LD    B,A             ; B=LINES THIS BLOCK
0164'  0E 08                 390          LD    C,8             ; C=SCAN COUNT
0166'  78                    391  LOOP0   LD    A,B
0167'  C5                    392          PUSH  BC              ; SAVE SCAN COUNT
0168'  0F                    393          RRCA
0169'  0F                    394          RRCA
016A'  0F                    395          RRCA                  ; CALCULATE NO. OF BYTES
016B'  4F                    396          LD    C,A
016C'  06 00                 397          LD    B,0             ; BC=32*# OF LINES
016E'  E8                    398  LOOP1   EX    DE,HL
016F'  21 FFE0               399          LD    HL,-20H         ; GET ADRS. OF PREV. LINE
0172'  19                    400          ADD   HL,DE
0173'  E8                    401          EX    DE,HL           ; TO DE
0174'  E5                    402          PUSH  HL              ; DISPLAY FILE ADRS.
0175'  ED B0                 403          LDIR                  ; DO MOVE
0177'  E1                    404          POP   HL
0178'  24                    405          INC   H               ; NEXT SCAN LINE
0179'  C1                    406          POP   BC              ; SCAN COUNT
017A'  00                    407          DEC   C
017B'  20 E9                 408          JR    NZ,LOOP0        ; MORE SCANS
017D'  C1                    409          POP   BC              ; RESIDUAL LINE COUNT
017E'  78                    410          LD    A,B
017F'  A7                    411          AND   A
0180'  28 04                 412          JR    Z,SCRLXT        ; DONE
0182'  2E 00                 413          LD    L,0             ; START OF NEXT BLOCK
0184'  18 C8                 414          JR    TESTA0          ; DO REMAINING LINE(S)
0186'  C1                    415  SCRLXT  POP   BC              ; ORIG.BC
0187'  01                    416          POP   DE              ; STARTING ADRS. TO DE
0188'  C5                    417          PUSH  BC              ; SAVE ORIG.BC
0189'  68                    418          LD    L,B             ; NO. OF LINES SCROLLED
018A'  26 00                 419          LD    H,0             ;
018C'  29                    420          ADD   HL,HL           ; 32*NO. OF LINES=
018D'  29                    421          ADD   HL,HL           ; NO. OF ATTRIBUTE BYTES
018E'  29                    422          ADD   HL,HL           ; TO BE SCROLLED
018F'  29                    423          ADD   HL,HL           ;
0190'  29                    424          ADD   HL,HL           ;
```

```
0191'  44
0192'  40
0193'  62
0194'  68
0195'  CC 0692'
0198'  E8
0199'  21 FFE0
019C'  19
019D'  E8
019E'  ED B0
01A0'  C1
01A1'  79
01A2'  80
01A3'  3D
01A4'  4F
01A5'  06 01
01A7'  18 45

01A9'  4F
01AA'  78
01AB'  91
01AC'  47
01AD'  C5
01AE'  79
01AF'  18 B2

01B1'  05
01B2'  C5
01B3'  DE 08
01B5'  C5
01B6'  E8
01B7'  21 F8E0
01BA'  19
01BB'  E8
01BC'  01 0020
01BF'  E5
01C0'  ED B0
01C2'  E1
01C3'  24
01C4'  C1
01C5'  0D
01C6'  20 ED
01C8'  C1
01C9'  78
01CA'  A7
01CB'  28 89
01CD'  11 F820
01D0'  19
01D1'  C3 0191'

01D4'
01D4'  ED 4B 800C'
01D8'

01D8'  78
01D9'  A7
01DA'  CA 00C4'
01DD'  81
01DE'  FE 01
01E0'  DA 00C4'
01E3'  FE 19
01E5'  D2 00C4'
01E8'  CD 01EE'
01EB'  C3 00BF'

01EE'  CD 06A2'
01F1'  C5
01F2'  3E 18
01F4'  91
01F5'  47
01F6'  3A 0033'
01F9'  3C
01FA'  4F
01FB'  CD 063A'
01FE'  50
01FF'  59
0200'  C1
0201'  05
0202'  7D
0203'  07
0204'  07
0205'  07
0206'  4F
0207'  3E 00
0209'  91
020A'  B8
020B'  38 47
020D'  78
020E'  06 00
0210'  C5
0211'  47
0212'  DE 08
0214'  78
0215'  C5
0216'  E6 07
0218'  0F
0219'  0F
021A'  0F
021B'  4F
021C'  06 00
021E'  0C
021F'  E5
0220'  54
0221'  5D
0222'  36 00
0224'  13
```

```
429            LD   B,M              ; COUNT TO BC
426            LD   C,L              ;
427            LD   M,D              ; ADRS. TO ML
428            LD   L,E              ;
429            CALL CALCATT          ; ATTRIBUTE ADRS. TO ML
430            EX   DE,ML            ;
431            LD   ML,-20H          ; ADRS. OF PREV.ATTR.BYTE
432            ADD  ML,DE            ;
433            EX   DE,ML            ; PREV.BYTE DE
434            LDIR                  ; SCROLL ATTRIBUTES
435            POP  BC               ; ORIG. BC
436            LD   A,C              ; STARTING LINE
437            ADD  A,B              ; ADD NO. OF LINES MOVED
438            DEC  A                ; LAST LINE SCROLLED
439            LD   C,A
440            LD   B,1              ; CLEAR 1 LINE
441            JR   CLRSC0           ; CLEAR VACATED LINE AND RETURN
442    ;
443    GRBLK   LD   C,A
444            LD   A,B
445            SUB  C                ; ADJUST TOTAL LINE COUNT BY NO.
446            LD   B,A              ; OF LINES THIS BLOCK
447            PUSH BC
448            LD   A,C              ; LOAD A NO. OF LINES THIS BLOCK
449            JR   LOOP
450    ;
451    STBLK   DEC  B                ; ADJUST TOTAL LINE COUNT-1
452            PUSH BC
453            LD   C,0              ; SCAN COUNT
454    STBLK0  PUSH BC               ; SAVE SCAN COUNT
455    STBLK1  EX   DE,ML
456            LD   ML,-720H
457            ADD  ML,DE
458            EX   DE,ML            ; ADRS. OF PREV. LINE
459            LD   BC,32            ; BYTE COUNT
460            PUSH ML
461            LDIR                  ; DO MOVE
462            POP  ML
463            INC  M                ; TO NEXT SCAN LINE
464            POP  BC
465            DEC  C                ; TEST IF MORE SCANS
466            JR   NZ,STBLK0
467            POP  BC               ; REMAINING LINE COUNT
468            LD   A,B
469            AND  A
470            JR   Z,SCRLXT
471            LD   DE,-780H         ; ADJUST TO LINE 1
472            ADD  ML,DE
473            JP   TESTAD           ; BC=LINE COUNT
474                                  ; ML=DF ADDRESS
475    ;
477            SUBTTL  CLEAR SCREEN
478    ;
479    ;
480    CLRSB0:                       ; HERE FROM BASIC ENTRY TO CLEAR SCREEN
481            LD   BC,(CCLRCTL)     ; GET CONTROL INFO.
482    ;
483    CLRSCN:                       ; ENTRY WITH BC=LINE COUNT AND
484                                  ; STARTING LINE NO. FOR CLEAR
485            LD   A,B
486            AND  A                ; TEST VALIDITY
487            JP   Z,INVPAR         ; ERROR IF COUNT=0
488            ADD  A,C
489            CP   1
490            JP   C,INVPAR         ; ERROR IF B+C<1
491            CP   25
492            JP   NC,INVPAR        ; ERROR IF B+C>24
493            CALL  CLRSC0          ; DO SERVICE
494            JP    GOODRET         ; RETURN BC=0
495    ;
496    ;
497    CLRSC0: CALL  LDATTR          ; GET ATTRIBUTE CTL
498            PUSH  BC
499            LD   A,SCRSZ          ; CONVERT TO INTERNAL FORMAT
500            SUB   C
501            LD   B,A
502            LD   A,(LINLEN)
503            INC  A                ; SET TO COL.0
504            LD   C,A
505            CALL  LNBO            ; GET ADDRESS
506            LD    D,B
507            LD    B,C             ; SAVE POSITION
508            POP   BC              ; ORIG. BC
509            PUSH  DE              ; SAVE CURSOR POSITION
510    CLRSC1  LD    A,L             ; GET # OF LINES THIS BLOCK
511            #LCA
512            #LCA
513            #LCA
514            LD    C,A
515            LD    A,B
516            SUB   C
517            CP    B               ; COMPARE TO TOTAL LINE COUNT
518            JR    C,CLRSC2
519    CLRSC7  LD    A,B             ; NO. OF LINES
520            LD    B,0             ; REMAINING LINES
521            PUSH  BC
522    CLRSC3  LD    B,A
523            LD    C,0             ; SCAN COUNT
524    CLRSC4  LD    A,B
525            PUSH  BC
526            AND   7
527            #RCA
528            #RCA
529            #RCA
530            LD    C,A             ; BC=32*NO. OF LINES
531            LD    B,0             ; (=0 IF 256 FOR 8 LINES)
532            DEC   C               ; ADJUST
533    CLRSC5  PUSH  HL
534            LD    D,H
535            LD    E,L
536            LD    (HL),00         ; CLEAR DF
537            INC   DE
```

```
0225'  ED B0                    938          LDIR
0227'  E1                       939          POP   HL
0228'  24                       940          INC   H                 ; NEXT SCAN ROW
0229'  C1                       941          POP   BC                ; POP SCAN COUNT
022A'  0D                       942          DEC   C
022B'  20 E7                    943          JR    NZ,CLRSC4
022D'  E5                       944          PUSH  HL                ; SAVE NEXT DF ADRS.
022E'  25                       945          DEC   H                 ; LAST SCAN ROW
022F'  CD 0692'                 946          CALL  CALC4TT           ; GET ADRS. IN ATTRIBUTE FILE
0232'  78                       947          LD    A,B               ; NO. OF LINES THIS BLOCK
0233'  E6 07                    948          AND   7
0235'  0F                       949          RRCA
0236'  0F                       950          RRCA
0237'  0F                       951          RRCA
0238'  4F                       952          LD    C,A               ; BC=32*NO. OF LINES
0239'  06 00                    953          LD    B,0
023B'  0D                       954          DEC   C                 ; ADJUST
023C'  54                       955          LD    D,H
023D'  5D                       956          LD    E,L
023E'  3A 0039'                 957          LD    A,(ATTRTT)        ; ATTRIBUTE VALUE
0241'  77                       958          LD    (HL),A            ; UPDATE ATTRIBUTE FILE
0242'  13                       959          INC   DE                ; NEXT BYTE
0243'  ED B0                    960          LDIR                    ;
0245'  E1                       961          POP   HL                ; DF ADRS.(LAST SCAN*100H)
0246'  C1                       962          POP   BC                ; TOTAL LINE COUNT
0247'  78                       963          LD    A,B
0248'  A7                       964          AND   A
0249'  28 05                    965          JR    Z,CLRSXT
024B'  2E 00                    966          LD    L,0               ; HL=START OF NEXT BLOCK
024D'  C3 0202'                 967          JP    CLRSC1            ; B=REMAINING LINE COUNT
0250'  C1                       968  CLRSXT  POP   BC                ; POSITION
0251'  C3 0628'                 969          JP    UPDPOSN           ; RETURN VIA UPDATE AND STORE POSITION
0254'  4F                       970  CLRSCY  LD    C,A               ; SAVE NO. LINES THIS BLOCK
0255'  78                       971          LD    A,B
0256'  91                       972          SUB   C                 ; ADJUST TOTAL LINE COUNT
0257'  47                       973          LD    B,A
0258'  C5                       974          PUSH  BC                ; REMAINING LINE COUNT
0259'  79                       975          LD    A,C               ; LINES THIS BLOCK
025A'  18 B5                    976          JR    CLRSC3            ;
                                978          SUBTTL "GET" ROUTINES
                                979  ;
                                980  ;
025C'                           981  GTCHR0:                         ; HERE FROM BASIC ENTRY TO GET CHARACTER
                                982                                  ; (RETURNS CODE FOR CHARACTER AT
                                983                                  ;  POSITION SPECIFIED IN GETCTL)
025C'  ED 4B 0019'              984          LD    BC,(GETCTL)       ; GET CONTROL INFO.
                                985  ;
0260'                           986  GTCHAR:                         ; ENTRY WITH POSITION IN BC
0260'  CD 060C'                 987          CALL  TSTPAR            ; TEST PARAMETERS
0263'  CD 061D'                 988          CALL  CONVFM            ; CONVERT TO INTERNAL FORMAT
0266'  CD 0620'                 989          CALL  CALCPOS           ; GET DISPLAY FILE ADRS.
0269'  ED 5B 002F'              990          LD    DE,(CHTBL)        ; CHAR.TABLE
026D'  06 60                    991          LD    B,96              ; NO. OF PRINTABLE CHARACTERS
026F'  3E 80                    992          LD    A,80H             ; SET ADJUSTMENT INDEX
0271'  32 003A'                 993  GTCH1   LD    (GTINDX),A
0274'  14                       994          INC   D                 ; ADJUST TO START OF TABLE
0275'  EB                       995          EX    DE,HL             ; DF ADRS.IN DESCHAR.SET IN HL
0276'  C5                       996  GTCH2   PUSH  BC                ; SAVE COUNT
0277'  D5                       997          PUSH  DE                ; SAVE ADRS. IN DF
0278'  E5                       998          PUSH  HL                ; SAVE ADRS. IN CHAR.TABLE
0279'  1A                       999          LD    A,(DE)            ; SCAN ROW FROM DF
027A'  AE                      1000          XOR   (HL)              ; TEST AGAINST CHAR. SET
027B'  28 10                   1001          JR    Z,GTCH3           ; MATCH
027D'  F5                      1002          PUSH  AF
027E'  3A 003A'                1003          LD    A,(GTINDX)
0281'  FE 90                   1004          CP    90H               ; TEST IF STD.GRAPHICS
0283'  20 03                   1005          JR    NZ,GTCH23
0285'  F1                     1006          POP   AF
0286'  18 27                  1007          JR    GTCH4             ; DO NOT TEST FOR INVERSE
0288'  F1                     1008  GTCH23  POP   AF
0289'  3C                     1009          INC   A                 ; TEST INVERSE
028A'  20 23                   1010          JR  NZ,GTCH4
028C'  3D                      1011          DEC   A                 ; A=-1 FOR INVERSE
028D'  4F                      1012  GTCH3   LD    C,A               ; C=0 FOR MATCH  -1 FOR INVERSE
028E'  06 07                   1013          LD    B,7
0290'  14                      1014  GTCH31  INC   D                 ; NEXT SCAN IN DF
0291'  23                      1015          INC   HL                ; NEXT BYTE IN CHAR.SET
0292'  1A                     1016          LD    A,(DE)
0293'  AE                     1017          XOR   (HL)
0294'  A9                     1018          XOR   C
0295'  20 18                  1019          JR    NZ,GTCH4          ; NO MATCH
0297'  10 F7                  1020          DJNZ  GTCH31    ; FOR NEXT SCAN
                               1021                          ;
                               1022                          ; HERE IF MATCH WITH (SP)=CHAR IN CHAR SET
                               1023                          ; (SP+2)=CHAR.IN DISPLAY FILE (SP+5)=CHAR.
                               1024                          ; COUNT
0299'  79                      1025          LD    A,C         ; A=-1 IF MATCH ON INVERSE
029A'  C1                      1026          POP   BC
029B'  C1                      1027          POP   BC
029C'  C1                      1028          POP   BC
029D'  4F                      1029          LD    C,A       ; B=COUNT C=-1 IF MATCH ON INVERSE
029E'  3A 003A'                1030          LD    A,(GTINDX)        ; ADJUST CHAR.CODE
02A1'  90                      1031          SUB   B         ; A=CHAR.CODE
02A2'  FE 20                   1032          CP    20H       ; TEST IF SPACE
02A4'  20 05                   1033          JR    NZ,GTCH32
02A6'  0C                      1034          INC   C         ; TEST IF MATCH ON INVERSE SPACE
02A7'  20 02                   1035          JR    NZ,GTCH32
02A9'  3E 8F                   1036          LD    A,8FH     ; SET CODE FOR GRAPHICS BLOCK
02AB'  4F                      1037  GTCH32  LD    C,A
02AC'  06 00                   1038          LD    B,0
02AE'  C9                      1039          RET             ; CHAR.CODE IN A AND IN BC
                               1040  ;
02AF'  E1                      1041  GTCH4   POP   HL        ; HERE WHEN NO MATCH
02B0'  11 0008                 1042          LD    DE,8      ; LOOK AT NEXT CHAR. IN SET
02B3'  19                      1043          ADD   HL,DE             ; CHAR. SET
02B4'  D1                      1044          POP   DE                ; CHAR. IN DF
02B5'  C1                      1045          POP   BC                ; CHAR.COUNT
02B6'  10 E8                   1046          DJNZ  GTCH2     ; LOOK AGAIN
                               1047                          ;
                               1048                          ; HERE TO TEST IF DONE
02B8'  EB                      1049          EX    DE,HL     ; DF ADRS.TO HL
02B9'  3A 003A'                1050          LD    A,(GTINDX)
02BC'  FE 80                   1051          CP    80H       ; TEST IF STD.CHR.SET
```

```
02BE'  20 0A                652             JR    NZ,GTCH5  ; TRY GRAPHICS
02C0'  ED 5B 0031'          653             LD    DE,(GRTBL); STD.GRAPHICS TABLE
02C4'  06 10                654             LD    B,16      ; NO. OF ENTRIES
02C6'  3E 90                655             LD    A,90H     ; INDEX
02C8'  18 A7                656             JR    GTCH1     !
02CA'  FE 90                657 GTCH5       CP    90H       ; TEST IF STD.GRAPHICS
02CC'  20 08                658             JR    NZ,GTCH6  ; DONE IF NOT
02CE'  ED 5B 5C78           659             LD    DE,(UDG)  ; TRY USER-DEFINED GRAPHICS AREA
02D2'  15                   660             DEC   D         ; ADJUST ADDRESS-100H
02D3'  06 15                661             LD    B,21      ; NO OF ENTRIES
02D5'  3E A5                662             LD    A,0A5H    ; INDEX
02D7'  18 98                663             JR    GTCH1
02D9'  48                   664 GTCH6       LD    C,B       ; HERE WHEN NO MATCH ANYWHERE
02DA'  AF                   665             XOR   A         ; SET BC AND A=ZERO
02DB'  C9                   666             RET
                            667
                            668                             ; GET ATTRIBUTE BYTE
                            669 !                           ; HERE FROM BASIC ENTRY TO GET ATTRIBUTE
                            670 !
02DC'  ED 4B 0019'          671 GETATB0: LD   BC,(GETCTL)
                            672 !                           ; HERE WITH LIN/COL IN BC
                            673 !
02E0'  CD 060C'             674 GETATT: CALL  TSTPAR    ; TEST PARAMETERS
02E3'  CD 0610'             675             CALL  CONVPM    ; CONVERT TO INTERNAL FORMAT
02E6'  CD 0620'             676             CALL  CALCPOS   ; CALCULATE DF POSITION
02E9'  CD 0692'             677             CALL  CALCATT   ; CALCULATE ATTRIBUTE POSITION
02EC'  7E                   678             LD    A,(HL)    ; ATTRIBUTE BYTE TO A
02ED'  4F                   679             LD    C,A       ; PASS BACK IN BC
02EE'  06 00                680             LD    B,0
02F0'  C9                   681             RET
                            682 !                           ; GET CURSOR POSITION
02F1'                       683 GETCB0:                     ; HERE FROM BASIC ENTRY
02F1'                       684 GETCUR:
02F1'  ED 4B 0034'          685             LD    BC,(CURPOS)       ; GET INTERNAL POSITION
02F5'  CD 0610'             686             CALL CONVPM             ; CONVERT TO USER FORMAT
02F8'  3A 0033'             687             LD    A,(LINLEN)
02FB'  B9                   688             CP    C                 ; TEST IF END OF LINE(=32)
02FC'  20 08                689             JR    NZ,GETC1          ; NO
02FE'  0E 00                690             LD    C,0               ; NEXT POSN.START OF NEXT LINE
0300'  78                   691             LD    A,B
0301'  3C                   692             INC   A                 ; BUMP TO NEXT LINE
0302'  47                   693             LD    B,A
0303'  FE 18                694             CP    24                ; TEST IF OFF SCREEN
0305'  38 02                695             JR    C,GETC1           ; NO
0307'  06 17                696             LD    B,23              ; POSN. IS AT BOTTOM LINE
0309'  ED 43 0007'          697 GETC1   LD    (LINCOL),BC       ; STORE FOR BASIC PROGRAM
030D'  C9                   698             RET                     ; VALUES ALSO IN BC
                            699 !
                            701             SUBTTL  VIDEO MODE CONTROL
                            702 !
                            703 !
030E'                       704 STMOB0:                     ; HERE FROM BASIC ENTRY WITH MODE IN
                            705                             ; "VIMODE"
030E'  3A 0024'             706             LD  A,(VIMODE)
                            707 !
0311'                       708 SETMODE:                    ; ENTRY WITH MODE IN A
                            709 !
0311'  47                   710             LD    B,A         ; MODE TO B
0312'  A7                   711             AND   A
0313'  28 15                712             JR    Z,SETMD0    ; TEST FOR VALIDITY
0315'  FE 80                713             CP    80H         !
0317'  28 11                714             JR    Z,SETMD0    !
0319'  FE 01                715             CP    1
031B'  28 0D                716             JR    Z,SETMD0
031D'  FE 04                717             CP    4
031F'  CA 038A'             718             JP    Z,PRPV      ; HANDLE 4 AND 8
0322'  FE 08                719             CP    8
0324'  CA 038A'             720             JP    Z,PRPV      !
0327'  C3 00C4'             721             JP    INVPAR      ; INVALID PARAMETER
032A'  3A 9CC2              722 SETMD0  LD    A,(VIDMOD)  ; TEST CURRENT MODE
032D'  A7                   723             AND   A
032E'  20 5A                724             JR    NZ,SETMD2   ; DF2 OPEN
0330'  B0                   725             OR    B           ; CURRENT MODE=0 - TEST IF NEW MODE=0
0331'  CA 008F'             726             JP    Z,GOODRET   ; NO ACTION
                            727 !
                            728 !                           ; INITIAL OPENING OF 2ND DF
0334'  21 1701              729 SETMG1  LD    HL,SCINIT   ; INITIALIZE VARIABLES
0337'  22 0028'             730             LD    (SCRCTL),HL ; SCROLL CONTROL
033A'  3E 17                731             LD    A,23        !
033C'  32 0020'             732             LD    (BOTLN),A   ; BOTTOM LINE
033F'  3E 01                733             LD    A,1         !
0341'  32 002E'             734             LD    (SCRLCT),A  ; SCROLL COUNT
0344'  3E 20                735             LD    A,32        !
0346'  32 0033'             736             LD    (LINLEN),A  ; LINE LENGTH
0349'  21 1000              737             LD    HL,CLINIT   !
034C'  22 000C'             738             LD    (CLRCTL),HL ; CLEAR SCREEN CONTROL
034F'  AF                   739             XOR   A
0350'  32 0038'             740             LD    (STRGCT),A  ; WRSTRG REMAINING COUNT
0353'  32 003C'             741             LD    (STRGCT+1),A !
0356'  21 3C00              742             LD    HL,CHRSET   ; STD.CHAR.SET
0359'  22 002F'             743             LD    (CHTBL),HL  !
035C'  21 05E0'             744             LD    HL,GRPHST   ; STD. GRAPHICS SET
035F'  22 0031'             745             LD    (GRTBL),HL  !
                            746 !                           !
                            747 !                           ; INITIALIZE BASIC INPUTS
0362'  32 0000'             748             LD    (DATAB),A   ; DATA BYTE
0365'  32 0007'             749             LD    (LINCOL),A  ; COLUMN
0368'  32 0008'             750             LD    (LINCOL+1),A ! LINE
036B'  32 0019'             751             LD    (GETCTL),A  ! "GET" COLUMN
036E'  32 001A'             752             LD    (GETCTL+1),A ! "GET" LINE
                            753 !                           ; TEST IF ENOUGH MEMORY
0371'  21 12C0              754             LD    HL,INSERTSZ
0374'  11 0840              755             LD    DE,MOVESZ
0377'  19                   756             ADD   HL,DE       ; HL=TOTAL ROOM NEEDED
0378'  ED 5B 5C65           757             LD    DE,(STKEND)
037C'  19                   758             ADD   HL,DE       ; ADD MEMORY IN USE
037D'  DA 0385'             759             JP    C,EXTNRM    ; NOT ENOUGH MEMORY TO OPEN DF2
0380'  ED 5B 5C82           760             LD    DE,(RAMTOP)
0384'  A7                   761             AND   A
0385'  ED 52                762             SBC   HL,DE
0387'  D2 0395'             763             JP    NC,EXTNRM ; NOT ENOUGH MEMORY
038A'  78                   764 SETMD2  LD    A,B         ; MODE TO A
038B'  CC 05F3'             765             CALL  ENBLEXT   ; ENABLE ROM EXT.
038E'  CD 05CD'             766             CALL  GETVAL    ; H/W VALUE TO B; VIDMOD VALUE TO C
```

```
0391' C5                767             PUSH  BC        ; SAVE
0392' 78                768             LD    A,B
0393' A7                769             AND   A         ; TEST IF 0
0394' 3C 01             770             JR    NZ,SETM21
0396' 79                771             LD    A,C       ; VALUE FOR CHNGVID IN C
0397' CD 0E8E           772   SETM21    CALL  CHNGVID   ; CALL RTN. TO OPEN/CLOSE 2ND DP
039A' CD 0600'          773             CALL  ENBLMSME  ; REENABLE HOME BANK
039D' C1                774             POP   BC
039E' 79                775             LD    A,C
039F' 32 5CC2           776   SETM03    LD    (VIOMOD),A; UPDATE VIOMOD
03A2' 3A 0033'          777   UPDATE    LD    A,(LINLEN)        ; GET LINE LENGTH
03A5' 3C                778             INC   A
03A6' 4F                779             LD    C,A               ; CCLUMN
03A7' 06 18             780             LD    B,SCRSZ           ; BC=HOME POSITION (LN.0/CCL.0)
03A9' 21 0000           781             LD    HL,0
03AC' 22 5C7D           782             LD    (CCCORDS),HL      ; INITIALIZE PLOT POSITION
03AF' CD 0628'          783             CALL  UPDPOSN           ; UPDATE AND STORE HOME POSITION
03B2' C3 08BF'          784             JP    GOCORET           ; RETURN BC=0
03B5' 0E 02             785   EXTNRM    LD    C,2               ; RETURN BC=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 VIOMOD   NEW VIOMOD
                        794   ;                               ; -----     --------------   ----------
                        795   ;
                        796   ;                               ;   4           80             80
                        797   ;                               ;   4           81             84
                        798   ;                               ;   4           84             84
                        799   ;                               ;   4           85             80
                        800   ;
                        801   ;                               ;   5           80             85
                        802   ;                               ;   5           81             81
                        803   ;                               ;   5           84             81
                        804   ;                               ;   5           85             85
                        805   ;
03BA' 3A 5CC2           806   PRPV      LD    A,(VIOMOD)        ; ONLY VALID IF VIOMOD=80,81,84 OR 85
03BD' 4F                807             LD    C,A               ; CURRENT VIOMOD TO C
03BE' CB 7F             808             BIT   7,A
03C0' CA 00C4'          809             JP    Z,INVPAR          ; INVALID PARAMETER
03C3' 78                810             LD    A,B               ; MODE 4 OR 5 TO A
03C4' FE 04             811             CP    4
03C6' 20 0A             812             JR    NZ,DTAOP2         ; MODE 5 - DATA TO DP2
                        813   ;                               ; MODE 4 - DATA TO DP1
03C8' CB 41             814   DTAOP1    BIT   0,C               ; TEST IF DP1 ALREADY DEST.
03CA' CA 08BF'          815             JP    Z,GOCORET         ; NO ACTION
03CD' A9                816             XOR   C                 ; SET VIOMOD=80 OR 84
03CE' E6 FE             817             AND   0FEH              ; BIT 0=0
03D0' 18 CD             818             JR    SETM03            ; UPDATE VARIABLES
                        819   ;
                        820   ;                               ; MODE 5 - DATA TO DP2
03D2' CB 41             821   DTAOP2    BIT   0,C               ; TEST IF DP2 ALREADY DEST.
03D4' C2 08BF'          822             JP    NZ,GOCORET        ; NO ACTION
03D7' A9                823             XOR   C                 ; SET VIOMOD=81 OR 85
03D8' 18 C5             824             JR    SETM03            ; UPDATE VARIABLES
                        825   ;
                        827             SUBTTL  COPY SCREEN
                        828   ;
                        829   ;
                        830   ;                               ; COPY LINES WITHIN OR BETWEEN
                        831   ;                               ; DISPLAY FILES
                        832   ;
                        833   ;
03DA'                   834   COPYSCR:                        ; HERE FROM BASIC ENTRY
03DA' 2A 5C5D           835             LD    HL,(CH_ADD)       ; SAVE CURRENT POINTER
03DD' E5                836             PUSH  HL
03DE' CD 0686'          837             CALL  GETSTRG           ; GET STRING OF PARAMETERS
03E1' 20 09             838             JR    NZ,COPY01         ; FOUND STRING
03E3' E1                839             POP   HL                ; RESTORE CH_ADD AND RETURN
03E4' 22 5C5D           840             LD    (CH_ADD),HL       ; ERROR (NO STRING OR NULL STRING)
03E7' C9                841             RET                     ; (BC CONTAINS STATUS)
                        842   ;
03E8' =0004             843   COPYWK    DEFS  4                 ; WORK AREA FOR BASIC PARAMETERS
                        844   ;
                        845   ;
03EC' 22 5C5D           846   COPY01    LD    (CH_ADD),HL       ; ADRS. OF STRING TO CH_ADD
03EF' CD 0418'          847             CALL  GETNO             ; GET PARAMETERS
03F2' 32 03E9'          848             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 0418'          851             CALL  GETNO
03FE' 32 03EB'          852             LD    (COPYWK+3),A      ; SOURCE DP
0401' CD 0418'          853             CALL  GETNO
0404' 32 03EA'          854             LD    (COPYWK+2),A      ; DEST. DP
0407' CD 0418'          855             CALL  GETNO
040A' E1                856             POP   HL                ; RESTORE CH_ADD
040B' 22 5C5D           857             LD    (CH_ADD),HL
040E' ED 4B 03E8'       858             LD    BC,(COPYWK)
0412' ED 5B 03EA'       859             LD    DE,(COPYWK+2)
0416' 18 25             860             JR    COPYSC            ; TEST VALIDITY AND DO COPY
                        861   ;
0418' 2A 5C5D           862   GETNO     LD    HL,(CH_ADD)       ; LOAD ADDRESS
041B' 7E                863             LD    A,(HL)            ; GET CHAR. FROM STRING
041C' FE 2C             864             CP    2CH               ; TEST IF COMMA
041E' 28 0F             865             JR    Z,GETN2
0420' FE 30             866             CP    30H               ; TEST IF DIGIT
0422' 38 11             867             JR    C,COPERR          ; INVALID PARAMETER
                        868   ;                               ; OR END OF STRING
0424' FE 3A             869             CP    3AH
0426' 30 0C             870             JR    NC,COPERR
0428' CC 30F9           871   GETN1     CALL  Z,NEINT           ; CONVERT TO BINARY
042B' CD 3193           872             CALL  FP2A
042E' D0                873             RET   NC                ; VALID DIGIT IN A
042F' 23                874   GETN2     INC   HL
0430' 22 5C5D           875             LD    (CH_ADD),HL
0433' 18 E3             876             JR    GETNO             ; TEST NEXT CHAR.
                        877   ;
0435' E1                878   COPERR    POP   HL                ; DISCARD RETURN FROM GETNO
```

