BASIS 108

INSTRUCTION MANUAL

NOTICE

BASIS Microcomputer GmbH reserves the right to make improvements in the product described in this manual at any time and without notice.

Warning. - This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

JMW 982

TABLE OF CONTENTS

SECTION 3 BASIS 108 OPERATION (Continued)

SECTION 4 SOFTWARE

SECTION 5 VIDEO

SECTION 5 VIDEO (Continued)

SECTION 6 INPUT/OUTPUT

SECTION 6 INPUT/OUTPUT (Continued)

SECTION 7 SYSTEM MONITOR

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1

GENERAL INFORMATION

INTRODUCTION

This Manual contains in-depth information on the installation, operation and maintenance of your BASIS 108 system. This manual also contains programming and operating system information required for the full use of system capabilities. This manual does not teach you how to program. The bibliography in Appendix J list several references which can aid you in the use of various languages and operating systems. This section contains an overview of your BASIS 108 system.

DESCRIPTION

The BASIS 108 system has two major components: Computer chassis and keyboard. The following paragraphs describe the chassis, the computerboard and the keyboard. The inside cover shows the system with a BASIS 100 monitor.

Chassis

The chassis is a rugged and attractive cast aluminum housing that contains the computerboard and power supply. This housing or chassis consists of two parts: the base and cover. The rear panel is part of the base section. On the front of the chassis, as shown in Figure 1-1, there are two openings for disk drives. If your system was supplied without disk drives, plastic covers are installed in these openings. The POWER ON/OFF SWITCH is located on the front-lower left corner. This switch contains a red indicator that illuminates when computer power is turned on.

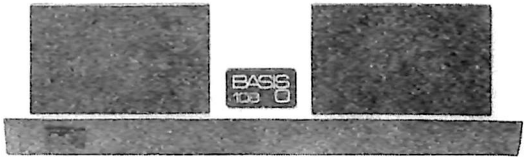


Figure 1-1. Front Panel

Chassis (Continued)

Located on the right side of the rear panel, as shown in Figure 1-2, are two power outlets for a monitor and printer. Below the outlets are the power input connector and fuse. The power cord supplied with the system is connected here.

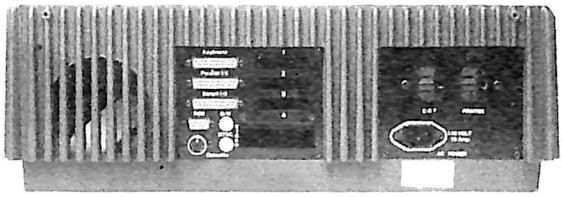


Figure 1-2. Rear Panel

In the center of the panel are three DB-25 connectors and precut holes for the mounting of three DB-25 connectors. The three mounted DB-25 connectors are for the keyboard, parallel output and serial input/output. The BNC connectors are for monochrome (black and white) and color NTSC video. The DE-9 connector is for the analog RGB video output. A cassette interface is provided through the DIN jack. The cable pass-through restrains other external cables and is wide enough to handle 50 conductor ribbon cables.

The chassis cover is removed by loosening and removing the two screws located under the front corners of the chassis and then sliding the cover towards the front of the computer. These cover hold-down screws are shown in Figure 1-3.

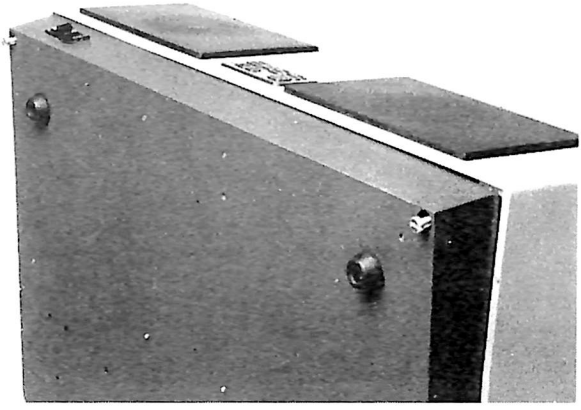


Figure 1-3. Chassis Cover Screws

Inside the Chassis

Inside the chassis are the computerboard, power supply and mounting brackets for two disk drives. Figure 1-4 shows the inside of the computer.