```
0436*  81                    879  COPYER1  POP     HL              ; RESTORE CH_ADD
0437*  22 5C5D               880           LD      (CH_ADD),HL
0438*  C3 00C4*              881           JP      INVPAR          ; RETURN BC=1 FOR INVALID
                             882                                   ; PARAMETER(S)
                             883  ;
                             884  ;
0439*                        885  COPYSC:                          ; ENTRY WITH PARAMETERS IN REGS.
                             886  ;                               ; AS ABOVE
0439*  A7                    887           AND     A               ; TEST VALIDITY
043E*  CA 00C4*              888           JP      Z,INVPAR        ; ERROR IF LINE COUNT=0
0441*  FE 19                 889           CP      25
0443*  D2 00C4*              890           JP      NC,INVPAR       ; ERROR IF > 24
0446*  F5                    891           PUSH    AF
0447*  CD 05A8*              892           CALL    GTADRS          ; GET DF ADRS. FOR SOURCE
044A*  DA 00C4*              893           JP      C,INVPAR        ; INVALID PARAMETER DETECTED
044D*  E5                    894           PUSH    HL              ; SAVE SOURCE ADDRESS
044E*  41                    895           LD      B,C             ; DEST. LINE
044F*  53                    896           LD      D,E             ; DEST. DF
0450*  CD 05A8*              897           CALL    GTADRS          ; GET DF ADRS. FOR DESTINATION
0453*  DA 00C4*              898           JP      C,INVPAR
0456*  D1                    899           POP     DE              ; SOURCE ADRS.
0457*  EB                    900           EX      DE,HL           ; HL=SOURCE  DE=DEST.
0458*  F1                    901           POP     AF              ; GET COUNT
0459*  47                    902           LD      B,A
045A*  E5                    903           PUSH    HL              ; SAVE STARTING ADDRESSES
045B*  D5                    904           PUSH    DE              ;
045C*  C5                    905           PUSH    BC              ; SAVE ORIG. LINE COUNT
045D*  7D                    906  COPYS1   LD      A,L             ; CALCULATE # OF LINES SOURCE BLOCK
045E*  07                    907           RLCA
045F*  07                    908           RLCA
0460*  07                    909           RLCA
0461*  4F                    910           LD      C,A             ;
0462*  3E 08                 911           LD      A,8
0464*  91                    912           SUB     C
0465*  4F                    913           LD      C,A             ; # OF LINES THIS BLOCK
0466*  C5                    914           PUSH    BC              ; SAVE NO.OF LINES/# OF LINES SOURCE BLK.
0467*  B8                    915           CP      B               ; COMPARE TO TOTAL LINES
0468*  38 71                 916           JR      C,COPYS3        ; TOTAL LINES > THIS BLOCK
046A*  78                    917           LD      A,B             ; CALCULATE # OF LINES DEST.BLOCK
046B*  07                    918           RLCA
046C*  07                    919           RLCA
046D*  07                    920           RLCA
046E*  4F                    921           LD      C,A
046F*  3E 08                 922           LD      A,8
0471*  91                    923           SUB     C               ; A=# OF LINES DEST.BLOCK
0472*  C1                    924           POP     BC              ; B=TOTAL LINES
0473*  4F                    925           LD      C,A             ; C=# OF LINES DEST BLOCK
0474*  B8                    926           CP      B               ; B=TOTAL LINES TO BE COPIED
0475*  38 79                 927           JR      C,COPYS7        ; TOTAL > DEST.BLOCK - USE DEST.LINE CNT.
                             928  ;
                             929  ;                               ; COPY IS WITHIN BOTH SOURCE AND DEST.BLOCKS
                             930  ;
0477*  78                    931  COPYS2   LD      A,B             ; NO. OF LINES
0478*  06 00                 932           LD      B,0             ; REMAINING LINES
047A*  C5                    933           PUSH    BC              ; SAVE REMAINING LINE COUNT
047B*  47                    934           LD      B,A             ; NO. OF LINES
047C*  0E 08                 935  COPYS3   LD      C,8             ; SCAN COUNT
047E*  78                    936           LD      A,B             ; NO. OF LINES
047F*  C5                    937  COPYS4   PUSH    BC              ; SAVE SCAN COUNT
0480*  E6 07                 938           AND     7
0482*  0F                    939           RRCA
0483*  0F                    940           RRCA
0484*  0F                    941           RRCA
0485*  4F                    942           LD      C,A             ; BC=32 * NO. OF LINES
0486*  06 00                 943           LD      B,0             ; (C=0 IF 256 FOR 8 LINES)
0488*  A7                    944           AND     A               ; TEST IF 8
0489*  20 02                 945           JR      NZ,COPY41
048B*  06 01                 946           LD      B,1             ; SET TO 256
048D*  E5                    947  COPY41   PUSH    HL              ; SAVE ADDRESSES
048E*  D5                    948           PUSH    DE
048F*  ED B0                 949           LDIR                    ; COPY "BC" BYTES
0491*  D1                    950           POP     DE              ; RESTORE ADDRESSES
0492*  E1                    951           POP     HL
0493*  14                    952           INC     D               ; ADJUST FOR NEXT SCAN  LINE
0494*  24                    953           INC     H               ;  (+100H)
0495*  C1                    954           POP     BC              ; SCAN COUNT
0496*  0D                    955           DEC     C               ; DECREMENT SCAN COUNT
0497*  20 E5                 956           JR      NZ,COPYS4       ; NEXT SCAN ROW
0499*  78                    957           LD      A,B             ; SAVE # OF LINES JUST  MOVED
049A*  C1                    958           POP     BC              ; RESIDUAL LINE COUNT
049B*  4F                    959           LD      C,A             ;
049C*  78                    960           LD      A,B             ;
049D*  A7                    961           AND     A
049E*  28 22                 962           JR      Z,COPYXT        ; DONE
04A0*  C5                    963           PUSH    BC              ; B=REMAINING LINE  COUNT
04A1*  79                    964           LD      A,C
04A2*  E6 07                 965           AND     7
04A4*  0F                    966           RRCA
04A5*  0F                    967           RRCA
04A6*  0F                    968           RRCA
04A7*  4F                    969           LD      C,A
04A8*  06 00                 970           LD      B,0
04AA*  A7                    971           AND     A
04AB*  20 02                 972           JR      NZ,COPY42
04AD*  06 01                 973           LD      B,1             ; BC = NO. OF BYTES JUST MOVED
04AF*  25                    974  COPY42   DEC     H               ; ADJUST ADDRESSES TO START
04B0*  09                    975           ADD     HL,BC           ; OF NEXT LINE IN EACH FILE
04B1*  3E 78                 976           LD      A,78H
04B3*  A4                    977           AND     H
04B4*  67                    978           LD      H,A
04B5*  E5                    979           PUSH    HL              ; SAVE ADJUSTED SOURCE ADRS
04B6*  15                    980           DEC     D
04B7*  EB                    981           EX      DE,HL           ; DEST. ADRS. TO HL
04B8*  09                    982           ADD     HL,BC
04B9*  3E 78                 983           LD      A,78H
04BB*  A4                    984           AND     H
04BC*  67                    985           LD      H,A
04BD*  D1                    986           POP     DE              ; SOURCE ADRS. TO DE
04BE*  EB                    987           EX      DE,HL           ; HL=SOURCE  DE=DEST.
04BF*  C1                    988           POP     BC
04C0*  18 98                 989           JR      COPYS1          ; B=REMAINING LINE COUNT
04C2*  C1                    990  COPYXT   POP     BC              ; ORIG.LINE COUNT
04C3*  46                    991           LD      L,B             ; CALCULATE # OF ATTRIBUTE BYTES
```

```
04C4'  26 00              992          LD      H,0
04C6'  29                 993          ADD     HL,HL
04C7'  29                 994          ADD     HL,HL
04C8'  29                 995          ADD     HL,HL
04C9'  29                 996          ADD     HL,HL
04CA'  29                 997          ADD     HL,HL           ; 32 * NO. OF LINES
04CB'  44                 998          LD      B,H
04CC'  4D                 999          LD      C,L
04CD'  E1                1000          POP     HL              ; DEST.ADDRESS TO HL
04CE'  CD 0692'          1001          CALL    CALCATT         ; ADRS. OF ATTRIBUTE FILE
0431'  EB                1002          EX      DE,HL           ; DEST.ADRS. TO DE
0432'  E1                1003          POP     HL              ; SOURCE ADDRESS
0433'  CD 0692'          1004          CALL    CALCATT         ; ADRS. OF ATTRIBUTE FILE
0436'  ED B0              1005          LDIR                    ; COPY ATTRIBUTES
0438'  C3 008F'          1006          JP      GOODRET         ; RETURN BC=0
                          1007  ;
0408'  41                 1008  COPYSS  LD      B,C             ; # OF LINES SOURCE BLOCK TO B
040C'  78                 1009          LD      A,E             ; CALCULATE # OF LINES DEST. BLOCK
040D'  07                 1010          RLCA
040E'  07                 1011          RLCA
040F'  07                 1012          RLCA
04E0'  4F                 1013          LD      C,A
04E1'  3E 08              1014          LD      A,8
04E3'  91                 1015          SUB     C               ; A=8 LINES DEST.BLOCK
04E4'  4F                 1016          LD      C,A             ; SAVE IN C
04E5'  B8                 1017          CP      B               ; COMPARE TO LINES SOURCE BLOCK
04E6'  38 08              1018          JR      C,COPYS7        ; DEST.SIZE SMALLER - USE DEST.LINE CNT.
04E8'  C1                 1019          POP     BC              ; B=TOTAL LINE COUNT/C=SOURCE LINE CNT.
04E9'  78                 1020  COPYS6  LD      A,B
04EA'  91                 1021          SUB     C
04EB'  47                 1022          LD      B,A             ; B=REMAINING LINECOUNT
04EC'  C5                 1023          PUSH    BC
04ED'  79                 1024          LD      A,C             ; # OF LINES
04EE'  18 88              1025          JR      COPYSJ          ; COPY 'A' LINES
04F0'  79                 1026  COPYS7  LD      A,C             ; # LINES DEST.BLK
04F1'  C1                 1027          POP     BC              ; B=TOTAL LINE COUNT
04F2'  4F                 1028          LD      C,A             ; C=DEST. LINE CNT.
04F3'  18 F4              1029          JR      COPYS6
                          1030
                          1032          SUBTTL  EXCHANGE SCREEN
                          1033  ;
                          1034  ;                               ; EXCHANGE LINES WITHIN OR BETWEEN
                          1035  ;                               ; DISPLAY FILES
                          1036  ;
04F5'                     1037  EXCHS0                          ; ENTRY FROM BASIC - GET PARAMETERS
                          1038  ;                               ; FROM STRING TO APPROPRIATE REGS.
                          1039  ;
04F5'  2A 9C30            1040          LD      HL,(CH_ADD)     ; SAVE CURRENT POINTER
04F8'  E5                 1041          PUSH    HL
04F9'  CD 0688'           1042          CALL    GETSTRG         ; GET STRING OF PARAMETERS
04FC'  20 09              1043          JR      NZ,EXCH01       ; FOUND STRING
04FE'  E1                 1044          POP     HL              ; RESTORE CH_ADD AND RETURN
04FF'  22 9C30            1045          LD      (CH_ADD),HL     ; ERROR (NO STRING OR NULL STRING)
0502'  C9                 1046          RET                     ; (BC CONTAINS STATUS)
                          1047  ;
0503'  22 9C30            1048  EXCH01  LD      (CH_ADD),HL     ; ADDRESS OF STRING
0506'  CD 0418'           1049          CALL    GETNO           ; GET PARAMETERS
0509'  32 03E8'           1050          LD      (COPYWK),A      ; SOURCE LINE NO.
050C'  CD 0418'           1051          CALL    GETNO
050F'  32 03EA'           1052          LD      (COPYWK+2),A    ; DEST. LINE NO.
0512'  CD 0418'           1053          CALL    GETNO           ;
0515'  32 03E9'           1054          LD      (COPYWK+1),A    ; SOURCE OF (1 OR 2)
0518'  CD 0418'           1055          CALL    GETNO
051B'  32 03EB'           1056          LD      (COPYWK+3),A    ; DEST. OF
051E'  CD 0418'           1057          CALL    GETNO           ; NO. OF LINES
0521'  E1                 1058          POP     HL              ; RESTORE CH_ADD
0522'  22 9C30            1059          LD      (CH_ADD),HL     ;
0525'  ED 4B 03E8'        1060          LD      BC,(COPYWK)     ; LOAD REGS.
0529'  ED 5B 03EA'        1061          LD      DE,(COPYWK+2)
                          1062  ;                               ;
                          1063  ;
052D'                     1064  EXCHSC1                         ; ROUTINE TO EXCHANGE LINES IN THE
                          1065  ;                               ; DISPLAY FILE(S).
                          1066  ;                               ;
                          1067  ;                               ; INPUTS:  B=OF FOR SOURCE 1 (1 OR 2)
                          1068  ;                               ;          C=LINE NO. FOR SOURCE 1
                          1069  ;                               ;          D=OF FOR SOURCE 2 (1 OR 2)
                          1070  ;                               ;          E=LINE NO. FOR SOURCE 2
                          1071  ;                               ;          A=COUNT (# OF LINES TO BE
                          1072  ;                               ;            EXCHANGED
                          1073  ;
                          1074  ;
                          1075  ;
                          1076  ;
                          1077  ;
0520'  A7                 1078          AND     A               ; TEST VALI COUNT
0521'  CA 00C4'          1079          JP      Z,INVPAR
0531'  FE 19              1080          CP      25
0533'  D2 00C4'          1081          JP      NC,INVPAR
0536'  F5                 1082          PUSH    AF              ; SAVE LINE COUNT
0537'  D5                 1083          PUSH    DE              ; SAVE SOURCE 2 INFO.
0538'  50                 1084          LD      D,B             ; OF
0539'  41                 1085          LD      B,C             ; LINE NO
053A'  CD 03A8'           1086          CALL    GTADRS          ; GET OF ADRS.FOR SOURCE 1
053D'  DA 00C4'           1087          JP      C,INVPAR        ; ERROR DETECTED
0540'  D1                 1088          POP     DE              ; SOURCE 2 INFO.
0541'  E5                 1089          PUSH    HL              ; SAVE SOURCE 1 ADRS.
0542'  43                 1090          LD      B,E             ; LINE NO.
0543'  CD 03A8'           1091          CALL    GTADRS          ; GET OF ADRS. FOR SOURCE 2
0546'  DA 00C4'           1092          JP      C,INVPAR
0549'  D1                 1093          POP     DE              ; ADDRESSES IN DE AND HL
054A'  F1                 1094          POP     AF              ; LINE COUNT IN A
054B'  E5                 1095          PUSH    HL              ; SOURCE 2 STARTING ADRS. FOR EXIT
054C'  D5                 1096          PUSH    DE              ; SOURCE 1 STARTING ADRS. FOR EXIT
054D'  F5                 1097          PUSH    AF              ; ORIG.LINE COUNT FOR EXIT
054E'  F5                 1098          PUSH    AF              ; WORKING LINE COUNT
054F'  06 08              1099  EXCH0   LD      B,8             ; SCAN COUNT
0551'  E5                 1100  EXCH1   PUSH    HL              ; SAVE ADDRESSES AND SCAN COUNT
0552'  D5                 1101          PUSH    DE
0553'  C5                 1102          PUSH    BC
```

```
0554'  01 0020          1103            LD      BC,32           ; NO.OF BYTES IN SCAN ROW FOR
                        1104    ;                               ; A LINE ON THE SCREEN
0557'  CD 0593'         1105            CALL    EXCHLP          ; EXCHANGE BYTES
055A'  C1               1106            POP     BC              ; SCAN COUNT IN B
055B'  01               1107            POP     DE
055C'  81               1108            POP     HL              ; ADDRESSES
055D'  14               1109            INC     D               ; ADJUST TO NEXT SCAN ROW
055E'  24               1110            INC     H
055F'  10 F0            1111            DJNZ    EXCH1           ; DO NEXT ROW
0561'  F1               1112            POP     AF              ; LINE COUNT IN A
0562'  15               1113            DEC     D               ; ADJUST ADDRESSES BACK 1 SCAN ROW
0563'  25               1114            DEC     H
0564'  3D               1115            DEC     A               ; ADJUST LINE COUNT
0565'  28 12            1116            JR      Z,EXCHXT        ; DONE
0567'  F5               1117            PUSH    AF              ; REMAINING LINE COUNT
0568'  01 0020          1118            LD      BC,32           ; ADJUST ADDRESSES TO START
056B'  09               1119            ADD     HL,BC           ; OF NEXT LINE
056C'  3E 78            1120            LD      A,78H
056E'  A4               1121            AND     H
056F'  67               1122            LD      H,A
0570'  EB               1123            EX      DE,HL
0571'  09               1124            ADD     HL,BC
0572'  3E 78            1125            LD      A,78H
0574'  A4               1126            AND     H
0575'  67               1127            LD      H,A
0576'  EB               1128            EX      DE,HL
0577'  18 06            1129            JR      EXCH0           ; DO NEXT LINE
0579'  F1               1130    EXCHXT  POP     AF              ; A=ORIG. LINE COUNT
057A'  6F               1131            LD      L,A             ; LINE COUNT
057B'  26 00            1132            LD      H,0
057D'  29               1133            ADD     HL,HL           ; # OF ATTRIBUTE BYTES TO EXCHANGE
057E'  29               1134            ADD     HL,HL
057F'  29               1135            ADD     HL,HL
0580'  29               1136            ADD     HL,HL
0581'  29               1137            ADD     HL,HL           ; (32*NO. OF LINES)
0582'  44               1138            LD      B,H
0583'  4D               1139            LD      C,L             ; COUNT TO BC
0584'  E1               1140            POP     HL              ; SOURCE 1 ADRS.
0585'  CD 0692'         1141            CALL    CALCATT         ; GET ADRS. OF ATTRIBUTE FILE
0588'  EB               1142            EX      DE,HL           ; SOURCE 1 ATTRIBUTE ADRS. IN DE
0589'  E1               1143            POP     HL              ; SOURCE 2 ADRS.
058A'  CD 0692'         1144            CALL    CALCATT         ; SOURCE 2 ATTRIBUTE ADRS. IN HL
058D'  CD 0593'         1145            CALL    EXCHLP          ; EXCHANGE ATTRIBUTE BYTES
0590'  C3 008F'         1146            JP      GOODRET         ; RETURN B=0
                        1147    ;
                        1148    ;
                        1149    ;                               ; EXCHANGE LOOP
                        1150    ;
                        1151    ;                               ; EXCHANGES DATA BETWEEN TWO
                        1152    ;                               ; MEMORY AREAS.  ADDRESSES IN
                        1153    ;                               ; DE AND HL, BYTE COUNT IN BC
                        1154    ;
0593'  C5               1155    EXCHLP  PUSH    BC              ; SAVE COUNT
0594'  7E               1156            LD      A,(HL)          ; DATA FROM SOURCE 2
0595'  4F               1157            LD      C,A             ; TO WKG.REG.
0596'  1A               1158            LD      A,(DE)          ; DATA FROM SOURCE 1
0597'  A9               1159            XOR     C               ; EXCHANGE VIA XOR
0598'  F5               1160            PUSH    AF              ; SAVE INTERMEDIATE RESULT
0599'  A9               1161            XOR     C               ; A NOW CONTAINS DATA FOR SOURCE 2
059A'  4F               1162            LD      C,A             ; C=NEW DATA FOR SOURCE 2
059B'  F1               1163            POP     AF              ; INTERMEDIATE RESULT
059C'  A9               1164            XOR     C               ; A=NEW DATA FOR SOURCE 1
059D'  12               1165            LD      (DE),A          ; WRITE NEW DATA
059E'  71               1166            LD      (HL),C
059F'  13               1167            INC     DE              ; ADJUST TO NEXT BYTE
05A0'  23               1168            INC     HL
05A1'  C1               1169            POP     BC              ;
05A2'  0B               1170            DEC     BC              ; DECREMENT COUNT
05A3'  78               1171            LD      A,B             ; TEST IF DONE
05A4'  B1               1172            OR      C
05A5'  20 EC            1173            JR      NZ,EXCHLP
05A7'  C9               1174            RET
                        1175
                        1176    ;
                        1177    ;                               ; SUB-RTN. TO GET
                        1178    ;                               ; DP ADRS. OF LINE IN B (0-23)
                        1179    ;                               ; FOR DP IN D (1 OR 2)
                        1180    ;                               ; RETURNS CARRY IF INVALID PARAMETER
05A8'  78               1181    GTADRS  LD      A,B
05A9'  FE 18            1182            CP      24              ; TEST VALID LINE NO.
05AB'  30 1E            1183            JR      NC,GAERR
05AD'  0F               1184            RRCA
05AE'  0F               1185            RRCA
05AF'  0F               1186            RRCA
0580'  E6 E0            1187            AND     0E0H
0582'  6F               1188            LD      L,A
0583'  78               1189            LD      A,B
0584'  E6 18            1190            AND     18H
0586'  F6 40            1191            OR      40H
0588'  67               1192            LD      H,A
0589'  7A               1193            LD      A,D
058A'  A7               1194            AND     A
058B'  28 08            1195            JR      Z,GAERR         ; ERROR IF NOT 1 OR 2
058D'  FE 03            1196            CP      3
058F'  30 0A            1197            JR      NC,GAERR
05C1'  CB 47            1198            BIT     0,A             ; TEST WHICH DP
05C3'  20 04            1199            JR      NZ,GTADR1       ; DP1
05C5'  7C               1200            LD      A,H
05C6'  F6 20            1201            OR      20H             ; DP2
05C8'  67               1202            LD      H,A
05C9'  A7               1203    GTADR1  AND     A               ; RETURN NO CARRY
05CA'  C9               1204            RET
                        1205    ;
05CB'  37               1206    GAERR   SCF                     ; SET CARRY
05CC'  C9               1207            RET
                        1208    ;
```

```
                        1210            SUBTTL  INTERNAL SUB-RTNS.
                        1211    ;
                        1212    ;
                        1213    ;
                        1214                                    ; INPUT:  VIDEO MODE IN A
                        1215                                    ; OUTPUT: M/W SETTING IN B
                        1216                                    ;         VIDMOD SETTING IN C
                        1217                                    ; RETURNS NZ STATUS IF MODE INVALID
                        1218                                    ;
                        1219                                    ;  VALID VALUES:  INPUT     M/W      VIDMOD
                        1220                                    ;                 -----     ---      ------
                        1221                                    ;   *=DISPLAY FILE ACTIVE AT SCREEN
                        1222                                    ;
                        1223                                    ;    DF1 ONLY       0        0         0
                        1224                                    ;   DF1*& DF2       80       0         90
                        1225                                    ;   DF1 & DF2*      1        1         91
                        1226                                    ;  HIGH RES.GR.     2        2         2
                        1227                                    ;  64/80-COLUMN  06 - 3E  06 - 3E   46 - 7E
05CD'  4F               1228    GETVAL  LD      C,A
05CE'  CB 47            1229            BIT     0,A
05D0'  20 16            1230            JR      NZ,TSTD1
05D2'  FE 80            1231            CP      80H
05D4'  28 1A            1232            JR      Z,SET0
05D6'  CB 7F            1233            BIT     7,A
05D8'  C8               1234            RET     NZ
05D9'  47               1235            LD      B,A
05DA'  FE 02            1236            CP      2
05DC'  C8               1237            RET     Z
05DD'  E6 C6            1238            AND     0C6H
05DF'  EE 06            1239            XOR     6
05E1'  C8               1240            RET     NZ
05E2'  3E 40            1241            LD      A,40H
05E4'  B1               1242            OR      C
05E5'  4F               1243            LD      C,A
05E6'  AF               1244            XOR     A
05E7'  C9               1245            RET
05E8'  FE 01            1246    TSTD1   CP      1
05EA'  C0               1247            RET     NZ
05EB'  06 01            1248            LD      B,1
05ED'  0E 81            1249            LD      C,81H
05EF'  C9               1250            RET
05F0'  AF               1251    SET0    XOR     A
05F1'  47               1252            LD      B,A
05F2'  C9               1253            RET
                        1254    ;
                        1255    ;
                        1256                                    ; ROUTINES TO ENABLE CHUNK 0 IN ROM EXT.
                        1257                                    ; FOR ACCESS TO CHNGVID ROUTINE
05F3'  F5               1258    ENBLEXT PUSH    AF
05F4'  DB FF            1259            IN      A,(MREXPT)
05F6'  F6 80            1260            OR      80H             ; SELECT ROM EXT.
05F8'  D3 FF            1261            OUT     (MREXPT),A
05FA'  3E 03            1262            LD      A,3
05FC'  D3 F4            1263            OUT     (DRMSPT),A
05FE'  F1               1264            POP     AF
05FF'  C9               1265            RET
                        1266    ;
0600'  F5               1267    ENBLMDME  PUSH  AF
0601'  AF               1268              XOR   A
0602'  D3 F4            1269              OUT   (DRMSPT),A
0604'  DB FF            1270              IN    A,(MREXPT)
0606'  E6 3F            1271              AND   03FH
0608'  D3 FF            1272              OUT   (MREXPT),A
060A'  F1               1273              POP   AF
060B'  C9               1274              RET
                        1275    ;
                        1276                                    ; TEST PARAMETER VALIDITY
                        1277                                    ; IF NOT VALID, DISCARDS RETURN
                        1278                                    ; AND EXITS VIA INVPAR
060C'  3E 17            1279    TSTPAR  LD      A,23            ; TEST LINE >23
060E'  B8               1280            CP      B
060F'  38 04            1281            JR      NC,TSTPR1
0611'  F1               1282    PARERR  POP     AF
0612'  C3 00C4'         1283            JP      INVPAR
0615'  3A 0033'         1284    TSTPR1  LD      A,(CLINLEN)
0618'  3D               1285            DEC     A
0619'  B9               1286            CP      C               ; TEST COL. >31
061A'  38 F5            1287            JR      C,PARERR
061C'  C9               1288            RET
                        1289    ;                               ; CONVERT LINE/COL. FROM USER TO
                        1290                                    ; INTERNAL FORMAT AND VICE VERSA
                        1291                                    ; USER FORMAT:  LINES 0-23
                        1292    ;                                               COL.0-31
                        1293                                    ; INTERNAL FORMAT: LINES 24-1
                        1294    ;                                                  COL.  33-2
                        1295    ;                                                (COL.1 END OF LINE)
061D'  3A 0033'         1296    CONVPM  LD      A,(CLINLEN)
0620'  3C               1297            INC     A
0621'  91               1298            SUB     C
0622'  4F               1299            LD      C,A
0623'  3E 18            1300            LD      A,SCRSZ
0625'  90               1301            SUB     B
0626'  47               1302            LD      B,A
0627'  C9               1303            RET
                        1304    ;
                        1305    ;
                        1306    ;
0628'                   1307    UPDPOSN                         ; ENTER HERE WITH BC=LINE/COL.IN
                        1308                                    ; INTERNAL FORMAT
0628'  CD 062D'         1309            CALL    CALCPOS         ; CALCULATE POSITION
062B'  18 26            1310            JR      STPOSN          ; STORE UPDATED POSITION AND RETURN
                        1311    ;
                        1312    ;
                        1313    ;                               ; ROUTINE TO CALCULATE POSITION IN DF
                        1314                                    ; RETURNS DF ADRS. IN HL.  PRESERVES
                        1315                                    ; LINE/COLUMN POSITION IN BC
                        1316    ;
062D'  CD 063A'         1317    CALCPOS CALL    LMBU            ; GET DISPLAY FILE ADRS.
0630'  3A 0033'         1318            LD      A,(CLINLEN)     ; ADD COLUMN OFFSET
0633'  3C               1319            INC     A
0634'  91               1320            SUB     C
0635'  5F               1321            LD      E,A
0636'  16 00            1322            LD      D,0
0638'  19               1323            ADD     HL,DE
0639'  C9               1324            RET
```

```
                        1325  ;
                        1326  ;
063A'                   1327  LNBUF:                        ; GET DISPLAY FILE ADDRESS
                        1328                                ; FOR START OF LINE IN B
063A' 3E 18             1329          LD   A,SCRSZ
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 5CC2           1341          LD   A,(VIDMOD)
064D' CB 47             1342          BIT  0,A                ; TEST IF DP1
064F' C8                1343          RET  Z
0650' 3E 20             1344          LD   A,20H              ; SET TO DP2
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   (DPADRS),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,(DPADRS)
0664' C9                1357          RET
                        1358  ;
                        1359  ;
                        1360  ;
                        1361  ;
0665'                   1362  UPDATT:                       ; UPDATE ATTRIBUTE BYTE FOR CHARACTER
                        1363                                ; JUST PRINTED
                        1364                                ; HL = ANY SCAN OF CHAR.IN DP
                        1365  ;
0665' CD 0692'          1366          CALL    CALCATT       ; ADRS. OF ATTRIBUTE BYTE TO HL
0668' 3A 0039'          1367          LD      A,(ATTBYT)    ;
066B' 5F                1368          LD      E,A           ; CURRENT ATTR.VALUE TO E
066C' 3A 003F'          1369          LD      A,(ATTMSK)    ;
066F' 57                1370          LD      D,A           ; MASK TO D
0670' 3A 0038'          1371          LD      A,(MASKB)
0673' 47                1372          LD      B,A           ; FLAGS TO B
0674' 7E                1373          LD      A,(HL)        ; BYTE FROM ATTRIBUTE FILE
0675' AB                1374          XOR     E
0676' A2                1375          AND     D
0677' AB                1376          XOR     E             ; NEW ATTR.FOR 0: OLD FOR 1
0678' CB 60             1377          BIT     4,B           ; SET INK COMPLEMENT OF PAPER?
067A' 28 09             1378          JR      Z,UPDAT1      ;
067C' E6 F8             1379          AND     0F8H          ; SET INK TO BLACK
067E' CB 6F             1380          BIT     5,A           ; IS PAPER 4-7
0680' 20 02             1381          JR      NZ,UPDAT1     ; 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 INK?
0686' 28 08             1384          JR      Z,UPDAT2      ;
0688' E6 C7             1385          AND     0C7H          ; SET PAPER TO BLACK
068A' CB 57             1386          BIT     2,A           ; IS INK 4-7
068C' 20 02             1387          JR      NZ,UPDAT2     ; 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 ADRS. OF
                        1393  ;                             ; ATTRIBUTE BYTE FROM ADRS. OF
                        1394  ;                             ; ANY SCAN ROW IN DP
                        1395  ;
0692' 7C                1396  CALCATT LD      A,H           ; UPPER BYTE OF ADDRESS
0693' 0F                1397          RRCA
0694' 0F                1398          RRCA
0695' 0F                1399          RRCA
0696' E6 03             1400          AND     03H
0698' F6 58             1401          OR      58H
069A' CB 6C             1402          BIT     5,H           ; TEST WHICH DP
069C' 28 02             1403          JR      Z,CALCA1
069E' F6 20             1404          OR      20H           ; DP2
06A0' 67                1405  CALCA1  LD      H,A           ; ADRS. OF ATTRIBUTE IN DP2
06A1' C9                1406          RET
                        1407  ;
06A2'                   1408  LOATTR                        ; LOAD INTERNAL ATTRIBUTE VARIABLES
                        1409                                ; FROM SYSTEM VARIABLES
06A2' F5                1410          PUSH    AF            ; SAVE A
06A3' 3A 5C8D           1411          LD      A,(ATTR_P)    ;
06A6' 32 0039'          1412          LD      (ATTBYT),A    ;
06A9' 3A 5C8E           1413          LD      A,(MASK_P)    ;
06AC' 32 003F'          1414          LD      (ATTMSK),A    ;
06AF' 3A 5C91           1415          LD      A,(P_FLAG)    ;
06B2' 0F                1416          RRCA                  ; SHIFT ODD BITS TO EVEN
06B3' 32 0038'          1417          LD      (MASKB),A     ;
06B6' F1                1418          POP     AF
06B7' C9                1419          RET
                        1420  ;
06B8' 3A 5CC3           1421  GETSTRG LD    A,(PARAMS)      ; GET STRING ID
06BB' E6 1F             1422          AND   1FH             ; MASK OFF UPPER BITS
06BD' F6 40             1423          OR    40H             ; STRING VARIABLE IDENTIFIER
06BF' 57                1424          LD    D,A             ; SAVE IN D
06C0' 2A 5C4B           1425          LD    HL,(VARS)       ; FIND STRING
06C3' 7E                1426  GTSTR1  LD    A,(HL)          ;
06C4' E6 7F             1427          AND   07FH            ; TEST IF END OF VARS AREA
06C6' 28 13             1428          JR    Z,NOSTRG        ; NO FIND - RETURN BC=2
06C8' BA                1429          CP    D               ; TEST IF MATCH
06C9' 28 08             1430          JR    Z,GTSTR2        ; FOUND STRING
06CB' D5                1431          PUSH  DE              ; SAVE STRING ID
06CC' CD 1720           1432          CALL  RECLEN          ; RETURNS ADRS. OF NEXT VAR. IN DE
06CF' EB                1433          EX    DE,HL           ; ADRS. TO HL
06D0' D1                1434          POP   DE              ; RESTORE STRING ID
06D1' 18 F0             1435          JR    GTSTR1          ; LOOK AGAIN
06D3' 23                1436  GTSTR2  INC   HL              ; GET LENGTH
06D4' 4E                1437          LD    C,(HL)
```

```
06D5' 23              1438          INC  HL
06D6' 46              1439          LD   B,(HL)
06D7' 23              1440          INC  HL                 ; FIRST TEXT CODE
06D8' 78              1441          LD   A,B                ; TEST IF NULL STRING
06D9' B1              1442          OR   C
06DA' C9              1443          RET                     ; RETURN (Z IF NULL STRING/NZ IF NOT)
                      1444  ;
06DB' 0E 02           1445  NOSTRG  LD   C,2
06DD' 06 00           1446          LD   B,0
06DF' C9              1447          RET                     ; (Z AND BC=2 FOR NO FIND)
                      1448  ;
                      1449  ;
                      1451          SUBTTL GRAPHICS CHAR.SET
                      1452  ;
                      1453  ;                               ;PIXEL PATTERNS FOR STD.GRAPHICS SET
                      1454  ;                               ; 0=BACKGROUND(PAPER) 1=FOREGROUND(INK)
                      1455  ;
06E0' 00              1456  GRPST   defb 00000000b         ; code 80       graphics space
06E1' 00              1457          defb 00000000b
06E2' 00              1458          defb 00000000b
06E3' 00              1459          defb 00000000b
06E4' 00              1460          defb 00000000b
06E5' 00              1461          defb 00000000b
06E6' 00              1462          defb 00000000b
06E7' 00              1463          defb 00000000b
                      1464  ;
06E8' 0F              1465          defb 00001111b          ; code 81       graphics
06E9' 0F              1466          defb 00001111b
06EA' 0F              1467          defb 00001111b
06EB' 0F              1468          defb 00001111b
06EC' 00              1469          defb 00000000b
06ED' 00              1470          defb 00000000b
06EE' 00              1471          defb 00000000b
06EF' 00              1472          defb 00000000b
                      1473  ;
                      1474
06F0' F0              1475          defb 11110000b          ; code 82       graphics
06F1' F0              1476          defb 11110000b
06F2' F0              1477          defb 11110000b
06F3' F0              1478          defb 11110000b
06F4' 00              1479          defb 00000000b
06F5' 00              1480          defb 00000000b
06F6' 00              1481          defb 00000000b
06F7' 00              1482          defb 00000000b
                      1483
06F8' FF              1484          defb 11111111b          ; code 83       graphics
06F9' FF              1485          defb 11111111b
06FA' FF              1486          defb 11111111b
06FB' FF              1487          defb 11111111b
06FC' 00              1488          defb 00000000b
06FD' 00              1489          defb 00000000b
06FE' 00              1490          defb 00000000b
06FF' 00              1491          defb 00000000b
                      1492
                      1493  ;
0700' 00              1494          defb 00000000b          ; code 84       graphics
0701' 00              1495          defb 00003300b
0702' 00              1496          defb 00003300b
0703' 00              1497          defb 00003300b
0704' 0F              1498          defb 00001111b
0705' 0F              1499          defb 00001111b
0706' 0F              1500          defb 00001111b
0707' 0F              1501          defb 00001111b
                      1502  ;
0708' 0F              1503          defb 00001111b          ; code 85       graphics
0709' 0F              1504          defb 00001111b
070A' 0F              1505          defb 00001111b
070B' 0F              1506          defb 00001111b
070C' 0F              1507          defb 00001111b
070D' 0F              1508          defb 00001111b
070E' 0F              1509          defb 00001111b
070F' 0F              1510          defb 00001111b
                      1511
                      1512
0710' F0              1513          defb 11110000b          ; code 86       graphics
0711' F0              1514          defb 11110000b
0712' F0              1515          defb 11110000b
0713' F0              1516          defb 11110000b
0714' 0F              1517          defb 00001111b
0715' 0F              1518          defb 00001111b
0716' 0F              1519          defb 00001111b
0717' 0F              1520          defb 00001111b
                      1521  ;
0718' FF              1522          defb 11111111b          ;code 87        graphics
0719' FF              1523          defb 11111111b
071A' FF              1524          defb 11111111b
071B' FF              1525          defb 11111111b
071C' 0F              1526          defb 00001111b
071D' 0F              1527          defb 00001111b
071E' 0F              1528          defb 00001111b
071F' 0F              1529          defb 00001111b
                      1530  ;
0720' 00              1531          defb 00003300b          ; code 88       graphics
0721' 00              1532          defb 00003300b
0722' 00              1533          defb 00003300b
0723' 00              1534          defb 00003300b
0724' F0              1535          defb 11110000b
0725' F0              1536          defb 11110000b
0726' F0              1537          defb 11110000b
0727' F0              1538          defb 11110000b
                      1539  ;
0728' 0F              1540          defb 00001111b          ; code 89       graphics
0729' 0F              1541          defb 00001111b
072A' 0F              1542          defb 00001111b
072B' 0F              1543          defb 00001111b
072C' F0              1544          defb 11110000b
072D' F0              1545          defb 11110000b
072E' F0              1546          defb 11110000b
072F' F0              1547          defb 11110000b
```

```
                                1548    ;
0730'  F0                       1549            defb 11110000b          ; code 8A       graphics
0731'  F0                       1550            defb 11110000b
0732'  F0                       1551            defb 11110000b
0733'  F0                       1552            defb 11110000b
0734'  F0                       1553            defb 11110000b
0735'  F0                       1554            defb 11110000b
0736'  F0                       1555            defb 11110000b
0737'  F0                       1556            defb 11110000b
                                1557    ;
0738'  FF                       1558            defb 11111111b          ; code 8B       graphics
0739'  FF                       1559            defb 11111111b
073A'  FF                       1560            defb 11111111b
073B'  FF                       1561            defb 11111111b
073C'  F0                       1562            defb 11110000b
073D'  F0                       1563            defb 11110000b
073E'  F0                       1564            defb 11110000b
073F'  F0                       1565            defb 11110000b
                                1566    ;
0740'  00                       1567            defb 00000000b          ; code 8C       graphics
0741'  00                       1568            defb 00000000b
0742'  00                       1569            defb 00000000b
0743'  00                       1570            defb 00000000b
0744'  FF                       1571            defb 11111111b
0745'  FF                       1572            defb 11111111b
0746'  FF                       1573            defb 11111111b
0747'  FF                       1574            defb 11111111b
                                1575    ;
0748'  0F                       1576            defb 00001111b          ; code 8D       graphics
0749'  0F                       1577            defb 00001111b
074A'  0F                       1578            defb 00001111b
074B'  0F                       1579            defb 00001111b
074C'  FF                       1580            defb 11111111b
074D'  FF                       1581            defb 11111111b
074E'  FF                       1582            defb 11111111b
074F'  FF                       1583            defb 11111111b
                                1584    ;
0750'  F0                       1585            defb 11110000b          ;code 8E        graphics
0751'  F0                       1586            defb 11110000b
0752'  F0                       1587            defb 11110000b
0753'  F0                       1588            defb 11110000b
0754'  FF                       1589            defb 11111111b
0755'  FF                       1590            defb 11111111b
0756'  FF                       1591            defb 11111111b
0757'  FF                       1592            defb 11111111b
                                1593    ;
0758'  FF                       1594            defb 11111111b          ;code 8F        graphics
0759'  FF                       1595            defb 11111111b
075A'  FF                       1596            defb 11111111b
075B'  FF                       1597            defb 11111111b
075C'  FF                       1598            defb 11111111b
075D'  FF                       1599            defb 11111111b
075E'  FF                       1600            defb 11111111b
075F'  FF                       1601            defb 11111111b
                                1602
                                1603            END
```

```
ADL - ASC 004 DUAL SCREEN MODE SUPPORT  CRZ80/11 version 10.36.14       16-Mar-84  12:47:30
GRAPHICS CHAR.SET   IGDS.SRC

A       Reserved        ATTBYT  0039            ATTMSK  00BF            ATTRBP  5C8D            B       Reserved
BOTLN   0020            C       Reserved        CALCAT  0692            CALCA1  06A0            CALCP0  0620
CHNGVI  0E8E            CMPSET  3C00            CNTBL   002F            CRSCN   0108 IN         CLINIT  1800
CLRCTL  000C            CLRS80  01D4            CLRSCB  C00E IN         CLRSCN  0108 IN         CLRSC0  01EE
CLRSC1  0202            CLRSC2  0200            CLRSC3  0211            CLRSC4  0214            CLRSC5  021F
CLRSC7  0254            CLRSXT  0250            CONVFM  9610            COORDS  5C7D            COPERR  0439
COPER1  0436            COPYS0  0040 IN         COPYSC  0430 IN         COPYS5  0408            COPYS6  0489
COPYS2  0477            COPYS3  0478            COPYS4  047E            COPY01  03EC            COPY41  040D
COPYS7  04F0            COPYWK  0388            COPYXT  04C2            DATAB   0000            DEST7   FFC0
COPY42  04AF            CURPOS  0034            D       Reserved        DRIVES  6040            DTADP1  03CB            DTADP2  03D2
DPADR1  0036            DXMSPT  00F4            E       Reserved        ENBLMD  0600            ERRRET  00C1            EXCMLP  0593
E       Reserved        ENBLEX  05F3            EXCMS0  04F3            EXCNXT  0570            EXCMD   054F
EXCMSB  0043 IN         EXCMSC  0520 IN         EXTNRM  0385            FIX     97C0            FIXTBL  1A00
EXCM01  0503            EXCM1   0551            GAERR   05CB            GETATB  02DC            GETATB  001E IN
FPIA    3193            FRFV    038A            GETCTL  0019            GETCUB  0021 IN         GETCUR  02F1 IN
GETATT  0280 IN         GETC80  02F1            GETN1   0428            GETN2   042F            GETSTR  0688
GETC1   0309            GETND   0418            GRBLK   01A9            GPPMST  05E0            GRPST   06E0
GETVAL  05C9            GGOORB  008F            GTADR1  05C9            GTADR2  05E6            GTCHAR  0260 IN         GTCMB0  025C
GRTBL   0031            GTADRS  05A8            GTCHN2  0276            GTCHN3  0288            GTCM3   028D
GTCHR8  0018 IN         GTCM1   0271            GTCM4   02AF            GTCM5   02C4            GTCM6   02D9
GTCM31  0290            GTCM32  02A8            GTSTR2  06D3            H       Reserved        HREXPT  00FF
GTINDX  003A            GTSTR1  06C3            INSERT  12C0            INVPAR  05C4            L       Reserved
INTNT   30F9            INSDSL  013F            LINCOL  0007            LINLEN  0033            LNBU    063A
LDATTR  06A2            LOPCSN  0650            LOOP1   016E            M       Reserved        MASKB   0038
LOOP    0163            LOOP0   0166            MOSTPG  6408            PARAM8  5CC3            PARERR  0611
MASK8P  5C8E            MOVBSZ  0840            P8FLAG  5D91            RAMTOP  5C82            RECLEN  1720
PRNT    5CB4            PSW     Reserved        SCRL8   002A IN         SCROLL  0128 IN         SCRLCT  002E
SCINIT  1701            SCRCTL  0028            SCRLD   0141            SETCB0  0117            SETCUB  0009 IN
SCRLXT  0186            SCRLD   0141            SETCUR  0115 IN         SETM00  032A            SETM01  0334
SETM02  038A            SETM03  039F            SETM0B  0325 IN         SETM21  0397            SETO    05F0
SETMOD  0311 IN         SPARE0  0011            SPARE1  0013            SPARE2  0015            SPARE3  0017
SP      Reserved        SPARE4  0030            STBLK0  0185            STBLK1  0186            STKEND  5C65
STBLK   0181            STPOSN  0655            STPGCT  9038            TESTAB  0151            TSTPAR  060C
STM080  030E            TSTP1   0615            TSTO1   05E8            TVFULG  00C8            TVFUL1  00E4            UDG     5C78
TSTPR1  0615            UPDATB  03A2            UPDATT  0665            UPDAT1  06B4            UPDAT2  0690            UP3POS  0628
UPDATE  03A2            VARS    5C48            VICMOD  5CC2            VIMODE  5324            WRCHAR  0049 IN         WRCHR8  0001 IN
VARS    5C48            WRCHNT  000C            WRCH0   0046            WRCH01  004C            WRCH11  0068            WRCH12  0073
WRCHNT  000C            WRCH13  0077            WRCH14  008D            WRCH15  008E            WRCH2   0096            WRCH3   009F
WRCH13  0077            WRCH5   00A7            WRCH7   00B3            WRSTLP  00F5            WRSTR8  0504 IN         WRSTRG  00F2 IN
WRCH5   00A7            WRSTR0  00EE            WRSTR3  00FF            WRSTX1  0100            WRSTX2  0114

    No errors detected
```

Cross reference listing (MREP version 4.7)

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

```
ATTBYT   1570    597   1367   14120
ATTMSK   1644#   1369   14140
ATTR_P    650   1411
BOTLN    1350    268   7320
CALCA1   1403   1405#
CALCAT    629    546    677   1001   1004   1141   1144
                1366   13960
CALCPO    589    676   1309   1317#
CHNGVI    520    772
CHRSET    490    147    762
CHTBL    1470    201    590   7430
CH_ADD    610    835   8401   8465   8570   862    875#
                8803   1040   10450  10480  10590
CLINIT    480    737
CLRCTL   1040    481   7380
CLRS80    107   4800
CLRSC0    461    493   4970
CLRSC1   5100    567
CLRSC2   5190
CLRSC3   5220    576
CLRSC4   5240    543
CLRSC5   5330
CLRSC7    518   5700
CLRSCB     40   1070
CLRSCN     38   4830
CLRSXT    565   5680
CONVPN    337    588    675    686   12900
COORDS    640   7820
COPER1   8790
COPERR    867    870   8780
COPY01    838   8460
COPY41    945   9470
COPY42    972   9740
COPY50    166   8340
COPY51   9960    189
COPY52   9310
COPY53   9340   1025
COPY54   9360    956
COPY55    916   10080
COPY56  10200   1029
COPY57    927   1318   10260
COPY58     43   1660
COPYSC     43    860   1850
COPYWK   8430   8480   8530   8520   8540    958    859
                10580  10520  10540  10560  1060   1061
COPYXY    962    990#
CWPOS    1920    685   13980  1399
DATAB     960    174   7400
DEST7     750     76
DFADRS   1540  13510   1356
DKWSPT    580  12630   12690
DRIVES    720     73     74
DTADP1   8140
DTADP2    812   9210
ENBLEK    765  12580
ENBLMO    773  12670
ERRRET   2390    263    280    706
EXCH0   10990   1120
EXCH01   1043  10480
EXCH1   11000   1111
EXCHLP   1105   1145  11550   1173
EXCHS0    167  10370
EXCHSB     44    1670
EXCHSC     44  10640
EXCHXT   1116  11300
EXTNRM    759    763    7850
FIX       760
FIXTBL    700
FP2A      550    872
FRPV      710    720   8060
GAERR    1100   1195   1197   12060
GETAB0    110   6710
GETAT      41   1130
GETATT     39   6740
GETC1     680    695   6970
GETC80    110   6830
GETCTL   1120    504    671   7510   7520
GETCUR     41   1160
GETCUR     39   6840
GETN1    8710
GETN2     865   8740
GETNO     847    849    951    953    855   8620   876
                1049   1051   1053   1055   1057
GETSTR    292    837   1042   1421#
GETVAL    766  12280
GOODKE    2580    339    362    494    726    784    819
                822   1006   1146
GRBLK     385   4630
GRPMST     530    150    744
GRPST      50  14560
GRTBL     1500    198    653   7450
GTADR1   1190  12030
GTADRS    892    897   1086   1091  11810
GTCM1    5930    656    663
GTCM2    5960    646
GTCM23    605   6080
GTCM3     601   6120
GTCM31   6140    620
GTCM32    633    635   6370
GTCM4     607    610    619   6410
GTCM5     652   6570
GTCM6     650   6640
GTCMAR     39   5860
GTCMBO    114   5810
GTCMRB     41   1140
GTINDX   1590   5930    603    630    690
```