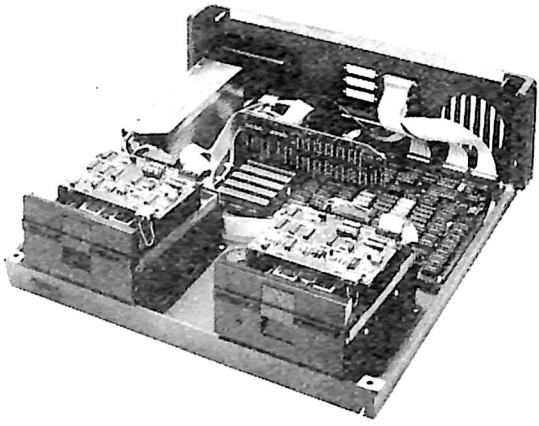


Figure 1-4. Inside the Chassis

Computerboard

The BASIS 108 is a single board computer. On this board are located the 6502 and Z-80™ microprocessors, RAM (Random Access Memory), ROM (Read-only Memory), ICs (Integrated Circuits) and other components required to make a functional computer.

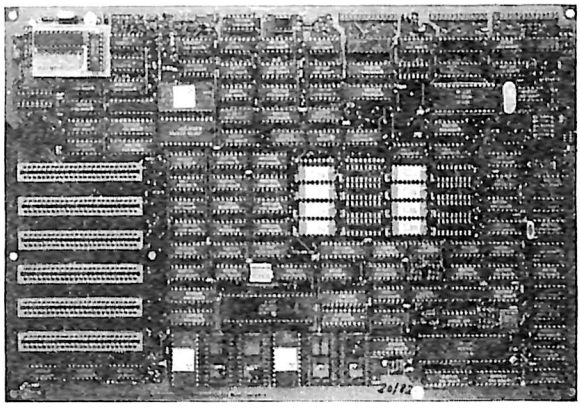


Figure 1-5. Computerboard

Computerboard (Continued)

On the left side on the board there are six 50-pin connectors (slots). These slots are numbered 2 through 7 and are used for memory expansion and for controllers and other special purpose cards. The power supply is connected to the computerboard through the connector located behind I/O slot 7. These features are shown in Figure 1-6.

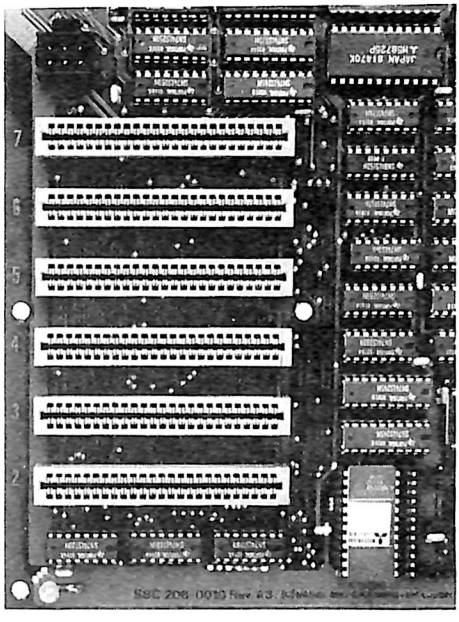


Figure 1-6. Peripheral Connectors

As shown in Figure 1-7, video output from the computerboard is available from three different places. The 10-pin connector provides RGB signals for connection to the rear panel DB-9 connector. To the right of the 10-pin RGB connector is a 2-pin connector for the B/W (monochrome) video. The 4-pin (molex) connector located on the rear left side of the board is for the color NTSC video. The External I/O connector is located in the left hand corner between the color video and power supply connectors. Controller paddles, a joystick or an external TTL interface can be connected here.

On the right rear portion of the board there are three 26-pin connectors. These connectors provide for connection of the keyboard, serial I/O, and parallel signals to the rear panel DB-25 connectors. The speaker and cassette connector is located on the right side of the board towards the rear. Figure 1-8 shows these connectors.