```
GTSTR1   1426#  1435
GTSTR2   1430   1436#
HBEXPT    57#   1259    12618   1270    12728
INTNT     54#    871
INSDBL   3846
INSERT    73#    754
INVPAR    186    188     262#    395     558     360     487
                 490     492     721     809     881     888
                 898     893     898    1079    1081    1087
                1092    1283
LOATTR    177    300     497    14088
LOPOSN    181   13548
LINCOL    160#   333     976    7498    7585
LINLEN    151#   213     668     982     687    7368     777
                1284    1296    1318
LWBU      371    505    1317    13278
LOOP      389#   449
LOOP0     391#   408
LOOP1     398#
MASKB     155#   227    1371    14178
MASK_P     66#  1413
MOVESZ     74#    75     755
MOSTRG    1428   1448#
PARAMS     71#  1421
PARERR   1282#  1287
PRMT      69#
P_FLAG    67#  1415
RAMTOP    68#   760
RECLEN    93#  1432
SCINIT    47#   729
SCRCTL   1199#   282     348    7388
SCRL0     284    361    3648
SCRL8      40   1229
SCRL88    122   3468
SCRLCT   1408   272     2768    2818    7348
SCRLRT    412   4158    470
SCROLL     38   3558
SCRSZ      46#   265     365     499     788    1388    1329
SET0     1232  12518
SETC80    103   331#
SETCUR     40   1038
SETCUR     38   3358
SETM21    770   772#
SETM00    712    714     716    7228
SETM01   7298
SETM02    724   7648
SETM03   7768    818     824
SETM08     42   1198
SETM00     42   7088
SPARE0   1088
SPARE1   1098
SPARE2   1108
SPARE3   1118
SPARE4   1638
STBLK     377   4318
STBLK0   4348    448
STBLK1   4538
STKEND     62#   757
STM080    118   7048
STPOSN    257   1318   13498
STRGCT   1618   3228    7488    7618
TBST40   3758    414     473
TST01    1238  12468
TSTPAR    336    987     674   12798
TSTPR1   1281  12848
TVPUL1    274   2818
TVPULS    222   2658
UBG        63#   194     659
UPDAT1   1378  1381    13838
UPDAT2   1384  1387    13898
UPDATE   7778
UPDATT    292  13628
UPDPOS    270    338     969     783    13078
VARS      688  1425
VIDMOD     78#   722    7768    886    1361
VIMODE    117#   706
WRCM0      97   1728
WRCM01   1818   384
WRCM11   1938  1988
WRCM12   1918   2018
WRCM13   1978   2088    2028
WRCM14    214   2198
WRCM15    218   2208
WRCM2     229#
WRCM3     238   2328
WRCM5     239#   249
WRCMT     2508
WRCMAR     38   1768
WRCMRB     40    978
WRCMRT   2578
WRSTLP   3818    315
WRSTR0     98   2928
WRSTR3   3098    328
WRSTRB     #0    988
WRSTRG     38   3888
WRSTR1    307   3198
WRSTR2   3258
```

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 loadec at SPRCODE, where SPRCODE = DE000H. 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 "0: 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: SPR$VC = 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 Init$sprites. 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: SPR$VC + 2D0H

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 + 24DH

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 + 268H

BASIC interface:

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

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

BASIC example: "O0,1" causes the interface routine to return TBS 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$ = "HK<direction>"


Description:  This service is used to scroll the entire
screen in the horizontal direction by one column. The
direction of scroll is to the left if direction is less than
0, otherwise the scroll is to the right. The position
property of all sprites is updated by one column to reflect
the effect of the scroll.  If 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.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
CMDINT  CRI80/11 version 10.36.14       13-Feb-84  16:18:38
Command Interpreter module    CMDINT.58C

            =E000                1                   SECT CMDINT, ABS, LOC=0E000H
                                 2                   NAME CMDINT
                                 3                   SUBTTL  Command Interpreter module
                                 4
                                 5
                                 6   ;***************************************************************************
                                 7   ;
                                 8   ; Command interpreter module
                                 9   ;
                                10   ; Inputs: none.
                                11   ;
                                12   ; Outputs: a Parameter Block (see below).
                                13   ;
                                14   ; Description: this module parses a sprites command string and, for
                                15   ;               each recognized command, produces a parameter block
                                16   ;               describing the command.  This parameter block is passed
                                17   ;               to the interface handler module.  To parse a command
                                18   ;               string first it must be found.  The global sprites
                                19   ;               variable NAME contains the name of the string variable
                                20   ;               containing the command string.  The ROM routine "IND_N
                                21   ;               is used to find the variable in the VARS area of memory.
                                22   ;               Note:  symbols in all upper case are external to this
                                23   ;               module and are defined under Imports below.
                                24   ;
                                25   ;               Following is an architecture overview showing the relation
                                26   ;               between the modules of the sprite services component of
                                27   ;               the application development library:
                                28   ;
                                29   ;
                                30   ;                                  command
                                31   ;                    ------------  string   ----------------
                                32   ;                    |  BASIC   |---------->|   Command    |
                                33   ;                    | program  |<----------| Interpreter  |
                                34   ;                    ------------  Status   ----------------
                                35   ;                                    Code      |         A
                                36   ;                                              |         |
                                37   ;                                Parameter Block |         |  Status Code
                                38   ;                                              V         |
                                39   ;                                           ----------------
                                40   ;                                           |     I/P      |
                                41   ;                                           |   Handler    |
                                42   ;                                           ----------------
                                43   ;                                              |         A
                                44   ;                                 Register values |         |  Status Code
                                45   ;                                              |         |
                                46   ;                                              V         |
                                47   ;                    ------------ Reg values  ----------------
                                48   ;                    | assembly |------------->|    Sprite    |
                                49   ;                    | program  |<-------------|   Services   |
                                50   ;                    ------------    Status    ----------------
                                51   ;                                    Code      |         A
                                52   ;                                              |         |
```

```
                        93   ;                          Register Values |         |  Status Code
                        94   ;                                          |         |
                        95   ;                                          V         |
                        96   ;                                    ---------------
                        97   ;                                    |  Low Level  |
                        98   ;                                    | I/O Services|
                        99   ;                                    ---------------
                       100   ;
                       101   ;
                       102   ;               Note: the interface between the I/O Handler module and the
                       103   ;               Sprite Services module is the same as the interface between
                       104   ;               an assembly language program and the Sprite Services module.
                       105   ;
                       106   ;
                       107   ;               Following is the layout in memory of the code and data used
                       108   ;               by the sprite services component of the application
                       109   ;               development library:
                       110   ;
                       111   ;
                       112   ;                         --------------------
                       113   ;                         |                  |
                       114   ;                         --------------------<-- (UDG)
                       115   ;                         |     sprites      |
                       116   ;                         |      code        |
                       117   ;                         --------------------<-- $PRCODE
                       118   ;                         |     sprites      |
                       119   ;                         |     global       |
                       120   ;                         |     variables    |
                       121   ;                         --------------------
                       122   ;                         |     sprites      |
                       123   ;                         |      data        |
                       124   ;                         ---------------------<-- (SPRITES)
                       125   ;                         |                  |
                       126   ;                         |                  |
                       127   ;                         |                  |
                       128   ;                         --------------------<-- (RAMTOP)
                       129   ;                         |                  |
                       130   ;                         |                  |
                       131   ;                         |                  |
                       132   ;                         |------------------|
                       133   ;                         | command string   |
                       134   ;                         |------------------|
                       135   ;                         --------------------<-- (VARS)
                       136   ;
                       137   ;**************************************************************************
                       138
                       139
                       140   ; Imports
                       141
 =$C34                 142   PLA@$           equ     $C34H
 =$C9D                 143   CH_ADD          equ     $C9DH
                       144
 =0000                 145   ERROR           equ     0
 =30F9                 146   ININT           equ     30F9H
 =3160                 147   FP28C           equ     3160H
 =2C70                 148   FIND_N          equ     2C70H
 =3009                 149   DIGIT           equ     3009H
                       150
                       151                   EXTERNAL  I_P_HANDLER
                       152
                       153
                       154   ; Globals
                       155
                       156                   INCLUDE  spritedef
                       157
                       158   ; Global constants
                       159
 =0018                 160   max_rows        equ     24              ;number of rows on the screen
 =0020                 161   max_cols        equ     32              ;number of columns on the screen
 =E000                 162   sprcode         equ     0E000H          ;location of sprites code
 =0010                 163   globallen       equ     29              ;length of global vars
                       164
                       165   ; Global variables
                       166
 =DFFE                 167   sprites         equ     sprcode-2       ;location of sprites data
 =DFFC                 168   name            equ     sprites-2       ;name of command string var.
 =DFFB                 169   gflags          equ     name-1          ;global flags
 =0000                 170   neg             equ     0               ;  bit 0 - parm < 0
 =0001                 171   automap         equ     1               ;  bit 1 - automap mode flag
 =DFFA                 172   max_sprites     equ     gflags-1        ;number of sprites allocated
 =DFF9                 173   screen_ht       equ     max_sprites-1   ;number of lines of screen used by
                       174                                           ;  sprites
 =DFF8                 175   backgcolor      equ     screen_ht-1     ;permanent background color
 =DFF7                 176   command         equ     backgcolor-1    ;number of command being executed
 =DFE8                 177   parms           equ     command-12      ;parameter block
 =DFE3                 178   scratchpad      equ     parms-8         ;workspace area
                       179
                       180   ; Data structures
                       181
                       182   ; parameter block
 =DFE8                 183   svc_code        equ     parms           ;code identifying service
 =DFEC                 184   num_parms       equ     svc_code+1      ;number of parms found
 =DFEC                 185   parm1           equ     num_parms+1     ;first parm
 =DFEF                 186   parm2           equ     parm1+2
 =DFF1                 187   parm3           equ     parm2+2
 =DFF3                 188   parm4           equ     parm3+2
 =DFF5                 189   parm5           equ     parm4+2         ;last possible parm
                       190
                       191   ; sprite data offsets
 =0000                 192   row             equ     0               ;row position
 =0001                 193   col             equ     row+1           ;column position
 =0002                 194   width           equ     col+1           ;sprite width
 =0003                 195   height          equ     width+1         ;sprite height
 =0004                 196   color           equ     height+1        ;sprite color
```

```
                        =0005           157   bitmap          equ     color+1         !definition address
                        =0007           158   sflags          equ     bitmap+2        !local sprite flags
                        =0000           159   alloc           equ     0               !  bit 0 - allocated?
                        =0001           160   overwrite       equ     1               !  bit 1 - sprite overwrote something
                        =0002           161   lhs             equ     2               !  bit 2 - sprite in lower half screen
                                        162
                                        163
                                        164
                                        165   ! Locals
                                        166
                                        167   ! state names for parser
                        =0000           168   s0              equ     0
                        =0001           169   s1              equ     1
                        =0002           170   s2              equ     2
                        =0003           171   s_end           equ     3
                        =00FF           172   s_error         equ     0FFH
                                        173
                                        174
                                        175   !**************************************************************************
                                        176   !
                                        177   ! CMDINT
                                        178   !
                                        179   ! Inputs: none.
                                        180   !
                                        181   ! Outputs: parameter block.
                                        182   !
                                        183   ! Globals Read: none.
                                        184   !
                                        185   ! Globals Written: none
                                        186   !                  command
                                        187   !                  sflags
                                        188   !                  parms
                                        189   !
                                        190   ! Procedures Called: find_string
                                        191   !                    u_or_l_alpha
                                        192   !                    ERROR
                                        193   !                    letter
                                        194   !                    minus
                                        195   !                    digit
                                        196   !                    number
                                        197   !                    comma
                                        198   !                    space
                                        199   !                    I_P_HANDLER
                                        200   !
                                        201   ! Procedures Called By: none.
                                        202   !
                                        203   ! Description: this is the main routine of the command interpreter.
                                        204   !              The command string is located (using FIND_N) and is
                                        205   !              recognized by the following automaton:
                                        206   !
                                        207   !                                        comma
                                        208   !                 blank          minus----------
                                        209   !                 -----          -----|        |
                                        210   !                 |   |          |   |v       |
                                        211   !                 --> s0 ----------> s1 ----> s2 ------------> s_end
                                        212   !                     |  bad letter  |  uint  |  space or eos
                                        213   !                     |              |        |
                                        214   !                     |              v        |
                                        215   !                     ----------> s_error <-----
                                        216   !
                                        217   !              where uint represents an unsigned integer and eos
                                        218   !              represents the end of the string.
                                        219   !
                                        220   !**************************************************************************
                                        221
                                        222
                        =E000           223                   ORG     cprcode
                                        224
                                        225
0000  3E 24                             226   cmdint          ld      a, '$'          !put a "$" in the 2'nd byte of name
0002  32 DFFD                           227                   ld      (name+1), a
0005  AF                               228                   xor     a               !initialize command variable
0006  32 DFF7                           229                   ld      (command), a
0009  21 DFF8                           230                   ld      hl, sflags
000C  CB 86                             231                   res     neg, (hl)       !assume positive number until
                                        232                                           !  shown otherwise
000E  CD E092                           233                   call    find_string     !hl -> command string record
0011  23                                234                   inc     hl              !hl -> string length
0012  5E                              235                   ld      e, (hl)
0013  23                              236                   inc     hl
0014  56                              237                   ld      d, (hl)         !de = string length
0015  23                              238                   inc     hl
0016  E5                              239                   push    hl
0017  19                              240                   add     hl, de
0018  EB                              241                   ex      de, hl          !de -> first byte after string
0019  E1                              242                   pop     hl              !hl -> first byte of string
001A  06 00                           243                   ld      b, s0           !state <- s0
001C  0E 00                           244   while0          ld      c, 0            !parms <- 0
001E  DD 21 DFED                      245                   ld      ix, parm1       !ix -> current parm
0022  3E 00                           246                   ld      a, s0
0024  B8                              247                   cp      b
0025  20 66                           248                   jr      nz, ends0       !state <> s0, so exit outer loop
0027  CB 78                           249   while1          bit     7, b
0029  28 02                           250                   jr      z, ok           !state <> s_error, so continue
002B  CF                              251                   rst     ERROR           !otherwise produce report "4"
002C  09                              252                   defb    9
002D  3E 03                           253   ok              ld      a, s_end
002F  B8                              254                   cp      b
0030  28 38                           255                   jr      z, ends1        !state = s_end, so exit inner loop
0032  E5                              256                   push    hl
0033  A7                              257                   and     a
0034  ED 52                          258                   sbc     hl, de
0036  E1                              259                   pop     hl
0037  28 30                           260                   jr      z, do_cmd       !at end of command string, so execute
```

```
8039  7E              261                   ld      a, (hl)         ;get next char
803A  CC 8089         262                   call    u_or_l_alpha
803D  30 05           263                   jr      nc, not_alpha   ;jump if not upper or lower case letter
803F  CD 80C8         264                   call    letter          ;otherwise handle letter
8042  18 E3           265                   jr      while1
8044  FE 2D           266   not_alpha       cp      '-'
8046  20 05           267                   jr      nz, not_minus   ;jump if not minus sign
8048  CC E10D         268                   call    minus           ;otherwise handle minus
804B  18 D4           269                   jr      while1
804D  CD 80D9         270   not_minus       call    DIGIT
8050  38 05           271                   jr      c, not_digit    ;jump if not digit
8052  CD 8121         272                   call    number          ;otherwise handle number
8055  18 D0           273                   jr      while1
8057  FE 2C           274   not_digit       cp      ','
8059  20 05           275                   jr      nz, not_comma   ;jump if not comma
805B  CD E150         276                   call    comma           ;otherwise handle comma
805E  18 C7           277                   jr      while1
8060  FE 20           278   not_comma       cp      ' '
8062  20 05           279                   jr      nz, bad_char    ;not a space either, so must be illegal
8064  CD E164         280                   call    space           ;handle space
8067  18 BE           281                   jr      while1
8069  06 FF           282   bad_char        ld      b, s_error      ;state <- s_error
806B  18 0A           283                   jr      while1
806D  E5              284   endw1           push    hl              ;here when exited inner loop
806E  A7              285                   and     a
806F  ED 52           286                   sbc     hl, de
8071  E1              287                   pop     hl
8072  28 02           288                   jr      z, do_cmd       ;at end of string, so execute
8074  06 00           289                   ld      b, s0           ;else more string left
8076  79              290   do_cmd          ld      a, c            ;fill in num_parms slot of parm block
8077  32 0FEC         291                   ld      (num_parms), a
807A  C5              292                   push    bc
807B  D5              293                   push    de
807C  E5              294                   push    hl
807D  CD 0001e        295                   call    I_P_HANDLER     ;invoke interface handler module
8080  C5              296                   push    bc              ;save status code in bc'
8081  D9              297                   exx
8082  C1              298                   pop     bc
8083  D9              299                   exx
8084  21 0FF7         300                   ld      hl, command     ;increment command counter
8087  34              301                   inc     (hl)
8088  E1              302                   pop     hl
8089  D1              303                   pop     de
808A  C1              304                   pop     bc
808B  18 BF           305                   jr      whileO
808D  D9              306   endw0           exx                     ;done with command string, so return
808E  C5              307                   push    bc              ;  last status code in bc
808F  D9              308                   exx
8090  C1              309                   pop     bc
8091  C9              310                   ret
                      311
                      312
                      313   ;*****************************************************************************
                      314   ;
                      315   ; FIND_STRING
                      316   ;
                      317   ; Inputs: name (see global variables).
                      318   ;
                      319   ; Outputs: hl -> record for string variable if found
                      320   ;
                      321   ; Globals Read: CM_ADD
                      322   ;               FLAGS
                      323   ;               name
                      324   ;
                      325   ; Globals Written: CM_ADD
                      326   ;                  FLAGS
                      327   ;
                      328   ; Procedures Called: FIND_N
                      329   ;                    ERROR
                      330   ;
                      331   ; Procedures Called By: cmdint
                      332   ;
                      333   ; Description: this routines attempts to find the record for a string
                      334   ;              variable with name equal to the value of NAME.  If found,
                      335   ;              HL is set to point to the length field of the record.  If
                      336   ;              the variable is not found, report "C" is produced.
                      337   ;
                      338   ;*****************************************************************************
                      339
                      340
8092  ED 48 5C5D      341   find_string     ld      bc, (CM_ADD)    ;save CM_ADD
8096  C5              342                   push    bc
8097  3A 5C3A         343                   ld      a, (FLAGS)      ;save FLAGS
809A  F5              344                   push    af
809B  01 0FFC         345                   ld      bc, name
809E  ED 43 5C5D      346                   ld      (CM_ADD), bc    ;set CM_ADD to point to name
80A2  3E 80           347                   ld      a, 80H
80A4  32 5C3A         348                   ld      (FLAGS), a      ;set runtime flag, reset others
80A7  CD 2C70         349                   call    FIND_N          ;find variable
80AA  30 02           350                   jr      nc, f_ok        ;jump if found variable
80AC  CF              351                   rst     ERROR           ;else produce report "C"
80AD  08              352                   defb    08H
80AE  C1              353   f_ok            pop     bc
80AF  78              354                   ld      a, b
80B0  32 5C3A         355                   ld      (FLAGS), a      ;restore FLAGS
80B3  C1              356                   pop     bc
80B4  ED 43 5C5D      357                   ld      (CM_ADD), bc    ;restore CM_ADD
80B8  C9              358                   ret
                      359
```

```
                                360
                                361  ;*****************************************************************************
                                362  ;
                                363  ; U_OR_L_ALPHA
                                364  ;
                                365  ; Inputs: current character in A.
                                366  ;
                                367  ; Outputs: sets carry flag is current letter is an upper or lower case
                                368  ;          letter.  Otherwise, resets the carry flag.
                                369  ;
                                370  ; Globals Read: none.
                                371  ;
                                372  ; Globals written: none.
                                373  ;
                                374  ; Procedures Called: none.
                                375  ;
                                376  ; Procedures Called By: cmdint
                                377  ;
                                378  ; Description: this routine checks if the current character is an upper
                                379  ;              or lower case letter.  If so, the carry flag is set.
                                380  ;              otherwise the carry flag is reset.  If the character is a
                                381  ;              lower case letter, it is converted to an upper case letter.
                                382  ;
                                383  ;*****************************************************************************
                                384
                                385
E089   FE 41                    386  u_or_l_alpha    cp      'A'             ;check if upper case letter
E08B   3F                       387                  ccf
E08C   D0                       388                  ret     nc              ;less than 'A', so return
E08D   FE 5B                    389                  cp      'Z'+1
E08F   D8                       390                  ret     c               ;is upper case letter
E0C0   FE 61                    391                  cp      'a'             ;check if lower case letter
E0C2   3F                       392                  ccf
E0C3   D0                       393                  ret     nc              ;less than 'a', so return
E0C4   FE 7B                    394                  cp      'z'+1
E0C6   D0                       395                  ret     nc              ;greater than 'z', so return
E0C7   D6 20                    396                  sub     'a'-'A'         ;convert to upper case
E0C9   37                       397                  scf                     ;set carry flag
E0CA   C9                       398                  ret
                                399
                                400
                                401  ;*****************************************************************************
                                402  ;
                                403  ; LETTER
                                404  ;
                                405  ; Inputs: current character in A.  The character is an upper case letter.
                                406  ;         hl = current string pointer.
                                407  ;         b  = parser state
                                408  ;
                                409  ; Outputs: svc_code in the parameter block.
                                410  ;          b = parser state
                                411  ;
                                412  ; Globals Read: none.
                                413  ;
                                414  ; Globals written: parms
                                415  ;
                                416  ; Procedures Called: none.
                                417  ;
                                418  ; Description: this routine translates the current character, which is
                                419  ;              known to be a letter, into a code.  If this code = err,
                                420  ;              then the letter is not a legal command character.  In this
                                421  ;              case state is set to s_error.  Otherwise, the code is the
                                422  ;              service code appropriate to the command letter and is put
                                423  ;              into svc_code in the parameter block.  state is set to s1.
                                424  ;
                                425  ;*****************************************************************************
                                426
                                427
        00FF                    428  err             equ     0FFH
                                429
                                430
E0CB   F5                       431  letter          push    af
E0CC   3E 00                    432                  ld      a, s0
E0CE   B8                       433                  cp      b
E0CF   28 04                    434                  jr      z, l_ok0        ;state = s0, so ok
E0D1   F1                       435                  pop     af
E0D2   06 FF                    436                  ld      b, s_error      ;otherwise, state is illegal
E0D4   C9                       437                  ret
E0D5   F1                       438  l_ok0           pop     af
E0D6   D5                       439                  push    de
E0D7   E5                       440                  push    hl
E0D8   21 E0F3                  441                  ld      hl, l_table     ;hl -> letter state table
E0DB   D6 41                    442                  sub     'A'
E0DD   16 00                    443                  ld      d, 0
E0DF   5F                       444                  ld      e, a
E0E0   19                       445                  add     hl, de          ;hl -> desired table entry
E0E1   7E                       446                  ld      a, (hl)
E0E2   FE FF                    447                  cp      err
E0E4   20 04                    448                  jr      nz, l_ok1       ;code is ok, so jump
E0E6   06 FF                    449                  ld      b, s_error      ;else code is illegal
E0E8   18 09                    450                  jr      l_exit
E0EA   32 DFE5                  451  l_ok1           ld      (svc_code), a   ;otherwise put code in parm block
E0ED   06 01                    452                  ld      b, s1           ;state <- s1
E0EF   E1                       453  l_exit          pop     hl
E0F0   23                       454                  inc     hl              ;advance string pointer
E0F1   D1                       455                  pop     de
E0F2   C9                       456                  ret
                                457
                                458
E0F3   07                       459  l_table         defb    7               ;Change_Sprite_Attribute
E0F4   FF                       460                  defb    err
E0F5   01                       461                  defb    1               ;Create_Sprite
E0F6   FF                       462                  defb    err
E0F7   04                       463                  defb    4               ;Erase_Sprite
E0F8   FF                       464                  defb    err
E0F9   FF                       465                  defb    err
E0FA   0A                       466                  defb    10              ;Horizontal_Scroll
```

```
E0F8  00                  469                 defb    0               !Init_Sprites
E0FC  FF                  470                 defb    eff
E0FD  FF                  471                 defb    eff
E0FE  06                  472                 defb    6               !Change_Sprite_Location
E0FF  05                  473                 defb    5               !Move_Sprite
E100  FF                  474                 defb    eff
E101  00                  475                 defb    0               !Overlap?
E102  03                  476                 defb    3               !Put_Sprite
E103  FF                  477                 defb    eff
E104  FF                  478                 defb    eff
E105  09                  479                 defb    9               !Init_Screen
E106  FF                  480                 defb    eff
E107  FF                  481                 defb    eff
E108  0B                  482                 defb    11              !Vertical_Scroll
E109  02                  483                 defb    2               !Set_Autowrap_Mode
E10A  FF                  484                 defb    eff
E10B  FF                  485                 defb    eff
E10C  FF                  486                 defb    eff
                          487
                          488
                          489 !**********************************************************************
                          490 !
                          491 ! MINUS
                          492 !
                          493 ! Inputs: hl = string pointer.
                          494 !         b  = parser state.
                          495 !
                          496 ! Outputs: b = parser state.
                          497 !
                          498 ! Globals Read: gflags
                          499 !
                          500 ! Globals Written: gflags
                          501 !
                          502 ! Procedures Called: none.
                          503 !
                          504 ! Procedures Called By: cmdint
                          505 !
                          506 ! Description: If the state is not s1, sets state to s_error.  Otherwise,
                          507 !              toggles bit neg of gflags and stays in state s1.
                          508 !
                          509 !**********************************************************************
                          510
                          511
E10D  23                  512 minus           inc     hl              !advance string pointer
E10E  3E 01               513                 ld      a, s1
E110  B8                  514                 cp      b
E111  28 03               515                 jr      z, s_ok         !state = s1, so ok
E113  04 FF               516                 ld      b, s_error      !otherwise state <- s_error
E115  C9                  517                 ret
E116  E5                  518 s_ok            push    hl
E117  21 DFF8             519                 ld      hl, gflags
E11A  7E                  520                 ld      a, (hl)
E11B  1F                  521                 rra
E11C  3F                  522                 ccf                     !toggle bit neg
E11D  17                  523                 rla
E11E  77                  524                 ld      (hl), a
E11F  E1                  525                 pop     hl
E120  C9                  526                 ret
                          527
                          528
                          529 !**********************************************************************
                          530 !
                          531 ! NUMBER
                          532 !
                          533 ! Inputs: current string pointer in HL.
                          534 !         b = parser state.
                          535 !
                          536 ! Outputs: a parameter in the parameter block.
                          537 !          b = parser state.
                          538 !
                          539 ! Globals Read: CM_ADD
                          540 !               gflags
                          541 !
                          542 ! Globals Written: CM_ADD
                          543 !                  gflags
                          544 !                  parms
                          545 !
                          546 ! Procedures Called: ININT
                          547 !                   FP28C
                          548 !
                          549 ! Procedures Called By: cmdint
                          550 !
                          551 ! Description: this routine converts the digit string starting at the
                          552 !              current value of the string pointer and ending with the
                          553 !              first non-digit character into an unsigned two byte number.
                          554 !              If the neg flag is set the 2's complement is taken.  The
                          555 !              number is put into the current parm slot in the parm block.
                          556 !              The value of #parms is incremented.  State becomes s2.
                          557 !
                          558 !**********************************************************************
                          559
                          560
E121  C5                  561 number          push    bc
E122  D5                  562                 push    de
E123  ED 5B 5C5D          563                 ld      de, (CM_ADD)    !save CM_ADD
E127  D5                  564                 push    de
E128  22 5C5D             565                 ld      (CM_ADD), hl    !set CM_ADD to point to digit string
E12B  CD 30F9             566                 call    ININT           !convert string to number. Number
                          567                                         !  ends up on calculator stack.
E12E  E5                  568                 push    hl
E12F  CD 3160             569                 call    FP28C           !put number in bc
E132  21 DFF8             570                 ld      hl, gflags
E135  CB 46               571                 bit     neg, (hl)
E137  28 07               572                 jr      z, pos
```

```
8139  78                573                ld      a, b            ;here to convert to 2's complement
813A  2F                574                cpl
813B  47                575                ld      b, a
813C  79                576                ld      a, c
813D  2F                577                cpl
813E  4F                578                ld      c, a
813F  03                579                inc     bc
8140  CB B6              580  aaa           res     neg, (hl)
8142  DD 71 00          581                ld      (ix), c         ;put number into current parm slot
8145  DD 23             582                inc     ix
8147  DD 70 00          583                ld      (ix), b
814A  DD 23             584                inc     ix
814C  E1                585                pop     hl
814D  D1                586                pop     de
814E  ED 53 9C9D        587                ld      (CM_ADD), de    ;restore CM_ADD
8152  D1                588                pop     de
8153  C1                589                pop     bc
8154  0C                590                inc     c               ;nparms <- nparms + 1
8155  06 02             591                ld      b, s2           ;state <- s2
8157  C9                592                ret
                        593
                        594
                        595  ;*****************************************************************
                        596  ;
                        597  ; COMMA
                        598  ;
                        599  ; Inputs: current string pointer in HL.
                        600  ;         B = parser state.
                        601  ;
                        602  ; Outputs: B = parser state.
                        603  ;
                        604  ; Globals Read: none.
                        605  ;
                        606  ; Globals Written: none.
                        607  ;
                        608  ; Procedures Called: none.
                        609  ;
                        610  ; Procedures Called By: cmdint
                        611  ;
                        612  ; Description: this routine recognizes a comma.  If state is s2, then
                        613  ;              the comma is skipped and state becomes s1.  Otherwise
                        614  ;              state becomes s_error.
                        615  ;
                        616  ;*****************************************************************
                        617
                        618
8158  3E 02             619  comma           ld      a, s2
815A  B8                620                  cp      b
815B  28 03             621                  jr      z, c_ok         ;if state = s2 then ok
815D  06 FF             622                  ld      b, s_error      ;else state <- s_error
815F  C9                623                  ret
8160  23                624  c_ok            inc     hl              ;advance string pointer
8161  06 01             625                  ld      b, s1           ;state <- s1
8163  C9                626                  ret
                        627
                        628
                        629  ;*****************************************************************
                        630  ;
                        631  ; SPACE
                        632  ;
                        633  ; Inputs: current string pointer in HL.
                        634  ;         B = parser state.
                        635  ;
                        636  ; Outputs: B = parser state.
                        637  ;
                        638  ; Globals Read: none.
                        639  ;
                        640  ; Globals Written: none.
                        641  ;
                        642  ; Procedures Called: none.
                        643  ;
                        644  ; Procedures Called By: cmdint
                        645  ;
                        646  ; Description: this routine recognizes a space.  If state is s0, then
                        647  ;              the space is skipped and state remains s0.  Otherwise
                        648  ;              state becomes s_end.
                        649  ;
                        650  ;*****************************************************************
                        651
                        652
8164  3E 00             653  space           ld      a, s0
8166  B8                654                  cp      b
8167  28 03             655                  jr      z, s_ok         ;if state = s0 then ok
8169  06 03             656                  ld      b, s_end        ;else state <- s_end
816B  C9                657                  ret
816C  23                658  s_ok            inc     hl              ;advance string pointer
816D  C9                659                  ret
                        660
                        661
                        662                  end
```

```
CMDINT  EK1100/11 version 10.36.14       13-Feb-84  1618:142
Command interpreter module   CMDINT.SRC

A       Reserved        ALLOC   0003            AUTOWR  0001            B       Reserved        BACKGC  0FF8
BADSCM  E069            C       Reserved        CMSADD  1C50            CMDINT  E030            COL     8001
COLOR   0004            COMMA   E15E            COMMAN  0FF7            COUR    E160            D       Reserved
DIGIT   3009            DOSCMD  E074            E       Reserved        ENDW0   E080            ENDW1   E368
ERR     00FF            ERROR   0008            FINDSN  0070            FINDSS  E092            FLAGS   5C3A
FPZBC   3160            FSOK    E0AE            GFLAGS  0FF8            GLOBAL  0010            M       Reserved
HEIGHT  0003            ININT   30F9            ISPSMA  3001 EX         L       Reserved        LETTER  E0C9
LMS     0002            LSEXIT  E0EF            LSOK0   E3D5            LSOK1   ECE6            LSTABL  E0F3
M       Reserved        MAXSCO  0020            MAXSPO  5010            MAXSSP  0FFA            MINUS   E105
MSOK    E116            NAME    0FFC            NEG     C000            NOTSAL  E044            NOTSCO  E060
NOTSDI  E057            NOTSMI  E040            NUMBER  F121            NUMSPA  0FEC            OK      E02C
OVERWR  0001            PARM1   0FE8            PARM2   0FE0            PARM3   0FE4            PARM4   0FF3
PARM5   0FF5            PBITMA  0C05            PDS     E140            PFLAGS  0007            PSW     Reserved
ROW     0000            SCRATC  0FE3            SCREEN  0FF9            SPACE   E164            SP      Reserved
SPRITE  0FFE            SSEND   0003            SSERRO  00FF            SSOK    E16C            S0      0000
S1      0001            S2      0002            USORSL  E089            WHILE0  E01C            WHILE1  E027
WIDTH   0002

     No errors detected
```

```
Cross reference listing (MREF version 4.7)

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

ALLOC     159#
AUTOWR    131#
BACKGC    135#   136
BAD_CM    279    282#
CM_ADD    103#   341    3468    3578    563    5658    5878
CMDINT    226#
COL       153#   154
COLOR     156#   157
COMMA     276    619#
COMMAN    136#   137    2298    300
C_OK      621    624#
DIGIT     109#   270
DO_CMD    260    288    290#
ENDW0     248    306#
ENDW1     255    284#
ERR       430#   449    462     466     466     467     470
                 471    474     477     478     480     481
                 484    485     486
ERROR     105#   251    351
FIND_N    108#   349
FIND_S    233    341#
FLAGS     102#   343    3488    3558
FPZBC     107#   569
F_OK      350    353#
GFLAGS    129#   132    230     519     570
GLOBAL    123#
HEIGHT    155#   156
ININT     106#   566
I_M_MA    111#   295
LETTER    264    433#
LMS       161#
L_EXIT    452    455#
L_OK0     436    440#
L_OK1     450    453#
L_TABL    443    461#
MAX_CO    121#
MAX_RO    120#
MAX_SP    132#   133
MINUS     268    512#
M_OK      515    518#

NAME      128#   129    2278    345
NEG       130#   231    571     580
NOT_AL    263    265#
NOT_CO    275    278#
NOT_DI    271    274#
NOT_MI    267    270#
NUMBER    272    561#
NUM_PA    144#   145    2918
OK        250    253#
OVERWR    160#
PARM1     145#   146    245
PARM2     146#   147
PARM3     147#   148
PARM4     148#   149
PARM5     149#
PARM5     137#   138    143
PBITMA    157#   158
PDS       572    580#
ROW       152#   153
S0        168#   243    246     289     434     693
S1        169#   454    513     625
S2        170#   591    619
SCRATC    138#
SCREEN    133#   135
SFLAGS    158#
SPACE     280    653#
SPRCSO    122#   127    223
SPRITE    127#   148
SVC_CO    143#   144    4538
S_END     171#   253    656
S_ERRO    172#   282    438     451     516     622
S_OK      655    658#
U_OR_L    262    386#
WHILE0    244#   305
WHILE1    249#   265    269     273     277     281     183
WIDTH     154#   155
```

```
                        1                       NAME  IFHANDLER
                        2                       SUBTTL Interface Handler module
                        3
                        4   ;***************************************************************************
                        5   ;
                        6   ; Interface Handler module
                        7   ;
                        8   ; Inputs: a parameter block.
                        9   ;
                       10   ; Outputs: a status code in BC.
                       11   ;
                       12   ; Description: this module translates a parameter block into a call to
                       13   ;               the appropriate sprite service.  The number of parameters
                       14   ;               specified in the parm block is checked against a table
                       15   ;               entry for the service containing the correct number of
                       16   ;               parameters.  If these values do not match, the report
                       17   ;               "CI Parameter error" is produced.  Otherwise the interface
                       18   ;               routine appropriate to the service is invoked.  This
                       19   ;               interface routine puts the values of the parms in the
                       20   ;               parm block into the appropriate registers.  The sprite
                       21   ;               service is then invoked.  When the service returns, the
                       22   ;               status code is checked and, if an error occurred, an
                       23   ;               appropriate report is produced.
                       24   ;
                       25   ;***************************************************************************
                       26
                       27
                       28   ; Imports
                       29
0000                   30   ERROR          equ     0
E219                   31   INIT_SPRITES   equ     0E219H
E4E9                   32   INITSCREEN     equ     0E4E9H
E278                   33   CREATE_SPRITE  equ     0E278H
E2C9                   34   SET_AUTOWRAP   equ     0E2C9H
E307                   35   PUT_SPRITE     equ     0E307H
E381                   36   ERASE_SPRITE   equ     0E381H
E426                   37   MOVE_SPRITE    equ     0E426H
E450                   38   CM_LOC         equ     0E450H
E466                   39   CM_ATTR        equ     0E466H
E484                   40   OVERLAP        equ     0E484H
E506                   41   H_SCROLL       equ     0E506H
E771                   42   V_SCROLL       equ     0E771H
                       43
                       44
                       45
                       46   ; Globals
                       47
                       48                   INCLUDE spritedef
                       49
                       50   ; Global constants
                       51
0018                   52   max_rows       equ     24              ;number of rows on the screen
0020                   53   max_cols       equ     32              ;number of columns on the screen
E000                   54   spredef        equ     0E000H          ;location of sprites code
0015                   55   globallen      equ     29              ;length of global vars
                       56
                       57   ; Global variables
                       58
0FFE                   59   sprites        equ     spredef-2       ;location of sprites data
0FFC                   60   name           equ     sprites-2       ;name of command string var.
0FFB                   61   gflags         equ     name-1          ;global flags
0000                   62   nos            equ     0               ;  bit 0 - parm < 0
0001                   63   autowrap       equ     1               ;  bit 1 - autowrap mode flag
0FFA                   64   max_sprites    equ     gflags-1        ;number of sprites allocated
0FF9                   65   screen_ht      equ     max_sprites-1   ;number of lines of screen used by
                       66                                          ;  sprites
0FF8                   67   backgcolor     equ     screen_ht-1     ;permanent background color
0FF7                   68   command        equ     backgcolor-1    ;number of command being executed
0FE8                   69   parms          equ     command-12      ;parameter block
0FE3                   70   scratchpad     equ     parms-8         ;workspace area
                       71
                       72   ; Data structures
                       73
                       74   ; parameter block
0FE8                   75   svc_code       equ     parms           ;code identifying service
0FEC                   76   num_parms      equ     svc_code+1      ;number of parms found
0FED                   77   parm1          equ     num_parms+1     ;first parm
0FEF                   78   parm2          equ     parm1+2
0FF1                   79   parm3          equ     parm2+2
0FF3                   80   parm4          equ     parm3+2
0FF5                   81   parm5          equ     parm4+2         ;last possible parm
                       82
                       83   ; sprite data offsets
0000                   84   row            equ     0               ;row position
0001                   85   col            equ     row+1           ;column position
0002                   86   width          equ     col+1           ;sprite width
0003                   87   height         equ     width+1         ;sprite height
0004                   88   color          equ     height+1        ;sprite color
0005                   89   pbitmap        equ     color+1         ;definition address
0007                   90   sflags         equ     pbitmap+2       ;local sprite flags
0000                   91   alloc          equ     0               ;  bit 0 - allocated?
0001                   92   overwrite      equ     1               ;  bit 1 - sprite overwrote something
0002                   93   lhs            equ     2               ;  bit 2 - sprite in lower half screen
                       94
                       95
```

```
                                    96
                                    97  !******************************************************************************
                                    98  !
                                    99  ! I_F_HANDLER
                                   100  !
                                   101  ! Inputs: a parameter block.
                                   102  !
                                   103  ! Outputs: a status code in BC.
                                   104  !
                                   105  ! Globals Read: parms
                                   106  !
                                   107  ! Globals Written: none.
                                   108  !
                                   109  ! Procedures Called: ERROR
                                   110  !                   jmp_hl
                                   111  !                   jmp_ix
                                   112  !
                                   113  ! Procedures Called By: CMDINT
                                   114  !
                                   115  ! Description: this is the main routine of this module.  It translates
                                   116  !              the values in the parm block into a call to a sprite
                                   117  !              service.  The interfaces to the sprite services are
                                   118  !              described in a table.  For each service the is an entry
                                   119  !              containing the correct number of parameters, the address
                                   120  !              of the interface routine appropriate to that number of
                                   121  !              parameters, and the address of the service.  The number
                                   122  !              of parms in the table entry is check against the value in
                                   123  !              the parm block.  If they do not agree, report "G" is produced.
                                   124  !              If they do agree, the interface routine indicated in the
                                   125  !              table entry is called.  The service indicated in the table
                                   126  !              entry is then called.   If the part of the status code in C
                                   127  !              is non-zero, then an error occurred in executing the sprite
                                   128  !              service, and an appropriate report is produced.
                                   129  !
                                   130  !******************************************************************************
        =E16E                      131                      ORG     0E16EH
                                   132
E16E'  3A 0FE5                     133  i_f_handler     ld      a, (svc_code)   !check num_parms against the
                                   134                                          !  value in i_f_table
E171'  47                          135                  ld      b, a
E172'  CB 27                       136                  sla     a
E174'  CB 27                       137                  sla     a
E176'  80                          138                  add     a, b            !a = 5*svc_code
E177'  21 E189'                    139                  ld      hl, i_f_table
E17A'  5F                          140                  ld      e, a
E17B'  16 00                       141                  ld      d, 0
E17D'  19                          142                  add     hl, de          !hl -> table entry
E17E'  3A 0FEC                     143                  ld      a, (num_parms)
E181'  BE                          144                  cp      (hl)
E182'  28 02                       145                  jr      z, ok           !if num_parms correct, then jump
E184'  CF                          146                  rst     ERROR           !else produce report "G"
E185'  19                          147                  defb    25
E186'  23                          148  ok              inc     hl              !hl -> i/f routine addr
E187'  5E                          149                  ld      e, (hl)
E188'  23                          150                  inc     hl
E189'  56                          151                  ld      d, (hl)
E18A'  23                          152                  inc     hl
E18B'  EB                          153                  ex      de, hl          !hl = i/f routine addr
E18C'  1A                          154                  ld      a, (de)         !de -> service addr
E18D'  4F                          155                  ld      c, a
E18E'  13                          156                  inc     de
E18F'  1A                          157                  ld      a, (de)
E190'  47                          158                  ld      b, a
E191'  DD 21 0000                  159                  ld      ix, 0
E195'  DD 09                       160                  add     ix, bc          !ix = service addr
E197'  DD E5                       161                  push    ix
E199'  DD 21 0FE0                  162                  ld      ix, parm1       !ix -> parameters
E19D'  CD E1B2'                    163                  call    jmp_hl          !invoke i/f routine
E1A0'  DD E1                       164                  pop     ix
E1A2'  CC E1B3'                    165                  call    jmp_ix          !invoke sprite service
E1A5'  CB 41                       166                  bit     0, c            !check status code
E1A7'  C8                          167                  ret     z               !if c0 = 0, then no error
E1A8'  79                          168                  ld      a, c
E1A9'  FE 01                       169                  cp      1
E1AB'  20 02                       170                  jr      nz, enough_room !if c <> 1, then not out of room
E1AD'  CF                          171                  rst     ERROR           !else produce report "A"
E1AE'  03                          172                  defb    3
E1AF'  CF                          173  enough_room     rst     ERROR           !produce report "A"
E1B0'  09                          174                  defb    9
E1B1'  C9                          175                  ret
                                   176
                                   177
                                   178  !******************************************************************************
                                   179  !
                                   180  ! JMP_HL
                                   181  !
                                   182  ! Inputs: HL = address to be called.
                                   183  !
                                   184  ! Outputs: none.
                                   185  !
                                   186  ! Globals Read: none.
                                   187  !
                                   188  ! Globals Written: none.
                                   189  !
                                   190  ! Procedures Called: none.
                                   191  !
                                   192  ! Procedures Called By: i_f_handler
                                   193  !
                                   194  ! Description: this routine is called to call the routine whose entry
                                   195  !              point is in HL.
                                   196  !
                                   197  !******************************************************************************
                                   198
                                   199
                                   200  jmp_hl          jp      (hl)
```

```
                        201
                        202
                        203   !*****************************************************************************
                        204   !
                        205   ! JMP_IX
                        206   !
                        207   ! Inputs: IX = address to be called.
                        208   !
                        209   ! Outputs: none.
                        210   !
                        211   ! Globals Read: none.
                        212   !
                        213   ! Globals Written: none.
                        214   !
                        215   ! Procedures Called: none.
                        216   !
                        217   ! Procedures Called By: i_f_handler
                        218   !
                        219   ! Description: this routine is called to call the routine whose entry
                        220   !              point is in IX.
                        221   !
                        222   !*****************************************************************************
                        223
                        224
8183' DD E9             225   jmp_ix          jp      (ix)
                        226
                        227
                        228   !This table contains the interface definitions for the sprite services.  The
                        229   !  first item is the number of parameters to be passed to the service.  The
                        230   !  second item is the interface routine for doing this.  The third item is the
                        231   !  address of the service.
                        232
8185' 01                233   i_f_table       defb    1
8186' 81F1'             234                   defw    i_f1
8188' 8219              235                   defb    INIT_SPRITES
                        236
818A' 05                237                   defb    5
818B' 8206'             238                   defb    i_f5
818D' 8278              239                   defw    CREATE_SPRITE
                        240
818F' 01                241                   defb    1
8190' 81F1'             242                   defw    i_f1
8192' 82C9              243                   defw    SET_AUTOWRAP
                        244
8194' 03                245                   defb    3
8195' 81FC'             246                   defb    i_f3
8197' 8307              247                   defb    PUT_SPRITE
                        248
8199' 01                249                   defb    1
819A' 81F1'             250                   defb    i_f1
819C' 8381              251                   defb    ERASE_SPRITE
                        252
819E' 03                253                   defb    3
819F' 81FC'             254                   defb    i_f3
81A1' 8426              255                   defb    MOVE_SPRITE
                        256
81A3' 03                257                   defb    3
81A4' 81FC'             258                   defb    i_f3
81A6' 8450              259                   defb    CH_LOC
                        260
81A8' 02                261                   defb    2
81A9' 81F9'             262                   defb    i_f2
81AB' 8466              263                   defb    CH_ATTR
                        264
81AD' 02                265                   defb    2
81AE' 81F9'             266                   defb    i_f2
81B0' 8484              267                   defb    OVERLAP
                        268
81B2' 03                269                   defb    3
81B3' 81FC'             270                   defb    i_f3
81B5' 84E9              271                   defb    INITSCREEN
                        272
81B7' 01                273                   defb    1
81B8' 81F1'             274                   defb    i_f1
81BA' 8506              275                   defb    H_SCROLL
                        276
81BC' 01                277                   defb    1
81BD' 81F1'             278                   defb    i_f1
81BF' 8771              279                   defb    V_SCROLL
                        280
                        281
                        282   !*****************************************************************************
                        283   !
                        284   ! i_f1
                        285   !
                        286   ! Inputs: ix -> first parameter
                        287   !
                        288   ! Outputs: a <- value of first parameter
                        289   !
                        290   ! Globals Read: none.
                        291   !
                        292   ! Globals Written: none.
                        293   !
                        294   ! Procedures Called: none.
                        295   !
                        296   ! Procedures Called By: i_f_handler
                        297   !
                        298   ! Description: this is the interface routine for sprite services with
                        299   !              one parameter.
                        300   !
                        301   !*****************************************************************************
                        302
                        303
81F1' DD 7E 00          304   i_f1            ld      a, (ix)
81F4' C9                305                   ret
```

```
                        306
                        307
                        308   !**********************************************************************
                        309   !
                        310   ! I_F2
                        311   !
                        312   ! Inputs: ix -> first parameter
                        313   !
                        314   ! Outputs: a <- value of first parameter
                        315   !          d <- value of second parameter
                        316   !
                        317   ! Globals Read: none.
                        318   !
                        319   ! Globals Written: none.
                        320   !
                        321   ! Procedures Called: none.
                        322   !
                        323   ! Procedures Called By: i_f_handler
                        324   !
                        325   ! Description: this is the interface routine for sprite services with
                        326   !              two parameters.
                        327   !
                        328   !**********************************************************************
                        329
                        330
E1F5' DD 7E 00          331   i_f2            ld      a, (ix)
E1F8' DD 56 02          332                   ld      d, (ix+2)
E1FB' C9                333                   ret
                        334
                        335
                        336   !**********************************************************************
                        337   !
                        338   ! I_F3
                        339   !
                        340   ! Inputs: ix -> first parameter
                        341   !
                        342   ! Outputs: a <- value of first parameter
                        343   !          d <- value of second parameter
                        344   !          e <- value of third parameter
                        345   !
                        346   ! Globals Read: none.
                        347   !
                        348   ! Globals Written: none.
                        349   !
                        350   ! Procedures Called: none.
                        351   !
                        352   ! Procedures Called By: i_f_handler
                        353   !
                        354   ! Description: this is the interface routine for sprite services with
                        355   !              three parameters.
                        356   !
                        357   !**********************************************************************
                        358
                        359
E1FC' DD 7E 00          360   i_f3            ld      a, (ix)
E1FF' DD 56 02          361                   ld      d, (ix+2)
E202' DD 5E 04          362                   ld      e, (ix+4)
E205' C9                363                   ret
                        364
                        365
                        366   !**********************************************************************
                        367   !
                        368   ! I_F5
                        369   !
                        370   ! Inputs: ix -> first parameter
                        371   !
                        372   ! Outputs: a <- value of first parameter
                        373   !          b <- value of second parameter
                        374   !          c <- value of third parameter
                        375   !          d <- value of fourth parameter
                        376   !         hl <- value of fifth parameter
                        377   !
                        378   ! Globals Read: none.
                        379   !
                        380   ! Globals Written: none.
                        381   !
                        382   ! Procedures Called: none.
                        383   !
                        384   ! Procedures Called By: i_f_handler
                        385   !
                        386   ! Description: this is the interface routine for sprite services with
                        387   !              five parameters.
                        388   !
                        389   !**********************************************************************
                        390
                        391
E206' DD 7E 00          392   i_f5            ld      a, (ix)
E209' DD 46 02          393                   ld      b, (ix+2)
E20C' DD 4E 04          394                   ld      c, (ix+4)
E20F' DD 56 06          395                   ld      d, (ix+6)
E212' DD 6E 08          396                   ld      l, (ix+8)
E215' DD 66 09          397                   ld      h, (ix+9)
E218' C9                398                   ret
                        399
                        400
                        401                   end
```

```
IPHANDLER       CRI80/11 version 10.36.14       15-May-84  9:52:10
Interface handler module   IPHANDLER.SRC

A       Reserved        ALLOC   0030            AUTOWR  0001            B       Reserved        BACKGC  0FF8
C       Reserved        CHRATT  E466            CHRLOC  E450            CCL     0001            CGLOR   0004
COMMAN  0FF7            CREATE  E278            D       Reserved        E       Reserved        ENOUGH  E1AF
ERASES  E381            ERROR   0008            GPLAGS  0FF8            GLOBAL  0010            H       Reserved
HEIGHT  0003            HSSCRG  E506            INITSC  E4E9            INITSS  E219            ISPSMA  E16E
ISPSTA  E185            ISR1    E1F1            ISR2    E1F5            ISP3    E1FC            ISPS    E206
JMPSML  E182            JMPSIZ  6183            L       Reserved        LMS     0002            M       Reserved
MAXSCO  0020            MAXSRC  0019            MAXSSP  0FFA            MCVESS  E424            NAME    0FFC
NEG     0000            NUMSPA  CFEC            OK      E186            OVERLA  E484            OVERMR  0001
PARM5   0FE8            PARM1   0FEC            PARM2   0FEF            PARM3   0FF1            PARM4   0FF3
PARM5   0FF5            PRITMA  0005            PSW     Reserved        PUTSSP  E307            ROM     0000
SCRATC  0FE3            SCREEN  0FF9            SETBAU  E2C9            SPLAGS  0007            SP      Reserved
SPRCOD  8000            SPRITE  0FFE            SVCSCO  0FE8            VBSCRO  E771            WIDTH   0002

     No errors detected
```

```
SPRSVC  CRI80/11 version 10.36.14       15-May-84  9:51:31
Sprite services module   SPRSVC.SRC

                                1               NAME: SPRSVC
                                2               SUBTTL  Sprite services module
                                3
                                4
                                5   ;*****************************************************************
                                6   ;
                                7   ; Sprite Services module
                                8   ;
                                9   ; Inputs: registers set to parameter values for called service.
                               10   ;
                               11   ; Outputs: status code in BC: 0000 - OK
                               12   ;                               0001 - not enough memory
                               13   ;                               0003 - illegal sprite id
                               14   ;                               0005 - illegal color
                               15   ;                               0007 - illegal number-lines
                               16   ;                               0100 - OK, something overwritten
                               17   ;                               0200 - OK, overlap
                               18   ;
                               19   ; Description: this module implements the various sprite graphics services.
                               20   ;               Upon entry to this module, the parameters to the desired
                               21   ;               service are put into the appropriate registers.  A status
                               22   ;               code is returned in BC (see above).
                               23   ;
                               24   ;*****************************************************************
                               25
                               26
                               27   ; Imports (Note:  all symbols in upper case are imported)
                               28
                               29   ;From the Home
                               30
                               31   ;Variables
                               32
 =9C3A                         33   Y               equ     9C3AH
 =9C82                         34   BANTOP          equ     9C82H
 =9C36                         35   CHARS           equ     9C36H
 =9C8D                         36   ATTR_P          equ     9C8DH
 =9C8F                         37   ATTR_T          equ     9C8FH
 =1C3C                         38   TYPLAG          equ     9C3CH
 =9C48                         39   BORDCR          equ     9C48H
 =0BFE                         40   BORDPT          equ     0FEH
                               41
                               42   ;Routines
                               43
 =8582                         44   SET_AT          equ     8582H
 =8500                         45   SENDTV          equ     8500H
 =1230                         46   SELECT          equ     1230H
 =097F                         47   CLS_B           equ     097FH
                               48
                               49
                               50   ;From module IOL
                               51
                               52   ;Variables
                               53
 =E890                         54   CHTBL   equ     0E890H                  ;  Address of ROM Char.Table
                               55
                               56   ;Services
                               57
 =E89D                         58   WRITE_CHAR      equ     0E89DH
 =E90E                         59   SET_PRINT_POS   equ     0E90EH
 =E91F                         60   GET_CHAR        equ     0E91FH
                               61
                               62
                               63   ; Globals
                               64
                               65                   INCLUDE spritedef
                               66
                               67   ; Global constants
                               68
 =0018                         69   max_rows        equ     24              ;number of rows on the screen
 =0020                         70   max_cols        equ     32              ;number of columns on the screen
 =E000                         71   sprcode         equ     0E000H          ;location of sprites code
 =0010                         72   globallen       equ     29              ;length of global vars
```

```
                                73
                                74  ; Global variables
                                75
=0FFE                           76  sprites         equ     sprcode-2       ;location of sprites data
=0FFC                           77  name            equ     sprites-2       ;name of command string var.
=0FFA                           78  gflags          equ     name-1          ;global flags
=C000                           79  neg             equ     0               ;  bit 0 - parm < 0
=0001                           80  autowrap        equ     1               ;  bit 1 - autowrap mode flag
=0FF9                           81  max_sprites     equ     gflags-1        ;number of sprites allocated
=0FF9                           82  screen_ht       equ     max_sprites-1   ;number of lines of screen used by
                                83                                          ;  sprites
=0FF8                           84  backgcolor      equ     screen_ht-1     ;permanent background color
=0FF7                           85  command         equ     backgcolor-1    ;number of command being executed
=0FE8                           86  parms           equ     command-12      ;parameter block
=0FE3                           87  scratchpad      equ     parms-5         ;workspace area
                                88
                                89  ; Data structures
                                90
                                91  ; parameter block
=0FE8                           92  svc_code        equ     parms           ;code identifying service
=0FEC                           93  num_parms       equ     svc_code+1      ;number of parms found
=0FED                           94  parm1           equ     num_parms+1     ;first parm
=0FEF                           95  parm2           equ     parm1+2
=0FF1                           96  parm3           equ     parm2+2
=0FF3                           97  parm4           equ     parm3+2
=0FF5                           98  parm5           equ     parm4+2         ;last possible parm
                                99
                               100  ; sprite data offsets
=0000                          101  row             equ     0               ;row position
=0001                          102  col             equ     row+1           ;column position
=0002                          103  width           equ     col+1           ;sprite width
=0003                          104  height          equ     width+1         ;sprite height
=0004                          105  color           equ     height+1        ;sprite color
=0005                          106  pbitmap         equ     color+1         ;definition address
=0007                          107  sflags          equ     pbitmap+2       ;local sprite flags
=0000                          108  alloc           equ     0               ;  bit 0 - allocated?
=0001                          109  overwrite       equ     1               ;  bit 1 - sprite overwrote something
=0002                          110  lhs             equ     2               ;  bit 2 - sprite in lower half screen
                               111
                               112
                               113
                               114  ;***************************************************************
                               115  ;
                               116  ; INIT_SPRITES
                               117  ;
                               118  ; Inputs: A = number-sprites, the number of sprites for which memory is
                               119  ;             to be allocated.
                               120  ;
                               121  ; Outputs: BC = 0000 - OK
                               122  ;                0001 - not enough memory
                               123  ;
                               124  ; Globals Read: RAMTOP
                               125  ;
                               126  ; Globals Written: screen_ht
                               127  ;                  sflags
                               128  ;                  max_sprites
                               129  ;                  sprites
                               130  ;                  sprites data structure
                               131  ;
                               132  ; Procedures Called: none.
                               133  ;
                               134  ; Procedures Called By:  I_P_HANDLER
                               135  ;                        user program
                               136  ;
                               137  ; Description: this routines allocates memory for sprite data.  (max_sprites)
                               138  ;              is set to number-sprites.  One sprite requires 8 bytes of
                               139  ;              data.  If there is not enough free memory between (RAMTOP)
                               140  ;              and the sprites global variables, then 01 is returned.
                               141  ;              Otherwise, (sprites) is set to point to the start of the
                               142  ;              sprite data area and the sprite data is cleared.
                               143  ;
                               144  ;***************************************************************
                               145
=8219                          146          ORG     08219H
                               147
                               148
8219' 21 0FF9                  149  init_sprites    ld      hl, screen_ht   ;initialize screen_ht to 24
821C' 36 18                    150                  ld      (hl), max_rows
821E' 21 0FFA                  151                  ld      hl, gflags
8221' 36 00                    152                  ld      (hl), 0         ;clear global flags
8223' 32 0FF9                  153                  ld      (max_sprites), a
8226' 4F                       154                  ld      c, a            ;calc. where the sprite data
                               155                                          ;  should go
8227' 06 00                    156                  ld      b, 0
8229' A7                       157                  and     a
822A' CB 11                    158                  rl      c
822C' CB 10                    159                  rl      b
822E' CB 11                    160                  rl      c
8230' CB 10                    161                  rl      b
8232' CB 11                    162                  rl      c
8234' CB 10                    163                  rl      b
8236' C5                       164                  push    bc              ;bc = #ofnumber-sprites
8237' 21 0000                  165                  ld      hl, sprcode
823A' A7                       166                  and     a
823B' ED 42                    167                  sbc     hl, bc
823D' 01 0010                  168                  ld      bc, globallen
8240' A7                       169                  and     a
8241' ED 42                    170                  sbc     hl, bc          ;hl -> start of sprite data
8243' C1                       171                  pop     bc
8244' ED 5B 5C02               172                  ld      de, (RAMTOP)    ;see if there is enough free memory
8248' E5                       173                  push    hl
8249' A7                       174                  and     a
824A' ED 52                    175                  sbc     hl, de
824C' E1                       176                  pop     hl
824D' 30 04                    177                  jr      nc, enough_room ;if enough room, then jump
824F' 01 0001                  178                  ld      bc, 0001        ;else return 01
8252' C9                       179                  ret
8253' 22 0FFE                  180  enough_room     ld      (sprites), hl
8256' 36 00                    181  clear_loop      ld      (hl), 0         ;clear sprite data area
8258' 0B                       182                  dec     bc
8259' 23                       183                  inc     hl
825A' 79                       184                  ld      a, c
825B' B1                       185                  or      c
825C' 20 F8                    186                  jr      nz, clear_loop
825E' C9                       187                  ret                     ;return 00
```

```
                        188
                        189
                        190   ;**************************************************************************
                        191   ;
                        192   ; SPR_ADDR
                        193   ;
                        194   ; Inputs: A = sprite-id
                        195   ;
                        196   ; Outputs: IX <- address of sprite record for sprite sprite-id
                        197   ;
                        198   ; Globals Read: sprites
                        199   ;
                        200   ; Globals Written: none.
                        201   ;
                        202   ; Procedures Called: none.
                        203   ;
                        204   ; Procedures Called By: create_sprite
                        205   ;                       check_id
                        206   ;                       put_sprite
                        207   ;                       erase_sprite
                        208   ;                       ch_loc
                        209   ;                       ch_attr
                        210   ;                       overlap
                        211   ;                       upd_cols
                        212   ;                       upd_rows
                        213   ;
                        214   ; Description: this routine calculates the address of the start of the
                        215   ;              record for the sprite with id = sprite-id.  This address
                        216   ;              is returned in IX.  Note that each record is 8 bytes long.
                        217   ;
                        218   ;**************************************************************************
                        219
                        220
8259'  C5                221   spr_addr        push    bc
825A'  06 00             222                   ld      b, 0
825C'  4F                223                   ld      c, a
825D'  A7                224                   and     a
825E'  CB 11             225                   rl      c
8260'  CB 10             226                   rl      b
8262'  CB 11             227                   rl      c
8264'  CB 10             228                   rl      b
8266'  CB 11             229                   rl      c
8268'  CB 10             230                   rl      b               ;bc = 8*sprite-id
826A'  DD 2A DFF8        231                   ld      ix, (sprites)   ;ix -> start of sprites data
826E'  DD 09             232                   add     ix, bc          ;ix -> start of record sprite-id
8270'  C1                233                   pop     bc
8271'  C9                234                   ret
                        235
                        236
                        237   ;**************************************************************************
                        238   ;
                        239   ; CREATE_SPRITE
                        240   ;
                        241   ; Inputs: A = sprite-id
                        242   ;         B = width
                        243   ;         C = height
                        244   ;         D = color
                        245   ;        HL = address of bitmap
                        246   ;
                        247   ; Outputs: BC = 0000 - OK
                        248   ;              = 0003 - illegal sprite-id
                        249   ;              = 0005 - illegal color
                        250   ;
                        251   ; Globals Read: sprites
                        252   ;
                        253   ; Globals Written: sprite data structure
                        254   ;
                        255   ; Procedures Called: spr_adder
                        256   ;
                        257   ; Procedures Called By: I_P_HANDLER
                        258   ;                       user program
                        259   ;
                        260   ; Description: this routine initializes a record for the sprite identified
                        261   ;              by sprite-id.  If the sprite-id parameter is not less than
                        262   ;              (max_sprites), then it is illegal and 03 is returned in BC.
                        263   ;              If the color parameter is greater than 7, then it is
                        264   ;              illegal and 05 is returned in BC.  If the parameters are
                        265   ;              legal, then the slots of the record are filled in and the
                        266   ;              alloc flag of sflags is set to indicate that the record has
                        267   ;              been filled in.
                        268   ;
                        269   ;**************************************************************************
                        270
                        271
8272'  E5                272   create_sprite   push    hl              ;validate sprite-id < (max_sprites)
8273'  21 DFFA           273                   ld      hl, max_sprites
8276'  BE                274                   cp      (hl)
8277'  E1                275                   pop     hl
8278'  38 04             276                   jr      c, id_ok        ;if sprite-id ok, then jump
827A'  01 0003           277                   ld      bc, 0003        ;else return 03 in BC
827D'  C9                278                   ret
827E'  F5                279   id_ok           push    af              ;validate color < 8
827F'  3E 07             280                   ld      a, 7
8281'  BA                281                   cp      d
8282'  30 05             282                   jr      nc, color_ok    ;if color ok, then jump
8284'  F1                283                   pop     af
8285'  01 0005           284                   ld      bc, 0005        ;else return 05 in BC
8288'  C9                285                   ret
8289'  F1                286   color_ok        pop     af
828A'  CD 8259'          287                   call    spr_addr        ;ix -> start of sprite record
828D'  DD 36 00 00       288                   ld      (ix+row), 0     ;row <- 0
8291'  DD 36 01 00       289                   ld      (ix+col), 0     ;col <- 0
8295'  DD 70 02          290                   ld      (ix+width), b   ;width <- b
8298'  DD 71 03          291                   ld      (ix+height), c  ;height <- c
829B'  DD 72 04          292                   ld      (ix+color), d   ;color <- d
829E'  DD 75 05          293                   ld      (ix+bitmap), l  ;bitmap <- hl
82A1'  DD 74 06          294                   ld      (ix+bitmap+1), h
82A4'  DD CB 37 C6       295                   set     alloc, (ix+sflags)      ;mark record as initialized
82A8'  01 0000           296                   ld      bc, 0           ;return 00 in BC
82AB'  C9                297                   ret
                        298
                        299
```

```
                        300   ;**********************************************************************
                        301   ;
                        302   ; CHECK_ID
                        303   ;
                        304   ; Inputs: A = sprite-id
                        305   ;
                        306   ; Outputs: CF is set if sprite-id is legal, reset if it is illegal
                        307   ;
                        308   ; Globals Read: sprites
                        309   ;               sprite data structure
                        310   ;
                        311   ; Globals Written: none.
                        312   ;
                        313   ; Procedures Called: spr_addr
                        314   ;
                        315   ; Procedures Called By: put_sprite
                        316   ;                       erase_sprite
                        317   ;                       ch_loc
                        318   ;                       ch_attr
                        319   ;                       overlap
                        320   ;
                        321   ; Description: this routine checks (1) that sprite-id is in the range
                        322   ;              (0..max_sprites-1) and (2) that the record for that
                        323   ;              sprite-id has been initialized.
                        324   ;
                        325   ;**********************************************************************
                        326
                        327
82B2*  00 E5            328   check_id        push    ix
82B4*  E5               329                   push    hl              ;validate sprite-id < (max_sprites)
82B5*  21 0FFA          330                   ld      hl, max_sprites
82B8*  BE               331                   cp      (hl)
82B9*  E1               332                   pop     hl
82BA*  D0               333                   ret     nc              ;sprite-id out of range
82BB*  CD 825F*         334                   call    spr_addr
82BE*  DD CB 07 0E      335                   rrc     (ix+0flags)
82C2*  DD CB 07 06      336                   rlc     (ix+0flags)     ;alloc flag is now in carry flag
82C6*  DD E1            337                   pop     ix
82C8*  C9               338                   ret
                        339
                        340
                        341   ;**********************************************************************
                        342   ;
                        343   ; SET_AUTOWRAP
                        344   ;
                        345   ; Inputs: A = 0 - off
                        346   ;           <> 0 - on
                        347   ;
                        348   ; Outputs: BC = 0000 - OK
                        349   ;
                        350   ; Globals Read: none.
                        351   ;
                        352   ; Globals Written: gflags
                        353   ;
                        354   ; Procedures Called: none.
                        355   ;
                        356   ; Procedures Called By: I_P_HANDLER
                        357   ;                       user program
                        358   ;
                        359   ; Description: this routines sets the autowrap flag according to the
                        360   ;              value of A.  If A = 0, the flag is reset; otherwise
                        361   ;              the flag is set.
                        362   ;
                        363   ;**********************************************************************
                        364
                        365
82C9*  E5               366   set_autowrap    push    hl
82CA*  21 0FFB          367                   ld      hl, gflags
82CD*  A7               368                   and     a
82CE*  28 04            369                   jr      z, off          ;if A = 0, then jump
82D0*  CB CE            370                   set     autowrap, (hl)  ;else turn flag on
82D2*  18 02            371                   jr      sa_exit
82D4*  CB 8E            372   off             res     autowrap, (hl)
82D6*  E1               373   sa_exit         pop     hl
82D7*  01 0000          374                   ld      bc, 0           ;return 00 in BC
82DA*  C9               375                   ret
                        376
                        377
                        378   ;**********************************************************************
                        379   ;
                        380   ; DISPLAY_CHAR
                        381   ;
                        382   ; Inputs: B = row
                        383   ;         C = column
                        384   ;         DE = location of dot pattern for char to be written
                        385   ;
                        386   ; Outputs: DE = location of next char
                        387   ;
                        388   ; Globals Read: none.
                        389   ;
                        390   ; Globals Written: CHTBL
                        391   ;
                        392   ; Procedures Called: SET_PRINT_POS
                        393   ;                    WRITE_CHAR
                        394   ;
                        395   ; Procedures Called By: put_sprite
                        396   ;                       erase_sprite
                        397   ;
                        398   ; Description: this routine displays the specified character at the
                        399   ;              specified screen location.
                        400   ;
                        401   ;**********************************************************************
                        402
                        403
82DB*  E5               404   display_char    push    hl
82DC*  C5               405                   push    bc
82DD*  C5               406                   push    bc
82DE*  CD 890E          407                   call    SET_PRINT_POS   ;set the print position to BC
82E1*  C1               408                   pop     bc
82E2*  3E 20            409                   ld      a, ' '
82E4*  ED 53 8890       410                   ld      (CHTBL), de     ;replace (CHTBL) with loc of char
82E8*  C5               411                   push    bc
82E9*  CD 8890          412                   call    WRITE_CHAR      ;write the character
82EC*  C1               413                   pop     bc
82ED*  21 0008          414                   ld      hl, 8           ;advance the bitmap pointer
82F0*  19               415                   add     hl, de
82F1*  EB               416                   ex      de, hl
```

```
E2F2'  21 3C00          417                     ld      hl, 3C00h
E2F5'  22 E890          418                     ld      (CHTBL), hl     ;restore (CHTBL) to original value
E2F8'  C1               419     de_exit         pop     bc
E2F9'  E1               420                     pop     hl
E2FA'  C9               421                     ret
                        422
                        423
                        424     ;***************************************************************************
                        425     ;
                        426     ; VISIBLE
                        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 Read: 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
                        449
E2F8'  3A 0FF9          450     visible         ld      a, (screen_ht)  ;is 0 <= row <= (screen_ht)-1
E2FB'  3D               451                     dec     a
E2FC'  B8               452                     cp      b
E2FD'  3F               453                     ccf
E2FE'  D0               454                     ret     nc              ;no, so return
E2FF'  3E 1F            455                     ld      a, max_cols-1   ;is 0 <= col <= max_cols-1
E301'  B9               456                     cp      c
E302'  3F               457                     ccf
E303'  C9               458                     ret                     ;return result of second test
                        459
                        460
                        461     ;***************************************************************************
                        462     ;
                        463     ; PUT_SPRITE
                        464     ;
                        465     ; Inputs: A = sprite-id
                        466     ;         D = row
                        467     ;         E = column
                        468     ;
                        469     ; Outputs: BC = 0000 - OK, nothing overwritten
                        470     ;             = 0100 - OK, something overwritten
                        471     ;             = 0003 - illegal sprite-id
                        472     ;
                        473     ; Globals Read: sprite data structure
                        474     ;               backgcolor
                        475     ;               gflags
                        476     ;               screen_ht
                        477     ;
                        478     ; Globals Written: sprite data structure
                        479     ;                  ATTR_P
                        480     ;
                        481     ; Procedures Called: check_id
                        482     ;                    spr_addr
                        483     ;                    SELECT
                        484     ;                    visible
                        485     ;                    GET_CHAR
                        486     ;                    display_char
                        487     ;
                        488     ; Procedures Called By: move_sprite
                        489     ;                       upd_cols
                        490     ;                       upd_rows
                        491     ;                       I_F_HANDLER
                        492     ;                       user program
                        493     ;
                        494     ; Description: this routine writes the specified sprite to the screen.
                        495     ;              If autowrap mode is set, then the specified row and column
                        496     ;              are wrapped to fit onto the screen size.  If autowrap
                        497     ;              mode is not set, then only the portion of the sprite
                        498     ;              which is visible on the screen will be written.
                        499     ;
                        500     ;***************************************************************************
                        501
                        502
E307'  F5               503     put_sprite      push    af
E308'  D5               504                     push    de
E309'  E5               505                     push    hl
E30A'  F3               506                     di
E30B'  CD E282'         507                     call    check_id        ;validate sprite-id
E30E'  38 06            508                     jr      c, p_id_ok      ;sprite-id ok, so jump
E310'  01 0003          509                     ld      bc, 0003        ;else return 03 in BC
E313'  C3 E398'         510                     jp      p_exit
E316'  CD E25F'         511     p_id_ok         call    spr_addr        ;get location of sprite record
E319'  3E 02            512                     ld      a, 2            ;select stream 2
E31B'  D5               513                     push    de
E31C'  CD 1230          514                     call    SELECT
E31F'  D1               515                     pop     de
E320'  DD 6E 05         516                     ld      l, (ix+pbitmap) ;hl -> sprite bitmap
E323'  DD 66 06         517                     ld      h, (ix+pbitmap+1)
E326'  42               518                     ld      b, d            ;b = row
E327'  48               519                     ld      c, e            ;c = col
E328'  DD 70 00         520                     ld      (ix+row), b     ;store sprite position
```

```
8328'  DD 71 01          921                 ld      (ix+col), c
832B'  3A DFF9           922                 ld      a, (backgcolor)
832E'  E6 F0             923                 and     11111000b
8330'  DD B6 04          924                 or      (ix+color)
8333'  32 9C80           925                 ld      (ATTR_P), a     ;set attr to screen background color
                         926                                         ;  and sprite foreground color
8336'  11 0100           927                 ld      de, 100h
8339'  A7                928                 and     a
833A'  ED 52             929                 sbc     hl, de
833C'  F8                930                 ex      de, hl          ;de -> first scan of sprite bitmap
                         931                                         ;  minus 100h, acts like (CHARS)
833D'  2E 00             932                 ld      l, 0            ;l = row count
833F'  C5                933                 push    bc
8340'  E3                934 outer1          ex      (sp), hl
8341'  40                935                 ld      c, l            ;c <- leftmost column of sprite
8342'  E3                936                 ex      (sp), hl
8343'  26 00             937                 ld      h, 0            ;h = column count
8345'  CD 82F8'          938 inner1          call    visible         ;is BC a legal screen location?
8348'  38 2D             939                 jr      c, nxtchar      ;yes, so jump
834A'  3A DFF8           940                 ld      a, (gflags)     ;is autowrap mode set?
834D'  CB 4F             941                 bit     autowrap, a
834F'  28 36             942                 jr      z, nxtchar      ;if not autowrap mode then jump
8351'  3E 1F             943 col_loop        ld      a, max_cols-1   ;keep current column onto screen
8353'  B9                944                 cp      c
8354'  30 06             945                 jr      nc, chrow1
8356'  79                946                 ld      a, c
8357'  D6 20             947                 sub     max_cols
8359'  4F                948                 ld      c, a
835A'  18 F5             949                 jr      col_loop
835C'  E5                950 chrow1          push    hl
835D'  21 DFF9           951                 ld      hl, screen_ht
8360'  7E                952                 ld      a, (hl)         ;keep current row onto screen
8361'  3D                953                 dec     a
8362'  B8                954 row_loop        cp      b
8363'  30 07             955                 jr      nc, cont1
8365'  08                956                 ex      af, af'
8366'  7E                957                 ld      a, b
8367'  96                958                 sub     (hl)
8368'  47                959                 ld      b, a
8369'  08                960                 ex      af,af'
836A'  18 F6             961                 jr      row_loop
836C'  E1                962 cont1           pop     hl
836D'  7C                963                 ld      a, h
836E'  B5                964                 or      l
836F'  20 06             965                 jr      nz, no_store
8371'  DD 70 00          966                 ld      (ix+row), b     ;first time through loops, store
8374'  DD 71 01          967                 ld      (ix+col), c     ;  wrapped screen location
8377'                    968 no_store
8377'  C5                969 nxtchar         push    bc
8378'  CD E91F           970                 call    GET_CHAR        ;check if writing the char will
837B'  C1                971                 pop     bc              ;  overwrite something
837C'  FE 20             972                 cp      ' '

                         973                 jr      z, no_overwrite
837E'  DD CB 07 CE       974                 set     overwrite, (ix+sflags)  ;save fact that something
                         975                                                 ;  was overwritten
8382'  CD 8208'          976 no_overwrite    call    display_char
8385'  0C                977 nxtchar         inc     c               ;increment column position
8386'  24                978                 inc     h               ;increment column count
8387'  DD 7E 02          979                 ld      a, (ix+width)
838A'  BC                980                 cp      h
838B'  20 06             981                 jr      nz, inner1      ;jump if more columns to write
838D'  04                982                 inc     b               ;else increment row position
838E'  2C                983                 inc     l               ;increment row count
838F'  DD 7E 03          984                 ld      a, (ix+height)
8392'  BD                985                 cp      l
8393'  20 A9             986                 jr      nz, outer1      ;jump if more rows to write
8395'  C1                987                 pop     bc
8396'  F8                988 e_exit          ex      af
8397'  E1                989                 pop     hl
8398'  D1                990                 pop     de
8399'  F1                991                 pop     af
839A'  DD CB 07 4E       992                 bit     overwrite, (ix+sflags)
839E'  DD CB 07 8E       993                 res     overwrite, (ix+sflags)
83A2'  28 04             994                 jr      nz, overwrote   ;sprite overwrote something
83A4'  01 0000           995                 ld      bc, 0           ;return 00 in BC, nothing overwritten
83A7'  C9                996                 ret
83A8'  01 0100           997 overwrote       ld      bc, 0100H
83AB'  C9                998                 ret
                         999
                        1000
                        1001 ;*****************************************************************************
                        1002 ;
                        1003 ; ERASE_SPRITE
                        1004 ;
                        1005 ; Inputs: A = sprite-id
                        1006 ;
                        1007 ; Outputs: BC = 0000 - OK
                        1008 ;              = 0003 - illegal sprite-id
                        1009 ;
                        1010 ; Globals Read: sprite data structure
                        1011 ;               backgcolor
                        1012 ;               gflags
                        1013 ;               screen_ht
                        1014 ;               CHARS
                        1015 ;
                        1016 ; Globals Written: sprite data structure
                        1017 ;                   ATTR_T
                        1018 ;                   CHARS
                        1019 ;
                        1020 ; Procedures Called: check_id
                        1021 ;                    spr_addr
                        1022 ;                    SELECT
                        1023 ;                    visible
                        1024 ;                    display_char
                        1025 ;
                        1026 ; Procedures Called By: move_sprite