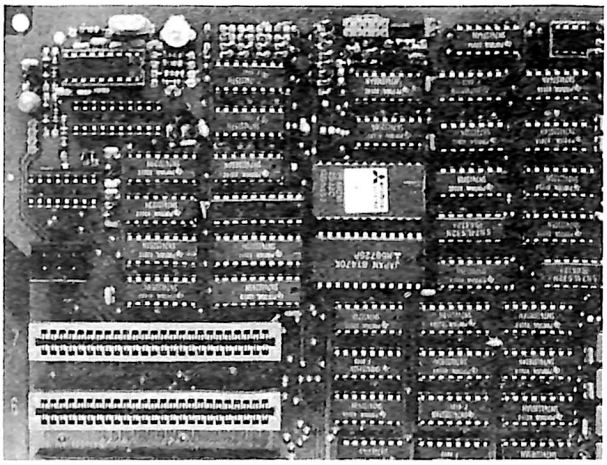


Figure 1-7. Video Outputs

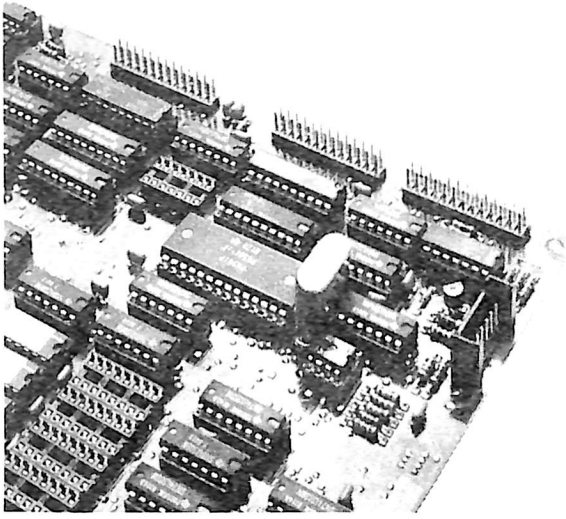


Figure 1-8. 26-pin Connectors

Computerboard (Continued)

RAM is located approximately in the middle of the board. Sockets are provided for 16 memory ICs. In the 128K configuration RAM is located in all of these sockets. In the 64K configuration (as shown in Figure 1-9) there are eight memory ICs with 64K bits each. Since both the 6502 and the Z-80 can only directly address 65,536 addresses, bank switching is used to access the second 64K of RAM.

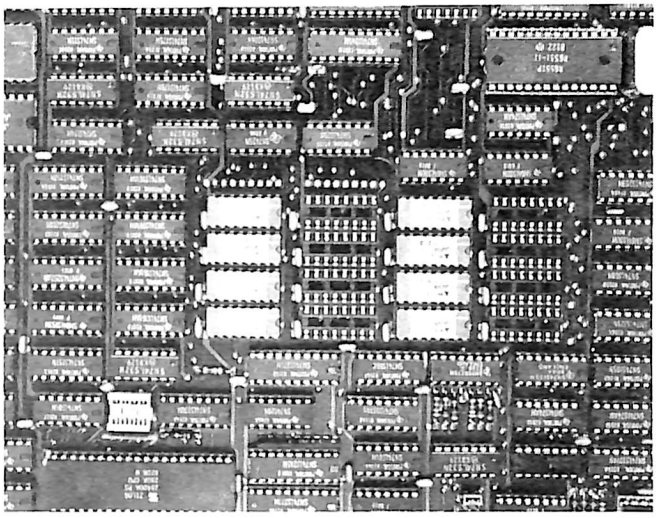


Figure 1-9. RAM Location

At the front of the computerboard, as shown in Figure 1-10, there are six ROM sockets. They can be used for resident languages or programs. The chip in the left-most socket is the BASIS 108 System Monitor. When system power is turned on, program routines in the monitor perform system initialization. Many system functions are performed by software routines located in this chip.

Power Supply

The metal housing to the left of the computerboard is the power supply. This device provides the power required to run the computer. The four output voltages are: +5 Vdc, -5 Vdc, +12 Vdc and -12 Vdc. The power supply input can either be 115 Vac or 230 Vac, over a frequency range of 47 to 400 Hz. This supply has output open and short protection.

Power Supply (Continued)

CAUTION

NEVER OPEN THE POWER SUPPLY OR WORK ON THE COMPUTER WITH THE POWER ON. ALWAYS TURN OFF THE COMPUTER AND DISCONNECT THE INPUT POWER (UNPLUG IT) BEFORE OPENING THE CHASSIS.

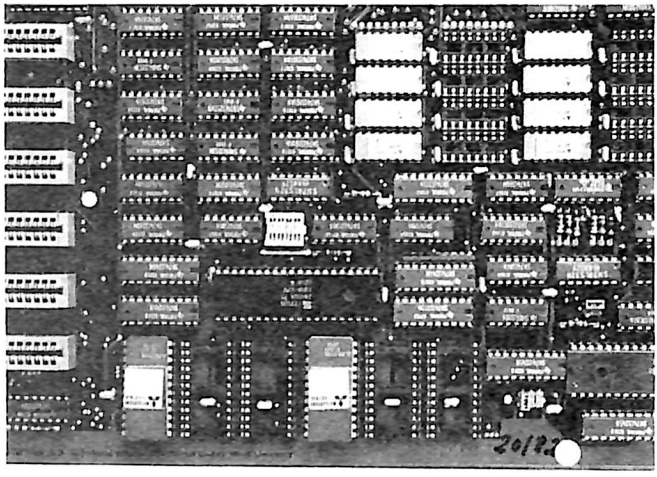


Figure 1-10. ROM/EPROM Sockets

Disk Drive Mounting Brackets

Brackets and the screws necessary to mount two disk drives are provided. If drives were not supplied with your system, the mounting screws can be found in a plastic bag attached to one of the brackets.