                        1027 ;                       I_P_HANDLER
                        1028 ;                       user program
                        1029 ;
                        1030 ; Description: this routine erases the specified sprite.  The screen
                        1031 ;              locations at which the sprite resides are replaced with
                        1032 ;              blanks.
                        1033 ;
                        1034 ;*****************************************************************************
```

```
E381'  F5                  635
E382'  D5                  636
E383'  E5                  637  erase_sprite    push    af
E384'  F5                  638                  push    de
E385'  CD E282'            639                  push    hl
E388'  38 06               640                  ei
E38A'  01 0003             641                  call    check_id        ;validate sprite-id
E38D'  C3 E421'            642                  jr      c, e_id_ok      ;sprite-id ok, so jump
E3C0'  CD E25F'            643                  ld      bc, 0003        ;else return 03 in BC
E3C3'  3E 02               644                  jp      e_exit
E3C5'  D5                  645  e_id_ok         call    ser_addr        ;get location of sprite record
E3C6'  CC 1233             646                  ld      a, 2            ;select stream 2
E3C9'  01                  647                  push    de
E3CA'  DD 46 00            648                  call    $SELECT
E3CD'  DD 4E 01            649                  pop     de
E3D0'  3A DFF8             650                  ld      b, (iy+row)     ;b = row
E3D3'  32 9C8F             651                  ld      c, (iy+col)     ;c = col
E3D6'  28 08               652                  ld      a, (backgcolor)
E3D8'  C5                  653                  ld      (ATTR_T), a     ;set attr to screen background color
E3D9'  E3                  654                  ld      l, 8            ;l = row count
E3DA'  4C                  655                  push    bc
E3DB'  E3                  656  outer2          ex      (sp), hl
E3DC'  26 08               657                  ld      d, 1            ;d <- leftmost column of sprite
E3DE'  CD E2F8'            658                  ex      (sp), hl
E3E1'  38 23               659                  ld      h, 8            ;h = column count
E3E3'  3A DFF8             660  inner2          call    visible         ;is BC a legal screen location?
E3E6'  CB 6F               661                  jr      c, era_char     ;yes; so jump
E3E8'  28 23               662                  ld      a, (gflags)     ;is autowrap mode set?
E3EA'  3E 1F               663                  bit     autowrap, a
E3EC'  B9                  664                  jr      z, nxtchar      ;if not autowrap mode then jump
E3ED'  30 06               665  c_loop          ld      a, max_cols-1   ;keep current column onto screen
E3EF'  79                  666                  cp      c
E3F0'  D6 20               667                  jr      nc, cont2
E3F2'  4F                  668                  ld      a, c
E3F3'  18 F5               669                  sub     max_cols
E3F5'  E5                  670                  ld      c, a
E3F6'  21 DFF9             671                  jr      c_loop
E3F9'  7E                  672  chrow2          push    hl
E3FA'  3D                  673                  ld      hl, screen_ht
E3FB'  BE                  674                  ld      a, (hl)         ;keep current row onto screen
E3FC'  30 07               675                  dec     a
E3FE'  08                  676  r_loop          cp      b
E3FF'  78                  677                  jr      nc, cont2
E400'  96                  678                  ex      af, af'
E401'  47                  679                  ld      a, b
E402'  08                  680                  sub     (hl)
E403'  10 F6               681                  ld      b, a
E405'  E1                  682                  ex      af,af'
E406'  ED 58 5C36          683                  jr      r_loop
E40A'  CD E2D8'            684  cont2          pop     hl
E40D'  0C                  685  era_char        ld      de, (CHARS)     ;de -> bitmap of blank char
E40E'  24                  686                  call    display_char
E40F'  DD 7E 02            687  nxtchar         inc     c               ;increment column position
E412'  BC                  688                  inc     h               ;incremtnt column count
E413'  20 C9               689                  ld      a, (ix+width)
E415'  04                  690                  cp      h
E416'  2C                  691                  jr      nz, inner2      ;jump if more columns to write
E417'  DD 7E 03            692                  inc     b               ;else increment row position
E41A'  BD                  693                  inc     l               ;increment row count
E41B'  20 BC               694                  ld      a, (ix+height)
E41D'  C1                  695                  cp      l
E41E'  01 0000             696                  jr      nz, outer2      ;jump if more rows to write
E421'  F5                  697                  pop     bc              ;else done
E422'  E1                  698                  ld      bc, 0           ;return 00 in BC, nothing overwritten
E423'  D1                  699  e_exit          ei
E424'  F1                  700                  pop     hl
E425'  C9                  701                  pop     de
                           702                  pop     af
                           703                  ret
                           704
                           705
                           706  ;*****************************************************************************
                           707  ;
                           708  ; MOVE_SPRITE
                           709  ;
                           710  ; Inputs: A = sprite-id
                           711  ;         D = relative vertical motion
                           712  ;         E = relative horizontal motion
                           713  ;
                           714  ; Outputs: BC = 0000 - OK, nothing overwritten
                           715  ;              = 0100 - OK, something overwritten
                           716  ;              = 0003 - illegal sprite-id
                           717  ;
                           718  ; Globals Read: screen_ht
                           719  ;               gflags
                           720  ;
                           721  ; Globals Written: sprite data structure
                           722  ;
                           723  ; Procedures Called: erase_sprite
                           724  ;                    put_sprite
                           725  ;
                           726  ; Procedures Called By: I_F_HANDLER
                           727  ;                       user progrss
                           728  ;
                           729  ; Description: this routine moves the specified sprite by the specified
                           730  ;              amount.  It is equivalent to Erase_Sprite followed by
                           731  ;              Put_Sprite.
                           732  ;
                           733  ;*****************************************************************************
                           734
                           735
```

```
E426'  E5                  736  move_sprite    push    hl
E427'  F5                  737                 push    af
E428'  CD E381'            738                 call    erase_sprite
E428'  CB 41               739                 bit     0, c            ;if an error occurred, exit
E42D'  20 1E               740                 jr      nz, m_exit
E42F'  DD 7E 00            741                 ld      a, (ix+row)     ;update sprite position
E432'  82                  742                 add     a, d
E433'  21 DFF8             743                 ld      hl, gflags
E436'  CB 4E               744                 bit     outscreen, (hl)
E438'  28 08               745                 jr      z, row_ok
E43A'  CB 7F               746                 bit     7, a            ;is row negative?
E43C'  28 04               747                 jr      z, row_ok       ;no, so jump
E43E'  21 DFF9             748                 ld      hl, screen_ht   ;yes, so add it to (screen_ht)
E441'  86                  749                 add     a, (hl)
E442'  57                  750  row_ok         ld      d, a
E443'  DD 7E 01            751                 ld      a, (ix+col)
E446'  83                  752                 add     a, e
E447'  5F                  753                 ld      e, a
E448'  F1                  754                 pop     af
E449'  F5                  755                 push    af
E44A'  CD E307'            756                 call    put_sprite
E44D'  F1                  757  m_exit         pop     af
E44E'  E1                  758                 pop     hl
E44F'  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 program
                           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
E450'  00 E282'            789  ch_loc         call    check_id        ;validate sprite-id
E453'  38 04               790                 jr      c, ch_l_id_ok   ;if id is ok then jump
E455'  01 0003             791                 ld      bc, 0003        ;else return 03 in BC
E458'  C9                  792                 ret
E459'  CD E29F'            793  ch_l_id_ok     call    spr_addr        ;ix -> loc of sprite record
E45C'  DD 72 00            794                 ld      (ix+row), d
E45F'  DD 73 01            795                 ld      (ix+col), e
E462'  01 0000             796                 ld      bc, 00          ;return 00 in BC
E465'  C9                  797                 ret
                           798
                           799
                           800  ;*******************************************************************************
                           801  ;
                           802  ; CH_ATTR
                           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 program
                           820  ;
                           821  ; Description: this routine puts a new color into the record for the
                           822  ;              sprite identified by sprite-id.
                           823  ;
                           824  ;*******************************************************************************
                           825
                           826
E466'  CD E282'            827  ch_attr        call    check_id        ;validate sprite-id
E469'  38 03               828                 jr      c, ch_a_id_ok   ;if id is ok then jump
E46B'  01 0003             829                 ld      bc, 0003        ;else return 03 in BC
E46E'  F5                  830  ch_a_id_ok     push    af              ;validate color
E46F'  3E 07               831                 ld      a, 7
E471'  BA                  832                 cp      d
E472'  30 05               833                 jr      nc, ch_a_color_ok       ;jump if color ok
E474'  F1                  834                 pop     af
E475'  01 0005             835                 ld      bc, 0005        ;else return 05 in BC
E478'  C9                  836                 ret
E479'  F1                  837  ch_a_color_ok  pop     af
E47A'  CD E29F'            838                 call    spr_addr
E47D'  DD 72 04            839                 ld      (ix+color), d
E480'  01 0000             840                 ld      bc, 0           ;return 00 in BC
E483'  C9                  841                 ret
                           842
```

```
                        843
                        844   ;*****************************************************************************
                        845   ;
                        846   ; OVERLAP
                        847   ;
                        848   ; Inputs: D = sprite-id1
                        849   ;         E = sprite-id2
                        850   ;
                        851   ; Outputs: BC = 0000 - OK, no overlap
                        852   ;             = 0200 - OK, overlap
                        853   ;             = 0003 - illegal sprite-id
                        854   ;
                        855   ; Globals Read: sprite data structure
                        856   ;
                        857   ; Globals Written: none.
                        858   ;
                        859   ; Procedures Called: check_id
                        860   ;                    spr_addr
                        861   ;
                        862   ; Procedures Called By: upd_cols
                        863   ;                       upd_rows
                        864   ;                       I_P_HANDLER
                        865   ;                       user program
                        866   ;
                        867   ; Description: this routine checks if two sprites overlap on the screen.
                        868   ;              This is done by comparing the ranges covered by each sprite
                        869   ;              in the x- and y-directions.  If there is overlap in both
                        870   ;              directions, then the sprites must overlap.  The following
                        871   ;              diagram defines terms mentioned below:
                        872   ;
                        873   ;      (ulx1, uly1)
                        874   ;        +-------------------------+
                        875   ;        |                         |
                        876   ;        |    (ulx2, uly2)         |
                        877   ;        |      +------------------|--------------+
                        878   ;        |      |                  |              |
                        879   ;        |      |                  |              |
                        880   ;        |      |                  |              |
                        881   ;        |      |                  |              |
                        882   ;        +------|------------------+              |
                        883   ;               |                  (lrx1, lry1)  |
                        884   ;               |                                |
                        885   ;               |                                |
                        886   ;               |                                |
                        887   ;               +--------------------------------+
                        888   ;                                                (lrx2, lry2)
                        889   ;
                        890   ;*****************************************************************************
                        891
                        892
8484' DD E5             893   overlap:    push    ix
8486' F5                894               push    af
8487' D5                895               push    de
8488' E5                896               push    hl
8489' 7A                897               ld      a, d            ;validate sprite-id1
848A' CD 8282'          898               call    check_id
848D' 38 04             899               jr      c, o_ok1
848F' 01 0003           900   o_id_ng:    ld      bc, 0003        ;illegal sprite-id, return 3 in BC
8492' C9                901               ret
8493' 42                902   o_ok1:      ld      b, d            ;save sprite-ids in bc
8494' 48                903               ld      c, e
8495' 7B                904               ld      a, e            ;validate sprite-id2
8496' CD 8282'          905               call    check_id
8499' 38 F4             906               jr      nc, o_id_ng
                        907                                       ;here if both sprite-ids ok
849B' 78                908               ld      a, b            ;check if there is any overlap in
                        909                                       ;  the x-direction
849C' CD 829F'          910               call    spr_addr
849F' DD 56 01          911               ld      d, (ix+col)
84A2' DD 5E 02          912               ld      e, (ix+width)
84A5' 79                913               ld      a, c
84A6' CD 829F'          914               call    spr_addr
84A9' DD 66 01          915               ld      h, (ix+col)
84AC' DD 6E 02          916               ld      l, (ix+width)
84AF' 7A                917               ld      a, d
84B0' 83                918               add     a, e
84B1' 3D                919               dec     a               ;a = lrx1
84B2' BC                920               cp      h               ;h = ulx2
84B3' 38 28             921               jr      c, no_coll      ;if ulx2 > lrx1, then no overlap
84B5' 7C                922               ld      a, h
84B6' 85                923               add     a, l
84B7' 3D                924               dec     a               ;a = lrx2
84B8' BA                925               cp      d               ;d = ulx1
84B9' 38 25             926               jr      c, no_coll      ;if ulx1 > lrx2, then no overlap
84BB' 78                927               ld      a, b            ;check if there is any overlap in the
                        928                                       ;  y-direction
84BC' CD 829F'          929               call    spr_addr
84BF' DD 56 00          930               ld      d, (ix+row)
84C2' DD 5E 03          931               ld      e, (ix+height)
84C5' 79                932               ld      a, c
84C6' CD 829F'          933               call    spr_addr
84C9' DD 66 00          934               ld      h, (ix+row)
84CC' DD 6E 03          935               ld      l, (ix+height)
84CF' 7A                936               ld      a, d
8430' 83                937               add     a, e
8401' 3D                938               dec     a               ;a = lry1
8402' BC                939               cp      h               ;h = uly2
8403' 38 08             940               jr      c, no_coll      ;if lry1 < uly2, then no overlap
8405' 7C                941               ld      a, h
8406' 85                942               add     a, l
8407' 3D                943               dec     a               ;a = lry2
8408' BA                944               cp      d               ;e = uly1
8409' 38 05             945               jr      c, no_coll      ;if lry2 < uly1, then no overlap
840B' 01 0200           946   yes_coll:   ld      bc, 0200H       ;return OK, overlap
840E' 18 03             947               jr      o_exit
8410' 01 0000           948   no_coll:    ld      bc, 0           ;return OK, no overlap
8413' E1                949   o_exit:     pop     hl
8414' D1                950               pop     de
8415' F1                951               pop     af
8416' DD E1             952               pop     ix
8418' C9                953               ret
                        954
```

```
                        ;
                        ;**************************************************************************
                        ;
                        ; INITSCREEN
                        ;
                        ; Inputs: A = screen-height
                        ;         B = background-color
                        ;         C = border-color
                        ;
                        ; Outputs: BC = 0000 - OK
                        ;            = 0005 - illegal color
                        ;            = 0007 - illegal screen-height
                        ;
                        ; Globals Read: ATTR_P
                        ;               IVFLAG
                        ;               screen_ht
                        ;
                        ; Globals Written: screen_ht
                        ;                  ATTR_P
                        ;                  backgcolor
                        ;                  BORDCR
                        ;                  IVFLAG
                        ;
                        ; Procedures Called: CLS_B
                        ;
                        ; Procedures Called By: I_P_HANDLER
                        ;                       user program
                        ;
                        ; 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 permanent 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' 38 04                             jr      c, sh_ok        ;if screen-height ok, the 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' B8                bc_ok           cp      e               ;validate border-color
E500' 30 05                             jr      nc, bdc_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           bdc_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' 78                                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, IVFLAG
E530' 7E                                ld      a, (hl)         ;save (IVFLAG)
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 097F                           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 05                             jr      z, no_clear
E544' CB C6                             set     0, (hl)
E546' CD 097F                           call    CLS_B
E549' F1                no_clear        pop     af
E54A' 32 5C3C                           ld      (IVFLAG), a     ;restore (IVFLAG)
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
```

```
                        1036  ;**********************************************************************
                        1037  ;
                        1038  ; UPD_COLS
                        1039  ;
                        1040  ; Inputs: A = column adjustment (1 for right scroll, -1 for left scroll)
                        1041  ;
                        1042  ; Outputs: none.
                        1043  ;
                        1044  ; Globals Read: max_sprites
                        1045  ;               screen_ht
                        1046  ;               gflags
                        1047  ;
                        1048  ; Globals Written: max_sprites
                        1049  ;                  sprite data structure
                        1050  ;                  scratchpad
                        1051  ;
                        1052  ; Procedures Called: spr_addr
                        1053  ;                    overlap
                        1054  ;                    put_sprite
                        1055  ;
                        1056  ; Procedures Called By: h_scroll
                        1057  ;
                        1058  ; Description: this routine updates the column position of all sprites.
                        1059  ;              It also checks if, as a result of the horizontal scroll,
                        1060  ;              each sprite should be written to the screen.  This is done
                        1061  ;              by creating two dummy sprites (sprite-id is max_sprites)
                        1062  ;              whose size and location correspond to the right and left
                        1063  ;              columns of the screen respectively.  If a given sprite
                        1064  ;              overlaps either of these dummy sprites, it is written
                        1065  ;              to the screen.
                        1066  ;
                        1067  ;**********************************************************************
                        1068
                        1069
E584' DD E5             1070  upd_cols        push    ix
E586' C5                1071                  push    bc
E587' D5                1072                  push    de
E588' E5                1073                  push    hl
E589' 67                1074                  ld      h, a            ;save column adjustment in h
E58A' 3A 0FFA           1075                  ld      a, (max_sprites)
E58D' 3C                1076                  inc     a               ;temporarily increment max_sprites
E58E' 32 0FFA           1077                  ld      (max_sprites), a  ;  to allow for the dummy sprite
E591' 3D                1078                  dec     a
E592' 5F                1079                  ld      e, a            ;e = sprite-id of the dummy sprite
E593' DD 21 0FE3        1080                  ld      ix, scratchpad  ;initialize the dummy sprite
E597' CC 36 00 00       1081                  ld      (ix+row), 0
E59B' DD 36 07 01       1082                  ld      (ix+gflags), 1
E59F' DD 36 02 01       1083                  ld      (ix+width), 1
E5A3' F5                1084                  push    af
E5A4' 3A 0FF9           1085                  ld      a, (screen_ht)
E5A7' 3D                1086                  dec     a
E5A8' DD 77 03          1087                  ld      (ix+height), a
E5AB' F1                1088                  pop     af
E5AC' DD 36 01 1F       1089                  ld      (ix+col), max_cols-1
E5B0' 3C                1090  updc_loop       dec     a               ;a = id of current sprite
E5B1' DD E35F'          1091                  call    spr_addr        ;update column position
E5B4' 6F                1092                  ld      l, a
E5B5' DD 7E 01          1093                  ld      a, (ix+col)
E5B8' 84                1094                  add     a, h
E5B9' E5                1095                  push    hl
E5BA' 21 0FF8           1096                  ld      hl, gflags
E5BD' CB 4E             1097                  bit     autowrap, (hl)
E5BF' E1                1098                  pop     hl
E5C0' 28 0D             1099                  jr      z, uc_cont      ;if not autowrap mode, then jump
E5C2' FE 28             1100                  cp      max_cols
E5C4' 20 03             1101                  jr      nz, ucl         ;if column <> max_cols then jump
E5C6' AF                1102                  xor     a               ;else column <- 0
E5C7' 18 06             1103                  jr      uc_cont         ;continue
E5C9' FE FF             1104  ucl             cp      -1
E5CB' 20 02             1105                  jr      nz, uc_cont     ;if column <> -1, then jump
E5CD' 3E 1F             1106                  ld      a, max_cols-1   ;else column <- max_cols-1
E5CF' DD 77 01          1107  uc_cont         ld      (ix+col), a     ;don't need to wrap position as this
E5D2' 55                1108                  ld      d, l            ;  is done by put_sprite
E5D3' CD E484'          1109                  call    overlap
E5D6' CB 48             1110                  bit     1, b
E5D8' 78                1111                  ld      a, 1
E5D9' 28 13             1112                  jr      nz, prt_spr
E5DB' DD E5             1113                  push    ix
E5DD' DD 21 0FE3        1114                  ld      ix, scratchpad
E5E1' DC 36 01 00       1115                  ld      (ix+col), 0
E5E5' DC E1             1116                  pop     ix
E5E7' CC E484'          1117                  call    overlap
E5EA' CB 48             1118                  bit     1, b
E5EC' 28 08             1119                  jr      z, next_spr     ;if no overlap, go on to the next one
E5EE' 68                1120  prt_spr         ld      l, e
E5EF' DD 56 00          1121                  ld      d, (ix+row)
E5F2' DD 5E 01          1122                  ld      e, (ix+col)
E5F5' CC E307'          1123                  call    put_sprite      ;overlaps, so write sprite to screen
E5F8' 93                1124                  ld      e, l
E5F9' A7                1125  next_spr        and     a
E5FA' 20 B4             1126                  jr      nz, updc_loop
E5FC' 21 0FFA           1127                  ld      hl, max_sprites
E5FF' 35                1128                  dec     (hl)            ;restore max_sprites to its original
                        1129                                          ;  value
E600' E1                1130                  pop     hl
E601' D1                1131                  pop     de
E602' C1                1132                  pop     bc
E603' DD E1             1133                  pop     ix
E605' C9                1134                  ret
                        1135
                        1136
```

```
                        1157  ;*****************************************************************
                        1158  ;
                        1159  ; H_SCROLL
                        1160  ;
                        1161  ; Inputs: A < 0 - left scroll
                        1162  ;         A >= 0 - right scroll
                        1163  ;
                        1164  ; Outputs: BC = 0000 - OK
                        1165  ;
                        1166  ; Globals Read: screen_ht
                        1167  ;               backgcolor
                        1168  ;
                        1169  ; Globals Written: none.
                        1170  ;
                        1171  ; Procedures Called: used_colo
                        1172  ;
                        1173  ; Procedures Called By: I_P_HANDLER
                        1174  ;                       user program
                        1175  ;
                        1176  ; Description: this routine causes the contents of the top (screen_ht)
                        1177  ;              lines of the screen to be scrolled left or right one
                        1178  ;              column, depending on the value of A.  All sprites have
                        1179  ;              their column position updated by 1 or -1, depending on
                        1180  ;              the direction of scroll.
                        1181  ;
                        1182  ;*****************************************************************
                        1183
                        1184
E5D6' F5                1185  h_scroll        push    af
E5D7' D5                1186                  push    de
E5D8' E5                1187                  push    hl
E5D9' D9                1188                  exx
E5DA' C5                1189                  push    bc
E5DB' D5                1190                  push    de
E5DC' E5                1191                  push    hl
E5DD' D9                1192                  exx
E5DE' F3                1193                  di
E5DF' 47                1194                  ld      b, a            ;save direction
E5E0' 3A DFF9           1195                  ld      a, (screen_ht)  ;calc. the number of scan lines
E5E3' CB 27             1196                  sla     a               ;  to be updated
E5E5' CB 27             1197                  sla     a
E5E7' CB 27             1198                  sla     a
E5E9' CB 78             1199                  bit     7, b
E5EB' 28 43             1200                  jr      z, right_scroll ;if A >=0, then jump
E5ED' D9                1201  left_scroll     exx                     ;use alt. regs. for attr pointers
E5EE' 21 5801           1202                  ld      hl, 5801H
E5F1' 11 5800           1203                  ld      de, 5800H
E5F4' D9                1204                  exx
E5F5' 3A DFF9           1205                  ld      a, (screen_ht)
E5F8' 47                1206                  ld      b, a            ;b = line counter
E5F9' 21 E686'          1207                  ld      hl, left_table
E5FC' E5                1208                  push    hl
E5FD' E1                1209  left_outer      pop     hl
E5FE' 5E                1210                  ld      e, (hl)
E5FF' 23                1211                  inc     hl
E600' 56                1212                  ld      d, (hl)         ;de -> dest for next line
E601' 23                1213                  inc     hl
E602' E5                1214                  push    hl
E603' 62                1215                  ld      h, d
E604' 68                1216                  ld      l, e
E605' 23                1217                  inc     hl              ;hl -> source for next line
E606' 3E 08             1218                  ld      a, 8            ;a = scan counter
E608' C5                1219                  push    bc
E609' 01 001F           1220  left_inner      ld      bc, max_cols-1
E60C' ED B0             1221                  ldir                    ;move scan left
E60E' EB                1222                  ex      de, hl
E60F' 36 00             1223                  ld      (hl), 0         ;clear last column
E611' 01 00E2           1224                  ld      bc, 256-(max_cols-2)
E614' 09                1225                  add     hl, bc          ;hl -> start of next scan + one byte
E615' 54                1226                  ld      d, h
E616' 50                1227                  ld      e, l
E617' 1B                1228                  dec     de              ;de -> start of next scan
E618' 3D                1229                  dec     a
E619' 20 EE             1230                  jr      nz, left_inner  ;more scans to do, so jump
E61B' D9                1231                  exx
E61C' 01 001F           1232                  ld      bc, max_cols-1
E61F' ED B0             1233                  ldir                    ;move attributes
E621' 3A DFF8           1234                  ld      a, (backgcolor) ;clear rightmost column to permanent
E624' 12                1235                  ld      (de), a         ;  attribute
E625' 23                1236                  inc     hl              ;hl -> second byte of next line
E626' 13                1237                  inc     de              ;de -> first byte of next line
E627' D9                1238                  exx
E628' C1                1239                  pop     bc
E629' 10 D2             1240                  djnz    left_outer      ;if more lines to do, then jump
E62B' E1                1241                  pop     hl
E62C' 3E FF             1242                  ld      a, -1           ;set parameter for used_colo
E62E' 18 46             1243                  jr      hs_cont
E630' D9                1244  right_scroll    exx                     ;use alt. regs. for attr pointers
E631' 21 581E           1245                  ld      hl, 581EH
E634' 11 581F           1246                  ld      de, 581FH
E637' D9                1247                  exx
E638' 3A DFF9           1248                  ld      a, (screen_ht)
E63B' 47                1249                  ld      b, a            ;b = line counter
E63C' 21 E686'          1250                  ld      hl, rt_table
E63F' E5                1251                  push    hl
E640' E1                1252  rt_outer        pop     hl
E641' 5E                1253                  ld      e, (hl)
E642' 23                1254                  inc     hl
E643' 56                1255                  ld      d, (hl)         ;de -> dest for next line
E644' 23                1256                  inc     hl
E645' E5                1257                  push    hl
E646' 62                1258                  ld      h, d
E647' 68                1259                  ld      l, e
E648' 2B                1260                  dec     hl              ;hl -> source for next line
E649' 3E 08             1261                  ld      a, 8            ;a = scan counter
E64B' C5                1262                  push    bc
E64C' 01 001F           1263  rt_inner        ld      bc, max_cols-1
E64F' ED B8             1264                  lddr                    ;move scan right
```

```
E651'  E5              1265                    ex      de, hl
E652'  36 00           1266                    ld      (hl), 0         ;clear first column
E654'  01 011E         1267                    ld      bc, 256-(max_cols-2)
E657'  09              1268                    add     hl, bc          ;hl -> end of next scan - 1 byte
E658'  54              1269                    ld      d, h
E659'  5D              1270                    ld      e, l
E65A'  13              1271                    inc     de              ;de -> end of next scan
E65B'  3D              1272                    dec     a
E65C'  20 EE           1273                    jr      nz, rt_inner    ;more scans to do, so jump
E65E'  D9              1274                    exx
E65F'  01 001F         1275                    ld      bc, 31
E662'  ED B8           1276                    lddr                    ;move attributes
E664'  3A 0FF8         1277                    ld      a, (backgcolor) ;clear leftmost column to permanent
E667'  12              1278                    ld      (de), a         ;  attribute
E668'  01 003F         1279                    ld      bc, 2*max_cols-1
E66B'  09              1280                    add     hl, bc          ;hl -> second to last byte of next line
E66C'  54              1281                    ld      d, h
E66D'  5D              1282                    ld      e, l
E66E'  13              1283                    inc     de              ;de -> last byte of next line
E66F'  D9              1284                    exx
E670'  C1              1285                    pop     bc
E671'  10 CD           1286                    djnz    rt_outer        ;more lines to do, so jump
E673'  E1              1287                    pop     hl
E674'  3E 01           1288                    ld      a, 1            ;set parameter for upd_cols
E676'  CD E554'        1289    hs_cont         call    upd_cols
E679'  FB              1290                    ei
E67A'  D9              1291                    exx
E67B'  E1              1292                    pop     hl
E67C'  D1              1293                    pop     de
E67D'  C1              1294                    pop     bc
E67E'  D9              1295                    exx
E67F'  E1              1296                    pop     hl
E680'  D1              1297                    pop     de
E681'  F1              1298                    pop     af
E682'  01 0000         1299                    ld      bc, 0
E685'  C9              1300                    ret
                       1301
                       1302
                       1303    ;This table contains the destination address for each row for scrolling left
                       1304
E686'  4000            1305    left_table      defw    4000H
E688'  4020            1306                    defw    4020H
E68A'  4040            1307                    defw    4040H
E68C'  4060            1308                    defw    4060H
E68E'  4080            1309                    defw    4080H
E690'  40A0            1310                    defw    40A0H
E692'  40C0            1311                    defw    40C0H
E694'  40E0            1312                    defw    40E0H
E696'  4800            1313                    defw    4800H
E698'  4820            1314                    defw    4820H
E69A'  4840            1315                    defw    4840H
E69C'  4860            1316                    defw    4860H
E69E'  4880            1317                    defw    4880H
E6A0'  48A0            1318                    defw    48A0H
E6A2'  48C0            1319                    defw    48C0H
E6A4'  48E0            1320                    defw    48E0H
E6A6'  5000            1321                    defw    5000H
E6A8'  5020            1322                    defw    5020H
E6AA'  5040            1323                    defw    5040H
E6AC'  5060            1324                    defw    5060H
E6AE'  5080            1325                    defw    5080H
E6B0'  50A0            1326                    defw    50A0H
E6B2'  50C0            1327                    defw    50C0H
E6B4'  50E0            1328                    defw    50E0H
                       1329
                       1330
                       1331    ;This table contains the destination address for each row for scrolling right
                       1332
E6B6'  401F            1333    rt_table        defw    401FH
E6B8'  403F            1334                    defw    403FH
E6BA'  405F            1335                    defw    405FH
E6BC'  407F            1336                    defw    407FH
E6BE'  409F            1337                    defw    409FH
E6C0'  40BF            1338                    defw    40BFH
E6C2'  40DF            1339                    defw    40DFH
E6C4'  40FF            1340                    defw    40FFH
E6C6'  481F            1341                    defw    481FH
E6C8'  483F            1342                    defw    483FH
E6CA'  485F            1343                    defw    485FH
E6CC'  487F            1344                    defw    487FH
E6CE'  489F            1345                    defw    489FH
E6D0'  48BF            1346                    defw    48BFH
E6D2'  48DF            1347                    defw    48DFH
E6D4'  48FF            1348                    defw    48FFH
E6D6'  501F            1349                    defw    501FH
E6D8'  503F            1350                    defw    503FH
E6DA'  505F            1351                    defw    505FH
E6DC'  507F            1352                    defw    507FH
E6DE'  509F            1353                    defw    509FH
E6E0'  50BF            1354                    defw    50BFH
E6E2'  50DF            1355                    defw    50DFH
E6E4'  50FF            1356                    defw    50FFH
                       1357
                       1358
```

```
                        1359  ;*****************************************************************
                        1360  ;
                        1361  ; UPD_ROWS
                        1362  ;
                        1363  ; Inputs: A = row adjustment (1 for scroll down, -1 for scroll up)
                        1364  ;
                        1365  ; Outputs: none.
                        1366  ;
                        1367  ; Globals Read: max_sprites
                        1368  ;               screen_ht
                        1369  ;               gflags
                        1370  ;
                        1371  ; Globals written: max_sprites
                        1372  ;                  sprite data structure
                        1373  ;                  scratchpad
                        1374  ;
                        1375  ; Procedures Called: spr_addr
                        1376  ;                    overlap
                        1377  ;                    out_sprite
                        1378  ;
                        1379  ; Procedures Called By: v_scroll
                        1380  ;
                        1381  ; Description: this routine updates the row position of all sprites.
                        1382  ;              It also checks if, as a result of the vertical scroll,
                        1383  ;              each sprite should be written to the screen.  This is done
                        1384  ;              by creating two dummy sprites (sprite-id is max_sprites)
                        1385  ;              whose size and location corresponds to the top and bottom
                        1386  ;              rows of the screen respectively.  If a given sprite
                        1387  ;              overlaps either of these dummy sprites, it is written
                        1388  ;              to the screen.
                        1389  ;
                        1390  ;*****************************************************************
                        1391
                        1392
E686' DD E5             1393  upd_rows        push    ix
E688' C5                1394                  push    bc
E689' D5                1395                  push    de
E68A' E5                1396                  push    hl
E68B' 67                1397                  ld      h, a            ;save row adjustment in h
E68C' 3A DFFA           1398                  ld      a, (max_sprites)
E68F' 3C                1399                  inc     a               ;temporarily increment max_sprites
E690' 32 DFFA           1400                  ld      (max_sprites), a  ; to allow for the dummy sprite
E693' 3C                1401                  inc     a
E694' 5F                1402                  ld      e, a            ;e = sprite-id of the dummy sprite
E695' DD 21 DFE3        1403                  ld      ix, scratchpad  ;initialize the dummy sprite
E699' DD 36 01 00       1404                  ld      (ix+col), 0
E69D' DD 36 07 01       1405                  ld      (ix+gflags), 1
E6A1' DD 36 02 20       1406                  ld      (ix+width), max_cols
E6A5' DD 36 03 01       1407                  ld      (ix+height), 1
E6A9' F5                1408                  push    af
E6AA' 3A DFF9           1409                  ld      a, (screen_ht)
E6AD' 3D                1410                  dec     a
E6AE' DD 77 00          1411                  ld      (ix+row), a
E6B1' F1                1412                  pop     af
E6B2' 3C                1413  updr_loop       inc     a               ;a = id of current sprite
E6B3' CD E25F'          1414                  call    spr_addr        ;update column position
E6B6' 6F                1415                  ld      l, a
E6B7' DD 7E 00          1416                  ld      a, (ix+row)
E6BA' 84                1417                  add     a, h
E6BB' E5                1418                  push    hl
E6BC' 21 DFF9           1419                  ld      hl, gflags
E6BF' CB 4E             1420                  bit     autowrap, (hl)
E6C1' E1                1421                  pop     hl
E6C2' 28 16             1422                  jr      z, wr_cont      ;if not autowrap mode, then jump
E6C4' E5                1423                  push    hl
E6C5' 21 DFF9           1424                  ld      hl, screen_ht
E6C8' BE                1425                  cp      (hl)
E6C9' E1                1426                  pop     hl
E6CA' 20 03             1427                  jr      nz, wr1         ;if row <> screen_ht then jump
E6CC' AF                1428                  xor     a               ;else row <- 0
E6CD' 18 08             1429                  jr      wr_cont         ;continue
E6CF' FE FF             1430  wr1             cp      -1
E6D1' 20 07             1431                  jr      nz, wr_cont     ;if row <> -1, then jump
E6D3' E5                1432                  push    hl
E6D4' 21 DFF9           1433                  ld      hl, screen_ht
E6D7' 7E                1434                  ld      a, (hl)         ;else row <- screen_ht-1
E6D8' 3D                1435                  dec     a
E6D9' E1                1436                  pop     hl
E6DA' DD 77 00          1437  wr_cont         ld      (ix+row), a
E6DD' 55                1438                  ld      d, l
E6DE' CD E484'          1439                  call    overlap
E6E1' CB 46             1440                  bit     0, b
E6E3' 7D                1441                  ld      a, l
E6E4' 20 13             1442                  jr      nz, ot_spr
E6E6' DD E5             1443                  push    ix
E6E8' DD 21 DFE3        1444                  ld      ix, scratchpad
E6EC' DD 36 00 00       1445                  ld      (ix+row), 0
E6F0' DD E1             1446                  pop     ix
E6F2' CD E484'          1447                  call    overlap
E6F5' CB 46             1448                  bit     0, b
E6F7' 28 06             1449                  jr      z, ns_spr       ;if no overlap, go on to the next one
E6F9' 4B                1450  ot_spr          ld      l, e
E6FA' DD 56 00          1451                  ld      d, (ix+row)
E6FD' DD 5E 01          1452                  ld      e, (ix+col)
E700' CD E307'          1453                  call    out_sprite      ;overlaps, so write sprite to screen
E703' 5D                1454                  ld      e, l
E704' A7                1455  ns_spr          and     a
E705' 20 AB             1456                  jr      nz, updr_loop
E707' 21 DFFA           1457                  ld      hl, max_sprites
E70A' 35                1458                  dec     (hl)            ;restore max_sprites to its original
                        1459                                          ;  value
E70B' E1                1460                  pop     hl
E70C' D1                1461                  pop     de
E70D' C1                1462                  pop     bc
E70E' DD E1             1463                  pop     ix
E710' C9                1464                  ret
                        1465
```

```
                        1466
                        1467    ;*****************************************************************************
                        1468    ;
                        1469    ; V_SCROLL
                        1470    ;
                        1471    ; Inputs: A < 0 - left scroll
                        1472    ;         A >= 0 - right scroll
                        1473    ;
                        1474    ; Outputs: BC = 0000 - OK
                        1475    ;
                        1476    ; Globals Read: screen_ht
                        1477    ;               backgcolor
                        1478    ;
                        1479    ; Globals Written: ATTR_T
                        1480    ;
                        1481    ; Procedures Called: SET_PRINT_POS
                        1482    ;                    WRITE_CHAR
                        1483    ;
                        1484    ; Procedures Called By: I_P_HANDLER
                        1485    ;                       user program
                        1486    ;
                        1487    ; Description: this routine causes the contents of the top (screen_ht)
                        1488    ;              lines of the screen to be scrolled left or right one
                        1489    ;              column, depending on the value of A.  All sprites have
                        1490    ;              their column position updated by 1 or -1, depending on
                        1491    ;              the direction of scroll.
                        1492    ;
                        1493    ;*****************************************************************************
                        1494
                        1495
E771' F5                1496    v_scroll        push    af
E772' D5                1497                    push    de
E773' E5                1498                    push    hl
E774' D9                1499                    exx
E775' C5                1500                    push    bc
E776' D5                1501                    push    de
E777' E5                1502                    push    hl
E778' D9                1503                    exx
E779' F3                1504                    di
E77A' CB 7F             1505                    bit     7, a
E77C' 28 59             1506                    jr      z, down_scroll  ;if A >=0, then jump
E77E' D9                1507    up_scroll       exx                     ;use alt. regs. for attr pointers
E77F' 21 5820           1508                    ld      hl, 5820H
E782' 11 5800           1509                    ld      de, 5800H
E785' D9                1510                    exx
E786' 3A DFF9           1511                    ld      a, (screen_ht)
E789' 3C                1512                    inc     a
E78A' 47                1513                    ld      b, a            ;b = line counter
E78B' 21 E960'          1514                    ld      hl, vs_table
E78E' E5                1515                    push    hl
E78F' E1                1516    up_outer        pop     hl
E790' 5E                1517                    ld      e, (hl)
E791' 23                1518                    inc     hl
E792' 56                1519                    ld      d, (hl)         ;de -> dest for next line
E793' 23                1520                    inc     hl
E794' D5                1521                    push    de              ;put dest pointer on stack
E795' 5E                1522                    ld      e, (hl)
E796' 23                1523                    inc     hl
E797' 56                1524                    ld      d, (hl)         ;de -> source for next line
E798' 2B                1525                    dec     hl
E799' F3                1526                    ex      (sp), hl        ;exchange dest pointer and tabl
                        1527                                            ;  pointer
E79A' EB                1528                    ex      de, hl          ;put pointers in proper regs
E79B' 3E 08             1529                    ld      a, 8            ;a = scan counter
E79D' C5                1530                    push    bc
E79E' 01 0020           1531    up_inner        ld      bc, max_cols
E7A1' ED B0             1532                    ldir                    ;move scan up
E7A3' 01 00E0           1533                    ld      bc, 256-max_cols
E7A6' 09                1534                    add     hl, bc          ;hl -> start of next scan
E7A7' EB                1535                    ex      de, hl
E7A8' 09                1536                    add     hl, bc
E7A9' EB                1537                    ex      de, hl          ;de -> start of next scan dest
E7AA' 3D                1538                    dec     a
E7AB' 20 F1             1539                    jr      nz, up_inner    ;more scans to do, so jump
E7AD' D9                1540                    exx
E7AE' 01 0020           1541                    ld      bc, max_cols
E7B1' ED B0             1542                    ldir                    ;move attributes
E7B3' D9                1543                    exx
E7B4' C1                1544                    pop     bc
E7B5' 10 D8             1545                    djnz    up_outer        ;if more lines to do, then jump
E7B7' E1                1546                    pop     hl
E7B8' 3A DFF9           1547                    ld      a, (screen_ht)  ;clear the bottom line
E7BB' 3C                1548                    inc     a
E7BC' 47                1549                    ld      b, a
E7BD' 0E 00             1550                    ld      c, 0
E7BF' CD E90E           1551                    call    SET_PRINT_POS
E7C2' 06 20             1552                    ld      b, max_cols
E7C4' 3A DFFB           1553                    ld      a, (backgcolor) ;set background color to permanent
                        1554                                            ;  value
E7C7' 32 5C8F           1555                    ld      (ATTR_T), a
E7CA' C5                1556    cl_bot_line     push    bc
E7CB' 3E 20             1557                    ld      a, ' '
E7CD' CC E890           1558                    call    WRITE_CHAR
E7D0' C1                1559                    pop     bc
E7D1' 10 F7             1560                    djnz    cl_bot_line
E7D3' 3E FF             1561                    ld      a, -1           ;set parameter for upd_cols
E7D5' 18 79             1562                    jr      vs_cont
E7D7' 3A DFF9           1563    down_scroll     ld      a, (screen_ht)  ;set up table pointer
E7DA' 3C                1564                    inc     a
E7DB' F5                1565                    push    af
E7DC' CB 27             1566                    sla     a
E7DE' 4F                1567                    ld      c, a
E7DF' 06 00             1568                    ld      b, 0
E7E1' 21 E860'          1569                    ld      hl, vs_table
E7E4' 09                1570                    add     hl, bc          ;hl -> entry for last line
E7E5' E5                1571                    push    hl
E7E6' E5                1572                    push    hl
E7E7' D9                1573                    exx                     ;use alt. regs. for attr pointers
```

```
E7E8'  E1                  1574                    pop     hl
E7E9'  5E                  1575                    ld      e, (hl)
E7EA'  23                  1576                    inc     hl
E7EB'  56                  1577                    ld      d, (hl)
E7EC'  EB                  1578                    ex      de, hl          ;hl -> 1st scan of last line
E7ED'  7C                  1579                    ld      a, h            ;calc. address of attr for last line
E7EE'  0F                  1580                    rrca                    ;  as done in ATTBYT in Home ROM
E7EF'  0F                  1581                    rrca
E7F0'  0F                  1582                    rrca
E7F1'  E6 03               1583                    and     000000118
E7F3'  F6 58               1584                    or      010110008
E7F5'  67                  1585                    ld      h, a
E7F6'  94                  1586                    ld      d, h            ;de -> 1st attr byte of last line
E7F7'  5D                  1587                    ld      e, l
E7F8'  01 0020             1588                    ld      bc, max_cols
E7FB'  A7                  1589                    and     a
E7FC'  ED 42               1590                    sbc     hl, bc          ;hl -> 1st attr byte of next to last
E7FE'  D9                  1591                    exx                     ;  line
E7FF'  E1                  1592                    pop     hl
E800'  F1                  1593                    pop     af
E801'  23                  1594                    inc     hl
E802'  E5                  1595                    push    hl
E803'  47                  1596                    ld      b, a            ;b = line counter
E804'  E1                  1597  down_outer       pop     hl
E805'  56                  1598                    ld      d, (hl)
E806'  2B                  1599                    dec     hl
E807'  5E                  1600                    ld      e, (hl)         ;de -> dest for next line
E808'  2B                  1601                    dec     hl
E809'  D5                  1602                    push    de              ;put dest pointer on stack
E80A'  56                  1603                    ld      d, (hl)
E80B'  2B                  1604                    dec     hl
E80C'  5E                  1605                    ld      e, (hl)         ;de -> source for next line
E80D'  23                  1606                    inc     hl
E80E'  E3                  1607                    ex      (sp), hl        ;exchange dest pointer and table
E80F'  EB                  1608                                            ;  pointer
E810'  3E 08               1609                    ex      de, hl          ;put pointers in proper regs
E812'  C5                  1610                    ld      a, 8            ;a = scan counter
E813'  01 0020             1611                    push    bc
E816'  ED B0               1612  down_inner       ld      bc, max_cols
E818'  01 0CE0             1613                    ldir                    ;move scan down
E81B'  09                  1614                    ld      bc, 256-max_cols