Keyboard

Most of your contact with the BASIS 108 will be through the keyboard. The 100 keys are divided into four main sections. These sections are: An expanded typewriter keyboard, cursor control keys, full function numeric keypad, and 15 programmable function keys. The keyboard is mounted in a very low profile plastic case. It is connected by cable to the top DB-25 connector on the computer rear panel.

Keyboard (Continued)

All keys are autorepeat. When you hold a key down, after approximately 0.6 seconds, the character automatically repeats. The status of the shift lock is shown by a red LED in the LOCK key. When the LED is illuminated the keyboard is locked in the alphalock or shiftlock condition. Alphalock is obtained by holding down the CONTROL key and pressing the LOCK key. Alphalock is the preferred mode for Apple II™ software that is not capable of accepting lowercase input or commands. Shiftlock is obtained by pressing the LOCK key alone. Either mode is removed by pressing the LOCK key. When the lock mode is removed, the LED is off.

The keyboard generates all 128 ASCII codes. The 15 function keys generate 60 different, unique codes (15 codes each in lowercase, shift (uppercase), control, and shift and control). Special codes are also generated by the cursor control keys.

Keyboard decoding is performed on the computerboard by a ROM. Thus, the keyboard can easily be remapped by replacing the ROM with a ROM for the configuration desired.

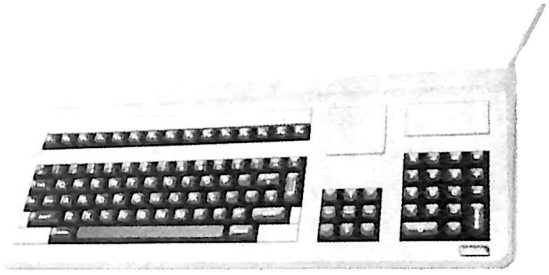


Figure 1-11. Keyboard

COMPATIBILITY

The BASIS 108 computer is hardware and software compatible with the Apple II computer. Operating systems and languages designed for use with the Apple II can be used with the BASIS 108. CP/M™, Pascal, and programs designed for use under these operating systems can also be used. Most peripheral cards designed for use with the Apple II computer can be used with the BASIS 108, except those specifically designated for slots zero (0) or one (1). These slots are not required for the BASIS 108 as they are for the Apple II. Cards or devices that require modification of the computerboard are not recommended. Most peripheral cards are not needed with the BASIS 108.

SECTION 2

BASIS 108 SET-UP

UNPACKING THE 108

If you have not already done so, carefully open the shipping container so that it is not damaged. Please save the packing materials. The packing materials should be used if you wish to ship or move the system, as they provide protection against damage to the system during shipping.

An inventory list is printed on the reverse side of the BASIS 108 Warranty card. You should check the equipment and materials in the shipping container to ensure that you have received all items listed on the inventory. The inventory is dependent upon the system configuration ordered. If any item is damaged or missing, contact your dealer immediately. Your dealer will take the actions necessary to correct the problem.

Video Monitor

To operate the BASIS 108 system you will also need a video monitor or, if you don't mind working in the 40 character display mode, a Television Set (TV) with a video input. (A normal TV set does not have the bandwidth or resolution to display more than 40 characters per line clearly.)

If you do not have a monitor and your TV set does not have a video input, then you will need a video modulator. When a VHF or UHF modulator is installed in your BASIS 108 system, it modulates the video output signal at a standard TV channel frequency so that the video signal may be used with a standard TV. The TV channel used depends upon the type of modulator you have purchased. Read the instructions provided with the modulator for channel tuning instructions for your TV. Remember, when a television is used, the quality of the display will not be as good as the display on a video monitor.

If you require a high resolution color display, a high resolution analog input RGB monitor can be directly connected to your BASIS 108.

Disk Mounting Brackets

If your system was not supplied with disk drives, you can mount your drives in the brackets provided for that purpose. Figure 2-1 shows the brackets and Figure 2-2 shows a drive mounted in these brackets. Information on how to mount your drives is located in the **DISK DRIVES** portion of this section.