E81C'  EB                  1615                    add     hl, bc          ;hl -> start of next scan
E81D'  09                  1616                    ex      de, hl
E81E'  EB                  1617                    add     hl, bc
E81F'  3D                  1618                    ex      de, hl          ;de -> start of next scan dest
E820'  20 F1               1619                    dec     a
E822'  D9                  1620                    jr      nz, down_inner  ;more scans to do, so jump
E823'  01 0020             1621                    exx
E826'  ED B0               1622                    ld      bc, max_cols
E828'  01 0040             1623                    ldir                    ;move attributes
E82B'  A7                  1624                    ld      bc, 2*max_cols
E82C'  ED 42               1625                    and     a
E82E'  EB                  1626                    sbc     hl, bc
E82F'  ED 42               1627                    ex      de, hl
E831'  EB                  1628                    sbc     hl, bc
E832'  D9                  1629                    ex      de, hl
E833'  C1                  1630                    exx
E834'  10 CE               1631                    pop     bc
E836'  E1                  1632                    djnz    down_outer      ;if more lines to do, then jump
E837'  01 0000             1633                    pop     hl
E83A'  CD E90E             1634                    ld      bc, 0           ;clear top line
E83D'  06 20               1635                    call    SET_PRINT_POS
E83F'  3A DFF8             1636                    ld      b, max_cols
E842'  32 9C8F             1637                    ld      a, (backgcolor) ;set background color to permanent
E845'  C5                  1638                                            ;  value
E846'  3E 20               1639                    ld      (ATTR_T), a
E848'  CD E89D             1640  cl_top_line      push    bc
E84B'  C1                  1641                    ld      a, ' '
E84C'  10 F7               1642                    call    WRITE_CHAR
E84E'  3E 01               1643                    pop     bc
E850'  CB E6E6'            1644                    djnz    cl_top_line
E853'  F8                  1645                    ld      a, 1            ;set parameter for usd_cols
E854'  D9                  1646  vs_cont          call    usd_rows
E855'  E1                  1647                    ei
E856'  D1                  1648                    exx
E857'  C1                  1649                    pop     hl
E858'  D9                  1650                    pop     de
E859'  E1                  1651                    pop     bc
E85A'  D1                  1652                    exx
E85B'  F1                  1653                    pop     hl
E85C'  01 0000             1654                    pop     de
E85F'  C9                  1655                    pop     af
                           1656                    ld      bc, 0
                           1657                    ret
                           1658
                           1659
                           1660  ;This table contains the starting address of each row for use in vertical
                           1661  ;  scrolling
                           1662
E860'  4000                1663  vs_table         defw    4000H
E862'  4020                1664                  defw    4020H
E864'  4040                1665                  defw    4040H
E866'  4060                1666                  defw    4060H
E868'  4080                1667                  defw    4080H
E86A'  40A0                1668                  defw    40A0H
E86C'  40C0                1669                  defw    40C0H
E86E'  40E0                1670                  defw    40E0H
E870'  4800                1671                  defw    4800H
E872'  4820                1672                  defw    4820H
E874'  4840                1673                  defw    4840H
E876'  4860                1674                  defw    4860H
E878'  4880                1675                  defw    4880H
E87A'  48A0                1676                  defw    48A0H
E87C'  48C0                1677                  defw    48C0H
E87E'  48E0                1678                  defw    48E0H
E880'  5000                1679                  defw    5000H
E882'  5020                1680                  defw    5020H
E884'  5040                1681                  defw    5040H
E886'  5060                1682                  defw    5060H
E888'  5080                1683                  defw    5080H
E88A'  50A0                1684                  defw    50A0H
E88C'  50C0                1685                  defw    50C0H
E88E'  50E0                1686                  defw    50E0H
                           1687                  end
                           1688
                           1689
```

```
SPRSVC  CRI80/11 version 10.36.14       15-May-84  9:51:11
Sprite services module   SPRSVC.SRC

A       Reserved        ALLOC   0000            ATTRSP  SC8D            ATTRST  SC8F            AUTOWR  0001
B       Reserved        BACKGC  0FF8            BC90K   E4FF            BDC90K  E507            BLACK8  E528
BORDCR  9C48            SCRDPY  00FE            C       Reserved        CHARS   9C36            CHECK8  E282
CHTBL   E890            CH9ATT  E446            CH9A9C  E479            CH9A9I  E46E            CH9LCC  E450
CH9L9I  E459            CHROW1  E35F            CHROW2  E3F5            CLEAR8  E256            CLS88   097F
CL98DT  E7CA            CL9TOP  E845            CCL     0001            CCLOR   0004            COLOR8  E28F
COL9LO  E356            COMMAN  0FF7            CONT1   E36F            CONT2   E405            CREATE  E278
CRLOOP  E3EA            D       Reserved        DC9EXI  E2F8            DISPLA  E2D8            DOWN9I  E413
DOWN9O  E804            DOWN8S  E7D7            E       Reserved        ENOUGH  E253            ERASE8  E381
ERA8CH  E406            E9EXIT  E421            E9IO9O  E3C0            GET9CH  E91F            GPLAGS  0FF8
GLOBAL  001D            H       Reserved        HEIGHT  0003            HS9CON  E676            HS9CRO  E5D6
IO90K   E284            INITSC  E4E9            INITS$  E219            INNER1  E348            INNER2  E3DE
L       Reserved        LEFT9I  E609            LEFT9O  E5F0            LEFT9S  E5E0            LEFT9T  E686
LMS     0002            M       Reserved        MAX9CO  0020            MAX9RO  0018            MAX9SP  0FFA
MOVE9S  E426            M9EXIT  E44D            NAME    0FF0            NEG     0000            NEXTCH  E38A
NEXT9S  E5C9            NO9CLE  E549            NO9CCL  E4E8            NO9CVE  E387            NO9STO  E37A
NUM9PA  0FEC            NXTCHA  E40C            NX9SPR  E764            OFF     E2D4            OUTER1  E343
OUTER2  E3D9            OVERLA  E484            OVERWR  0001            OVRWRO  E3AD            O9EXIT  E4E3
O9IO9N  E48F            O9OK1   E493            PARM3   0FF3            PARM4   0FF4            PARM1   0FE0            PARM2   0FEF
PARM3   0FF1            PARM4   0FF3            PARM5   0FF5            PBITMA  0005            PSW     Reserved
PUT9SP  E307            P9EXIT  E399            P9IO9O  E316            RAMTOP  9C92            RIGHT8  E430
ROW     0000            ROW9LO  E365            ROW9OK  E442            RT9INN  E64C            RT9OUT  E640
RT9TAB  E686            R9LOOP  E3F9            SA9EXI  E2D6            SCRATC  0FE3            SCREEN  0FF9
SELECT  1230            SENDTV  0500            SET9AT  0582            SET9AU  E2C9            SET9PR  E90E
SFLAGS  0007            SH9OK   E4F1            SP      Reserved        SPRCOD  E000            SPRITE  0FFE
SPR9AD  E25F            SVC9CO  0FE8            TVFLAG  9C3C            UC9CON  E59F            UC1     E599
UPDC9L  E580            UPOR9L  E712            UP39CO  E554            UPD9RO  E6E6            UP9INN  E79E
UP9OUT  E78F            UP9SCR  E77E            UR9CON  E73A            URI     E72F            VIS18L  E2F8
VS9CON  E850            VS9TAB  E960            V89CRO  E771            WIDTH   0002            WRITE8  E890
WRTCHA  E37A            WRT9SP  E58E            WT9SPR  E759            Y       9C3A            YES9CO  E408

     No errors detected
```

```
                                1                       %NAME  IOLOW
                                2                       SUBTTL Low level I/O module
                                3
                                4
                                5   !*******************************************************************************
                                6   !
                                7   ! Low Level I/O Module
                                8   !
                                9   ! Inputs: parameters to desired service in appropriate registers.
                               10   !
                               11   ! Outputs: none.
                               12   !
                               13   ! Description: this module provides the following low level i/o services:
                               14   !
                               15   !       Set_Print_Pos - sets the print position
                               16   !       Write_Char - writes a character to the current print position
                               17   !       Get_Char - returns the character code of the character at the
                               18   !                  desired position
                               19   !
                               20   !             These services are based on the routines SETCUR, WPCHAR,
                               21   !             and GTCHAR of the dual-column support component of the
                               22   !             Applications Development Library.
                               23   !
                               24   !*******************************************************************************
                               25
                               26
                               27   ! Imports (Note: all symbols in upper case are imported)
                               28
                               29   !From the Mono ROM
                               30
                               31   !Variables
                               32
          =9C78                33   UDG             EQU     9C78H
          =9C8D                34   ATTR_P          EQU     9C8DH
          =9C8E                35   MASK_P          EQU     9C8EH
          =9C91                36   P_FLAG          EQU     9C91H
          =9CC2                37   VIDMOD          EQU     9CC2H
                               38
                               39
                               40   ! Exports
                               41
                               42   !Variables
                               43
                               44   !               chtbl
                               45
                               46   !Services
                               47
                               48   !               write_Char, Set_Print_Pos, Get_Char
                               49
                               50
                               51   ! Locals
                               52
                               53   !Constants
                               54
          =0018                55   scrsz           equ     24      ; 24 lines
          =3C00                56   chrset          equ     3c00h   ; rom char.table-100h
          =893A                57   grphst          equ     ((grpst)-100h) std.graphics chars.
                               58
                               59
                               60   !Variables
          =E890                61           ORG     0E890H
                               62
E890  3C00                     63   chtal           defw    chrset  ; address of character table  57391
                               64                                   ; (defaults to to2068 system
                               65                                   ;  rom)
E892  893A                     66   grtbl           defw    grphst  ; address of std.graphics tbl.57393
E894  20                       67   linlen          defb    20h     ; line length (32)            57395
E895  1821                     68   curpos          defw    1821h   ; current position (internal  57396
                               69                                   ;  format)
E897  4000                     70   dfadrs          defw    4000h   ; current display file addrs. 57399
E899  00                       71   maskb           defb    0       ; mask to be applied to       57400
                               72                                   ; display characters
E89A  00                       73   attbyt          defb    0h      ; current attributes          57401
E89B  00                       74   gtindx          defb    00h     ; index for adjusting         57402
                               75                                   ; ch.code in gtchar
E89C  00                       76   attmsk          defb    0       ; attribute mask              57407
                               77
```

```
                                    78
                                    79  ;*****************************************************************
                                    80  ;
                                    81  ; WRITE_CHAR
                                    82  ;
                                    83  ; Inputs: A = character code.
                                    84  ;
                                    85  ; Outputs: BC = 0000 - OK
                                    86  ;              = 00XX - error, XX <> 00
                                    87  ;
                                    88  ; Globals Read: UDG
                                    89  ;                grtbl
                                    90  ;                chtbl
                                    91  ;                linlen
                                    92  ;                maskb
                                    93  ;
                                    94  ; Globals Written: none.
                                    95  ;
                                    96  ; Procedures Called: ldattr
                                    97  ;                     ldposn
                                    98  ;                     updatt
                                    99  ;                     stposn
                                   100  ;
                                   101  ; Procedures Called By: DISPLAY_CHAR
                                   102  ;                       V_SCROLL
                                   103  ;
                                   104  ; Description: This service writes the specified character to the current
                                   105  ;              print position.
                                   106  ;
                                   107  ;*****************************************************************
                                   108
                                   109
E89D  F5                           110  write_char      push    af
E89E  D5                           111                  push    de
E89F  E5                           112                  push    hl
E8A0  CD EA24                      113                  call    ldattr          ; entry with code in a
                                   114                                          ; get attribute and mask controls
                                   115
                                   116                                          ; entry to use attribute values in
                                   117                                          ; internal variables without reloading
E8A3  CD E9DF                      118  wrch01          call    ldposn          ; load registers for current position
                                   119                                          ; bc=cursor position from 'curpos'
                                   120                                          ; hl=display file address from 'dfsadr'
                                   121                                          ; a=code for char. to be displayed
                                   122
E8A6  C5                           123                  push    bc              ; save cursor position
E8A7  FE 80                        124                  cp      80h             ; test if ascii printable (20h-7fh)
E8A9  38 19                        125                  jr      c, Wrch12       ; yes
E8AB  FE 90                        126                  cp      90h             ; test if std. graphics (80h-8fh)
E8AD  38 09                        127                  jr      c, Wrch11
E8AF  ED 48 9C78                   128                  ld      bc, (UDG)       ; user-defined graphics(90h-a4h)
E8B3  05                           129                  dec     b               ; adjust address-1C00h
E8B4  D6 70                        130                  sub     70h             ; adjust for index into table
E8B6  18 0C                        131                  jr      Wrch13
E8B8  ED 48 E892                   132  wrch11          ld      bc, (grtbl)     ; address of graphics table
E8BC  D6 60                        133                  sub     60h             ; adjust for index into table
E8BE  18 04                        134                  jr      Wrch13
E8C0  ED 48 E890                   135  wrch12          ld      bc, (chtbl)     ; character table
E8C4  E8                           136  wrch13          ex      de, hl          ; dfsadr.in display file
E8C5  26 00                        137                  ld      h, 0
E8C7  6F                           138                  ld      l, a            ; code is index into table
E8C8  29                           139                  add     hl, hl
E8C9  29                           140                  add     hl, hl
E8CA  29                           141                  add     hl, hl
E8CB  09                           142                  add     hl, bc          ; hl=pixel pattern in table
E8CC  C1                           143                  pop     bc
E8CD  E8                           144                  ex      de, hl
E8CE  79                           145                  ld      a, c            ; test if end of line
E8CF  3C                           146                  inc     a
E8D0  3A E894                      147                  ld      a, (linlen)
E8D3  20 05                        148                  jr      nz, Wrch14
E8D5  3C                           149                  inc     a               ; start of new line
E8D6  05                           150                  dec     b               ; bump line no.
E8D7  4F                           151                  ld      c, a            ; linlen-1=start of line
E8D8  18 01                        152                  jr      Wrch15
E8DA  3C                           153  wrch14          inc     a
E8DB                               154  wrch15                                  ; here to put character in display file
E8DB  C5                           155  wrch2           push    bc              ; cursor position
E8DC  E5                           156                  push    hl              ; display file adrs.
E8DD  3A E899                      157                  ld      a, (maskb)
E8E0  06 FF                        158                  ld      b, -1
E8E2  1F                           159                  rra
E8E3  30 01                        160                  jr      c, wrch3        ; if xoring new into old
E8E5  04                           161                  inc     b               ; b=-1 if xoring  =0 if not
E8E6  1F                           162  wrch3           rra
E8E7  1F                           163                  rra
E8E8  9F                           164                  sbc     a, a
E8E9  4F                           165                  ld      c, a            ; c=-1 for inverse  =0 if not
E8EA  3E 09                        166                  ld      a, 9            ; scan count
E8EC  A7                           167                  and     a
E8ED  EB                           168                  ex      de, hl
E8EE  08                           169  wrch5           ex      af, af'
E8EF  1A                           170                  ld      a, (de)         ; old bit row
E8F0  A0                           171                  and     b
E8F1  AE                           172                  xor     (hl)
E8F2  A9                           173                  xor     c               ; invert if c=-1
E8F3  12                           174                  ld      (de), a         ; update display file
E8F4  08                           175                  ex      af, af'
E8F5  14                           176                  inc     d               ; next scan
E8F6  23                           177                  inc     hl              ; next byte in char.file
E8F7  3D                           178                  dec     a               ; test if more scans
E8F8  20 F4                        179                  jr      nz, wrch5
E8FA  15                           180  wrch7           dec     d                       ; adjust df adrs.
E8FB  EB                           181                  ex      de, hl          ; df adrs. to hl
E8FC  CD E9E7                      182                  call    updatt          ; update attribute byte
```

```
E8FF   E1                  184                    pop     hl              ; starting address
E900   C1                  185                    pop     bc              ; cursor position
E901   0D                  186                    dec     c               ; adjust cursor position
E902   23                  187                    inc     hl              ; adjust df address
E903   CD E907             188    wrchar          call    steeen          ; store new position
E906   0E 00               189    goodret         ld      c, 0            ; return bc=0
E908   06 00               190    errret          ld      b, 0            ; enter here with c non-zero
E90A   E1                  191                    pop     hl
E90B   D1                  192                    pop     de
E90C   F1                  193                    pop     af
E90D   C9                  194                    ret
                           195
                           196
                           197    ;*****************************************************************************
                           198    ;
                           199    ; SET_PRINT_POS
                           200    ;
                           201    ; Inputs: B = row
                           202    ;         C = column
                           203    ;
                           204    ; Outputs: BC = 0000 - OK
                           205    ;
                           206    ; Globals Read: none.
                           207    ;
                           208    ; Globals Written: none.
                           209    ;
                           210    ; Procedures Called: convfm
                           211    ;                    updposn
                           212    ;
                           213    ; Procedures Called By: DISPLAY_CHAR
                           214    ;                       V_SCROLL
                           215    ;
                           216    ; Description: this service sets the print position to the specified row
                           217    ;              and column.  The row position must be in the range 0..23
                           218    ;              and the column position must be in the range 0..31.
                           219    ;
                           220    ;*****************************************************************************
                           221
                           222
E90E   F5                  223    set_print_pos   push    af
E90F   D5                  224                    push    de
E910   E5                  225                    push    hl
E911   CD E99D             226                    call    convfm          ; entry with line/col. in bc reg.
                           227                                            ; convert to internal format
E914   CD E948             228                    call    updposn         ; calc. and save the print posn.
E917   0E 00               229                    ld      c, 0            ; return bc=0
E919   06 C0               230                    ld      b, 0            ; enter here with c non-zero
E91B   E1                  231                    pop     hl
E91C   D1                  232                    pop     de
E91D   F1                  233                    pop     af
E91E   C9                  234                    ret
                           235
                           236
                           237    ;*****************************************************************************
                           238    ;
                           239    ; GET_CHAR
                           240    ;
                           241    ; Inputs: B = row
                           242    ;         C = column
                           243    ;
                           244    ; Outputs: A = character code, if a char is found, 0 otherwise
                           245    ;          BC = 00XX - OK, match found, XX = A
                           246    ;             = 0000 - OK, no match found
                           247    ;
                           248    ; Globals Read: chtbl
                           249    ;               gtindx
                           250    ;               grtbl
                           251    ;               UGS
                           252    ;
                           253    ; Globals Written: gtindx
                           254    ;
                           255    ; Procedures Called: convfm
                           256    ;                    calcpos
                           257    ;
                           258    ; Procedures Called By: PUT_SPRITE
                           259    ;
                           260    ; Description: this service returns the character code of the character
                           261    ;              at the specified screen location.  The row and column must
                           262    ;              be in the range specified above for Set_Print_Pos.
                           263    ;
                           264    ;*****************************************************************************
                           265
                           266
E91F   D5                  267    get_char        push    de
E920   E5                  268                    push    hl
E921   CD E99D             269                    call    convfm          ; entry with position in bc
                           270                                            ; convert to internal format
E924   CD E9AF             271                    call    calcpos         ; get display file adrs.
E927   ED 5B E890          272                    ld      de, (chtbl)     ; char.table
E928   06 60               273                    ld      b, 96           ; no. of printable characters
E92D   3E 80               274                    ld      a, 80h          ; set adjustment index
E92F   32 E898             275    gtch1           ld      (gtindx), a
E932   14                  276                    inc     d               ; adjust to start of table
E933   EB                  277                    ex      de, hl          ; df adrs.in de;char.set in hl
E934   C5                  278    gtch2           push    bc              ; save count
E935   D5                  279                    push    de              ; save adrs. in df
E936   E5                  280                    push    hl              ; save adrs. in char.table
E937   1A                  281                    ld      a, (de)         ; scan row from df
E938   AE                  282                    xor     (hl)            ; test against char. set
E939   28 10               283                    jr      z, gtch3        ; match
E93B   F5                  284                    push    af
E93C   3A E898             285                    ld      a, (gtindx)
E93F   FE 90               286                    cp      90h             ; test if std.graphics
E941   20 03               287                    jr      nz, gtch23
E943   F1                  288                    pop     af
E944   18 28               289                    jr      gtch4           ; do not test for inverse
E946   F1                  290    gtch23          pop     af
E947   3C                  291                    inc     a               ; test inverse
```

```
E948  20 24           jr      nz, gtch4
E94A  3C              dec     a               ! a=-1 for inverse
E94B  4F              ld      c, a            ! c=0 for match  -1 for inverse
E94C  06 07           ld      b, 7
E94E  14              inc     d               ! 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+5)=char.
                      ! count

E957  79              ld      a, c            ! a=-1 if match on inverse
E958  C1              pop     bc
E959  C1              pop     bc
E95A  C1              pop     bc
E95B  4F              ld      c, a            ! bccount c=-1 if match on inverse
E95C  3A E999         ld      a, (gtindx)     ! adjust char.code
E95F  9C              sub     b               ! ancher.code
E960  FE 20           cp      20h             ! test if space
E962  20 05           jr      nz, gtch32
E964  0C              inc     c               ! test if match on inverse space
E965  20 02           jr      nz, gtch32
E967  3E 8F           ld      a, 8Fh          ! set code for graphics block
E969  4F              ld      c, a
E96A  06 00           ld      b, 0
E96C  18 2C           jr      gc_exit         ! char.code in a and in bc

E96E  E1              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 80           djnz    gtch2           ! look again

                      ! here to test if done

E977  EB              ex      de, hl          ! df adrs.to hl
E978  3A E898         ld      a, (gtindx)
E97B  FE 80           cp      80h             ! test if std.chr.set
E97D  20 04           jr      nz, gtch5       ! try graphics
E97F  ED 5B E892      ld      de, (grtbl)     ! std.graphics table
E983  06 10           ld      b, 16           ! no. of entries
E985  3E 40           ld      a, 90h          ! index
E987  18 46           jr      gtch1
E989  FE 90           cp      90h             ! test if std.graphics
E98B  20 08           jr      nz, gtch6       ! done if not
E98D  ED 5B 9C7B      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 A9           ld      a, 0a9h         ! index
E996  18 97           jr      gtch1
E998  48              ld      c, b            ! here when no match anywhere
E999  AF              xor     a               ! set bc and anzero
E99A  81              add     c
E99B  D1              pop     de
E99C  C9              ret

                      !**************************************************************************
                      !
                      ! CONVFM
                      !
                      ! 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 E894         convfm  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
```

```
                        391   !**************************************************************************
                        392   !
                        393   ! UPDPOSN
                        394   !
                        395   ! Inputs: B = row in internal format
                        396   !         C = column in internal format
                        397   !
                        398   ! Outputs: none
                        399   !
                        400   ! Globals Read: none.
                        401   !
                        402   ! Globals Written: none
                        403   !
                        404   ! Procedures Called: calcpos
                        405   !                   stposn
                        406   !
                        407   ! Procedures Called By: set_print_pos
                        408   !
                        409   ! Description: this routine set the print position to the specified value.
                        410   !
                        411   !**************************************************************************
                        412
                        413
                        414   updposn                           ; enter here with bc=line/col.in
                        415                                     ; internal format
                        416
8948  CD 894F           417                   call    calcpos         ; calculate position
8948  CD 8987           418                   call    stposn          ; store updated position and return
894E  C9                419                   ret
                        420
                        421
                        422   !**************************************************************************
                        423   !
                        424   ! CALCPOS
                        425   !
                        426   ! Inputs: B = row in internal format
                        427   !         C = column in internal format
                        428   !
                        429   ! Outputs: HL = address in display file
                        430   !
                        431   ! Globals Read: linlen
                        432   !
                        433   ! Globals Written: none.
                        434   !
                        435   ! Procedures Called: lnbu
                        436   !
                        437   ! Procedures Called By: get_char
                        438   !                       updposn
                        439   !
                        440   ! Description: this routine calculates the position in the display file
                        441   !              corresponding to the specified row and column position.
                        442   !
                        443   !**************************************************************************
                        444
                        445
894F  CD 898C           446   calcpos           call    lnbu            ; get display file adrs.
8952  3A 8894           447                     ld      a, (linlen)     ; add column offset
8955  3C                448                     inc     a
8956  91                449                     sub     c
8957  5F                450                     ld      e, a
8958  16 00            451                     ld      d, 0
895A  19                452                     add     hl, de
895B  C9                453                     ret
                        454
                        455
                        456   !**************************************************************************
                        457   !
                        458   ! LNBU
                        459   !
                        460   ! Inputs: B = row in internal format
                        461   !
                        462   ! Outputs: HL = address in display file of start of line B.
                        463   !
                        464   ! Globals Read: VIDMOD
                        465   !
                        466   ! Globals Written: none.
                        467   !
                        468   ! Procedures Called: none.
                        469   !
                        470   ! Procedures Called By: calcpos
                        471   !
                        472   ! Description: this routine calculates the address of the start of row B
                        473   !              in the display file.  This is the first byte of the first
                        474   !              scan line.
                        475   !
                        476   !**************************************************************************
                        477
                        478
895C                    479   lnbu                              ; get display file address
                        480                                     ; for start of line in b
895C  3E 18             481                     ld      a, scrsz
895E  90                482                     sub     b
895F  57                483                     ld      d, a
8960  8F                484                     rrca
8961  8F                485                     rrca
8962  8F                486                     rrca
8963  E6 E0            487                     and     0e0h
8965  6F                488                     ld      l, a
8966  7A                489                     ld      a, d
8967  E6 18            490                     and     18h
8969  F6 40            491                     or      40h
896B  67                492                     ld      h, a
896C  3A 9CC2           493                     ld      a, (VIDMOD)
896F  CB 47             494                     bit     0, a            ; test if df1
8971  C8                495                     ret     z
8972  3E 20            496                     ld      a, 20h          ; set to df2
8974  B4                497                     or      h
8975  67                498                     ld      h, a
8976  C9                499                     ret
                        500
                        501
```

```
                                502  ;**********************************************************************
                                503  ;
                                504  ; STPOSN
                                505  ;
                                506  ; Inputs: BC = current print position
                                507  ;         HL = display file address
                                508  ;
                                509  ; Outputs: none
                                510  ;
                                511  ; Globals Read: none.
                                512  ;
                                513  ; Globals Written: curpos
                                514  ;                  dfadrs
                                515  ;
                                516  ; Procedures Called: none.
                                517  ;
                                518  ; Procedures Called By: write_char
                                519  ;                       updscan
                                520  ;
                                521  ; Description: this routine saves the inputs in the appropriate variables.
                                522  ;
                                523  ;**********************************************************************
                                524
                                525
E9D7                            526  stposn                          ; store cursor position
E9D7   ED 43 E895               527                  ld      (curpos), bc
E9DB   22 E897                  528                  ld      (dfadrs), hl
E9DE   C9                       529                  ret
                                530
                                531
                                532  ;**********************************************************************
                                533  ;
                                534  ; LDPOSN
                                535  ;
                                536  ; Inputs: none.
                                537  ;
                                538  ; Outputs: BC = current print position
                                539  ;          HL = current display file address
                                540  ;
                                541  ; Globals Read: curpos
                                542  ;               dfadrs
                                543  ;
                                544  ; Globals Written: none.
                                545  ;
                                546  ; Procedures Called: none.
                                547  ;
                                548  ; Procedures Called By: write_char
                                549  ;
                                550  ; Description: this routine retrieves the current print position and
                                551  ;              display file address from the appropriate variables.
                                552  ;
                                553  ;**********************************************************************
                                554
                                555
E9DF                            556  ldposn                          ; load cursor position
E9DF   ED 4B E895               557                  ld      bc, (curpos)
E9E3   2A E897                  558                  ld      hl, (dfadrs)
E9E6   C9                       559                  ret
                                560
                                561
                                562  ;**********************************************************************
                                563  ;
                                564  ; UPDATT
                                565  ;
                                566  ; Inputs: HL = address of any scan line of a character position
                                567  ;
                                568  ; Outputs: none.
                                569  ;
                                570  ; Globals Read: attbyt
                                571  ;               attmsk
                                572  ;               maskb
                                573  ;
                                574  ; Globals Written: none.
                                575  ;
                                576  ; Procedures Called: calcatt
                                577  ;
                                578  ; Procedures Called By: write_char
                                579  ;
                                580  ; Description: this routine updates the attribute byte for the character
                                581  ;              position identified by the address in HL.  The attribute
                                582  ;              value written is taken from the attbyt variable.
                                583  ;
                                584  ;**********************************************************************
                                585
                                586
E9E7                            587  updatt                          ; update attribute byte for character
                                588                                  ; just printed
                                589                                  ; hl = any scan of char.in of
                                590
E9E7   CD EA14                  591                  call    calcatt         ; adrs. of attribute byte to hl
E9EA   3A E89A                  592                  ld      a, (attbyt)
E9ED   5F                       593                  ld      e, a            ; current attr.value to e
E9EE   3A E89C                  594                  ld      a, (attmsk)
E9F1   57                       595                  ld      d, a            ; mask to d
E9F2   3A E899                  596                  ld      a, (maskb)
E9F5   47                       597                  ld      b, a            ; flags to b
E9F6   7E                       598                  ld      a, (hl)         ; byte from attribute file
E9F7   AB                       599                  xor     e
E9F8   A2                       600                  and     d
E9F9   AB                       601                  xor     e               ; new attr.for 0: old for 1
E9FA   CB 60                    602                  bit     4, b            ; set ink complement of paper?
E9FC   28 08                    603                  jr      z, updat1
E9FE   E6 F8                    604                  and     0f8h            ; set ink to black
EA00   CB 6F                    605                  bit     5, a            ; is paper 4-7
EA02   20 02                    606                  jr      nz, updat1      ; yes - use black ink
EA04   F6 07                    607                  or      7               ; no -  use white ink
EA06   CB 70                    608  updat1          bit     6, b            ; set paper complement of ink?
EA08   28 08                    609                  jr      z, updat2       ;
EA0A   E6 C7                    610                  and     0c7h            ; set paper to black
EA0C   CB 57                    611                  bit     2, a            ; is ink 4-7
EA0E   20 02                    612                  jr      nz, updat2      ; yes - use black paper
EA10   F6 38                    613                  or      38h             ; no -  use white paper
EA12   77                       614  updat2          ld      (hl), a         ; to attribute file
EA13   C9                       615                  ret
```

```
                        616
                        617
                        618   !*************************************************************************
                        619   !
                        620   ! CALCATT
                        621   !
                        622   ! Inputs: HL = address of any scan of a character position
                        623   !
                        624   ! Outputs: HL = address of attribute byte
                        625   !
                        626   ! Globals Read: none.
                        627   !
                        628   ! Globals Written: none.
                        629   !
                        630   ! Procedures Called: none.
                        631   !
                        632   ! Procedures Called By: updatt
                        633   !
                        634   ! Description: this routine calculates the address of the attribute byte
                        635   !              corresponding to the character position identified with
                        636   !              HL.
                        637   !
                        638   !*************************************************************************
                        639
                        640
8A14  7C                641   calcatt         ld      a, h            ; upper byte of address
8A15  0F                642                   rrca
8A16  0F                643                   rrca
8A17  0F                644                   rrca
8A18  E6 03             645                   and     03h
8A1A  F6 58             646                   or      58h
8A1C  CB 6C             647                   bit     5, h            ; test which df
8A1E  28 02             648                   jr      z, calcal
8A20  F6 20             649                   or      20h             ; df2
8A22  67                650   calcal          ld      h, a            ; adrs. of attribute in df2
8A23  C9                651                   ret
                        652
                        653
                        654   !*************************************************************************
                        655   !
                        656   ! LOATTR
                        657   !
                        658   ! Inputs: none.
                        659   !
                        660   ! Outputs: none.
                        661   !
                        662   ! Globals Read: ATTR_P
                        663   !               MASK_P
                        664   !               P_FLAG
                        665   !
                        666   ! Globals Written: attbyt
                        667   !                  attmsk
                        668   !                  maskb
                        669   !
                        670   ! Procedures Called: none.
                        671   !
                        672   ! Procedures Called By: write_char
                        673   !
                        674   ! Description: this routine sets the values of the variables used by this
                        675   !              module for manipulating attributes with values from the
                        676   !              variables used by the Home ROM.
                        677   !
                        678   !*************************************************************************
                        679
                        680
8A24                    681   loattr                          ; load internal attribute variables
                        682                                   ; from system variables
                        683
8A24  F5                684                   push    af              ; save a
8A25  3A 5C8D           685                   ld      a, (ATTR_P)
8A28  32 E894           686                   ld      (attbyt), a
8A2B  3A 5C8E           687                   ld      a, (MASK_P)
8A2E  32 E89C           688                   ld      (attmsk), a
8A31  3A 5C91           689                   ld      a, (P_FLAG)
8A34  0F                690                   rrca                    ; shift odd bits to even
8A35  32 E899           691                   ld      (maskb), a
8A38  F1                692                   pop     af
8A39  C9                693                   ret
                        694
                        695
                        696   ! Standard graphics characters
                        697
8A3A  00                698   grast   defb 00000000b          ; code 80       graphics space
8A3B  00                699           defb 00000000b
8A3C  00                700           defb 00000000b
8A3D  00                701           defb 00000000b
8A3E  00                702           defb 00000000b
8A3F  00                703           defb 00000000b
8A40  00                704           defb 00000000b
8A41  00                705           defb 00000000b
                        706
8A42  0F                707           defb 00001111b          ; code 81       graphics
8A43  0F                708           defb 00001111b
8A44  0F                709           defb 00001111b
8A45  0F                710           defb 00001111b
8A46  00                711           defb 00000000b
8A47  00                712           defb 00000000b
8A48  00                713           defb 00000000b
8A49  00                714           defb 00000000b
                        715
8A4A  F0                716           defb 11110000b          ; code 82       graphics
8A4B  F0                717           defb 11110000b
8A4C  F0                718           defb 11110000b
8A4D  F0                719           defb 11110000b
8A4E  00                720           defb 00000000b
8A4F  00                721           defb 00000000b
8A50  00                722           defb 00000000b
8A51  00                723           defb 00000000b
                        724
8A52  FF                725           defb 11111111b          ; code 83       graphics
8A53  FF                726           defb 11111111b
8A54  FF                727           defb 11111111b
8A55  FF                728           defb 11111111b
```

```
EA56   00                729         defb 00000000b
EA57   00                730         defb 00000000b
EA58   00                731         defb 00000000b
EA59   00                732         defb 00000000b
                         733
EA5A   00                734         defb 00000000b          ; code 04       graphics
EA5B   00                735         defb 00000000b
EA5C   00                736         defb 00000000b
EA5D   00                737         defb 00000000b
EA5E   0F                738         defb 00001111b
EA5F   0F                739         defb 00001111b
EA60   0F                740         defb 00001111b
EA61   0F                741         defb 00001111b
                         742
EA62   0F                743         defb 00001111b          ; code 05       graphics
EA63   0F                744         defb 00001111b
EA64   0F                745         defb 00001111b
EA65   0F                746         defb 00001111b
EA66   0F                747         defb 00001111b
EA67   0F                748         defb 00001111b
EA68   0F                749         defb 00001111b
EA69   0F                750         defb 00001111b
                         751
EA6A   F0                752         defb 11110000b          ; code 06       graphics
EA6B   F0                753         defb 11110000b
EA6C   F0                754         defb 11110000b
EA6D   F0                755         defb 11110000b
EA6E   0F                756         defb 00001111b
EA6F   0F                757         defb 00001111b
EA70   0F                758         defb 00001111b
EA71   0F                759         defb 00001111b
                         760
EA72   FF                761         defb 11111111b          ;code 07        graphics
EA73   FF                762         defb 11111111b
EA74   FF                763         defb 11111111b
EA75   0F                764         defb 11111111b
EA76   0F                765         defb 00001111b
EA77   0F                766         defb 00001111b
EA78   0F                767         defb 00001111b
EA79   0F                768         defb 00001111b
                         769
EA7A   00                770         defb 00000000b          ; code 08       graphics
EA7B   00                771         defb 00000000b
EA7C   00                772         defb 00000000b
EA7D   00                773         defb 00000000b
EA7E   F0                774         defb 11110000b
EA7F   F0                775         defb 11110000b
EA80   F0                776         defb 11110000b
EA81   F0                777         defb 11110000b
                         778
EA82   0F                779         defb 00001111b          ; code 09       graphics
EA83   0F                780         defb 00001111b
EA84   0F                781         defb 00001111b
EA85   0F                782         defb 00001111b
EA86   F0                783         defb 11110000b
EA87   F0                784         defb 11110000b
EA88   F0                785         defb 11110000b
EA89   F0                786         defb 11110000b
                         787
EA8A   F0                788         defb 11110000b          ; code 0A       graphics
EA8B   F0                789         defb 11110000b
EA8C   F0                790         defb 11110000b
EA8D   F0                791         defb 11110000b
EA8E   F0                792         defb 11110000b
EA8F   F0                793         defb 11110000b
EA90   F0                794         defb 11110000b
EA91   F0                795         defb 11110000b
                         796
EA92   FF                797         defb 11101111b          ; code 0B       graphics
EA93   FF                798         defb 11101111b
EA94   FF                799         defb 11101111b
EA95   FF                800         defb 11101111b
EA96   F0                801         defb 11100000b
EA97   F0                802         defb 11100000b
EA98   F0                803         defb 11100000b
EA99   F0                804         defb 11100000b
                         805
EA9A   00                806         defb 00000000b          ; code 0C       graphics
EA9B   00                807         defb 00000000b
EA9C   00                808         defb 00000000b
EA9D   00                809         defb 00000000b
EA9E   FF                810         defb 11111111b
EA9F   FF                811         defb 11111111b
EAA0   FF                812         defb 11111111b
EAA1   FF                813         defb 11111111b
                         814
EAA2   0F                815         defb 00001111b          ; code 0D       graphics
EAA3   0F                816         defb 00001111b
EAA4   0F                817         defb 00001111b
EAA5   0F                818         defb 00001111b
EAA6   FF                819         defb 11111111b
EAA7   FF                820         defb 11111111b
EAA8   FF                821         defb 11111111b
EAA9   FF                822         defb 11111111b
                         823
EAAA   F0                824         defb 11110000b          ;code 0E        graphics
EAAB   F0                825         defb 11110000b
EAAC   F0                826         defb 11110000b
EAAD   F0                827         defb 11110000b
EAAE   FF                828         defb 11111111b
EAAF   FF                829         defb 11111111b
EAB0   FF                830         defb 11111111b
EAB1   FF                831         defb 11111111b
                         832
EAB2   FF                833         defb 11111111b          ;code 0F        graphics
EAB3   FF                834         defb 11111111b
EAB4   FF                835         defb 11111111b
EAB5   FF                836         defb 11111111b
EAB6   FF                837         defb 11111111b
EAB7   FF                838         defb 11111111b
EAB8   FF                839         defb 11111111b
EAB9   FF                840         defb 11111111b
                         841
                         842
                         843         end
```

```
IOLOW   CP/M-80/11 version 10.36.14       7-May-84  15:39:10
Low level I/O module   IOLOW.SRC

A       Reserved        ATTBYT  E89A        ATTMSK  E89C        ATTRSP  5C80        9       Reserved
C       Reserved        CALCAT  E814        CALCA1  E822        CALCPO  E9AP        CHRSET  3C00
CNTBL   E890            CONVPN  E90D        CURPOS  E895        D       Reserved    DPADRS  E897
E       Reserved        ERRRET  E908        GCSERI  E99A        GETSCH  E91P        GOODRE  E906
GRPNST  E93A            GRPST   E43A        GRTBC   E692        GTCH1   E92P        GTCH2   E934
GTCH23  E946            GTCH3   E948        GTCH31  E94E        GTCH32  E969        GTCH4   E96E
GTCH5   E989            GTCH6   E998        GTIND1  E898        H       Reserved    L       Reserved
LOATTR  EA26            LOPOSN  E9DP        LINLEN  E894        LNBU    E98C        M       Reserved
MASKB   E899            MASKBP  5C0E        PSW     Reserved    PSPLAG  5C91        SCRSZ   0318
SETSPR  E90E            SP      Reserved    STPOSN  E9D7        UDG     5C7B        UPDATT  E9E7
UPDAT1  EA06            UPDAT2  EA12        UPDPOS  E9A8        VIDMOD  5CC2        WACHRT  E903
WRCH01  E8A3            WRCH11  E988        WRCH1E  E8C0        WRCH13  E8C4        WRCH14  E8DA
WRCH15  E8D8            WRCH2   E8DB        WRCH3   E8F6        WRCH4   E8FE        WRCH7   E8FA
WRITE8  E890

     No errors detected
```

APPENDIX D

TS2068 PCB Assembly and Schematic Diagram

The following Appendix contains the PCB Assembly Drawing, the PCB Parts List, and PCB Schematic Diagram (a "fold-out" page located just inside the back cover). The Table below contains some corrections to the Schematic Diagram.

TS2068 PCB Schematic Diagram Corrections

Page 34 of the Technical Manual shows pin 9 of the joystick ports grounded as it should be. The traces were left off the TS2068 PCB.

VR1: U3-33 goes to VR1/Q5

Q4: Connect base to R55/R54

Solder dots on horizontal lines below keyboard:

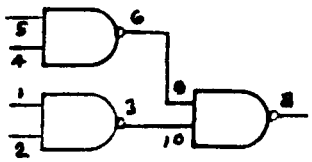
U12-4 to U3-65 ($\overline{WR}$)

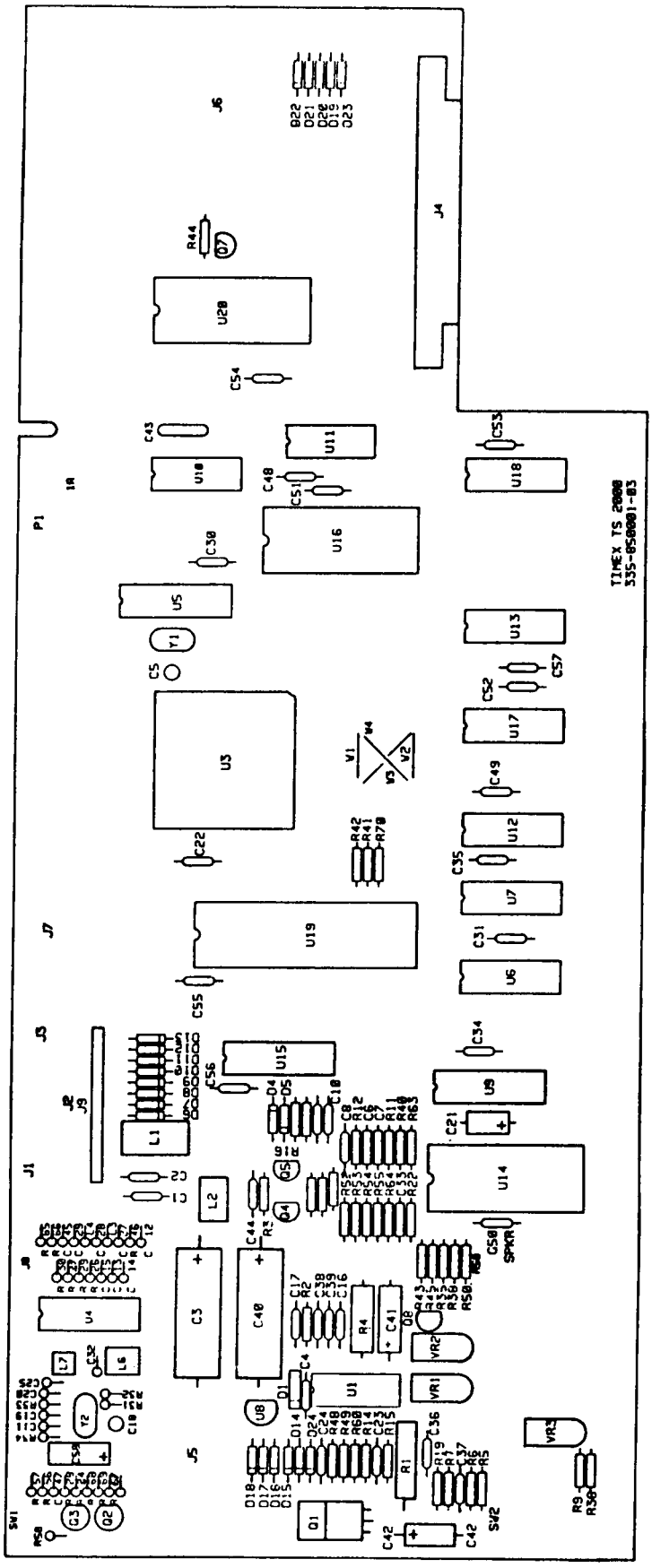
U12-5 to U3-66 ($\overline{MREQ}$)

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			

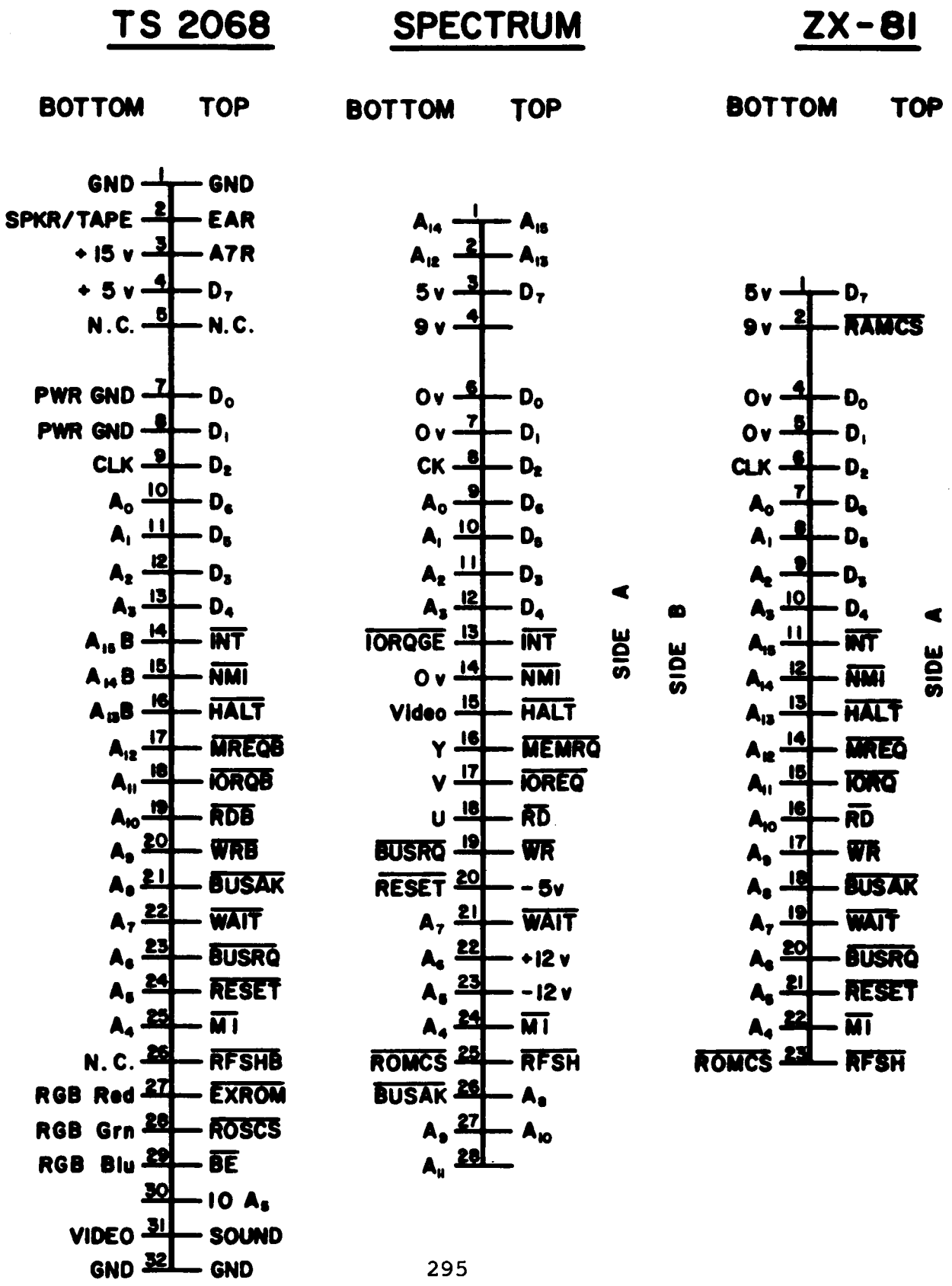
APPENDIX D

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

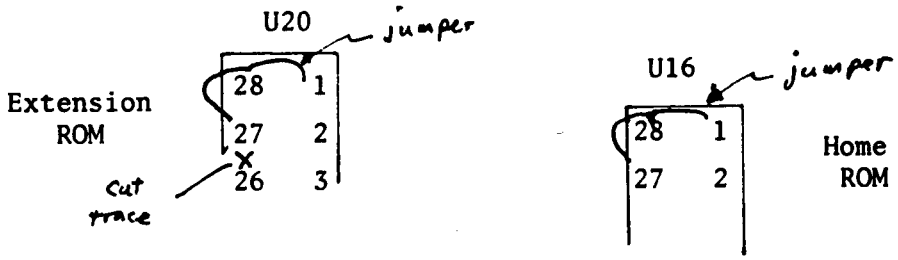


APPENDIX F

August 1985
Bob Orrfelt

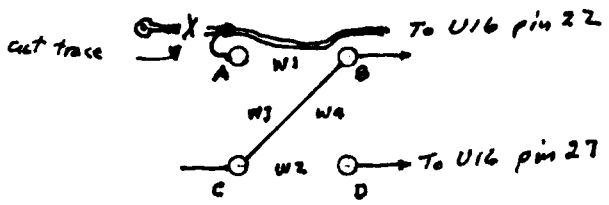
TS2068 MODIFICATIONS FOR EPROMS

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 D2[illegible]28C. INSERT W1 & W2.
FOR [illegible] INSERT W3 & W4 (NOT USED)

CHANGES

ISSUE NO	[illegible]

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

DIR	OPERATION
L	B→A
H	A→B

MATERIAL	
SPEC	
SIZE & TOL	
TEMPER	
PHY SPEC	

DO NOT SCALE DRAWING

CHK	DWN N PARSONS
REF	DATE FEB 84
FIRST USED ON M2068	SCALE NONE

SUPERSEDES
SUPERSEDED BY

TIMEX
COMPUTER CORPORATION
WATERBURY CONN 06720

NAME TS2000 SYSTEM SCHEMATIC DIAGRAM

DWG NO. 335-870500

BRUNING 44131 22870 3