CONNECTING THE SYSTEM

Your BASIS 108 system is very easy to connect and set up. The rest of this section provides the information you need to get your system up and running.

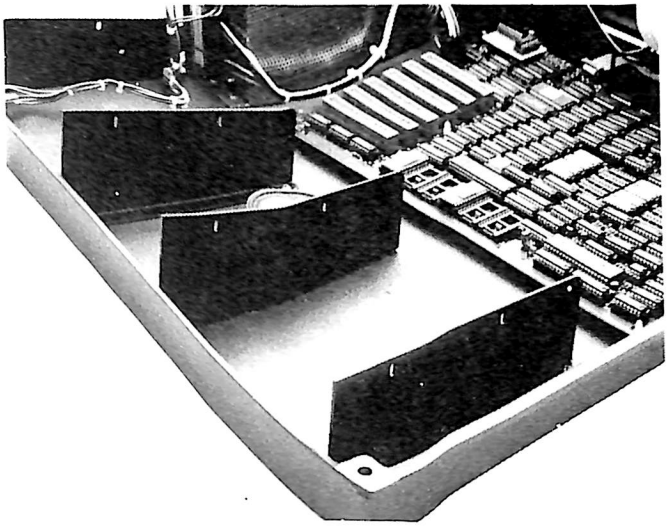


Figure 2-1. Disk Mounting Brackets

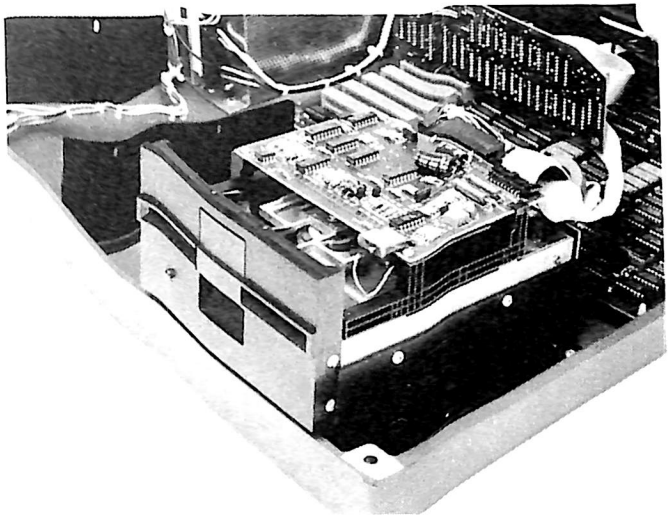


Figure 2-2. Mounted Disk Drive

External Connections

There are three connections you must make. They are POWER, VIDEO OUTPUT, and KEYBOARD. They are normally made on the computer rear panel. Refer to Figure 2-3 for the location of these connectors.

The POWER input is located on the lower right corner below the utility outlets. Insert the end of the supplied power cord in this connector.

VIDEO OUTPUT is provided at either of two BNC connectors or the DE-9 connector. If a video modulator is being used, refer to the **Internal Connections** portion of this section. The top BNC connector (B/W) provides monochrome video output. The bottom BNC connector (NTSC) provides color video output and should not be connected to a monochrome monitor. The DE-9 output is used for RGB monitors. Use the video output desired. Figure 2-3. shows the output connected to the B/W output. As shown in Figure 2-3, the keyboard is connected to the top, KEYBOARD, DB-25 connector.

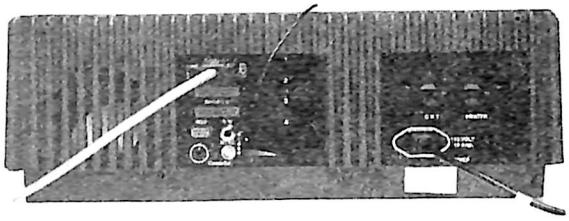


Figure 2-3. Rear Panel Connections

Internal Connections

Several internal connections should be checked to ensure proper connection. These connections are found on the computerboard inside the chassis. Figures 2-4 and 2-5 show the location of these connectors.

On the right side of the board, towards the rear, is the SPEAKER and CASSETTE connector. The plug for this connector has wires running to the CASSETTE DIN connector on the rear panel and to the speaker.

On the right rear of the computerboard are three 26-pin jacks. Ribbon cables from the rear panel DB-25 connectors are connected to these jacks. The right most jack is connected to the SERIAL I/O, bottom, DB-25 connector. The center 26-pin jack is connected to the PARALLEL OUTPUT, middle, DB-25 connector. The left most 26-pin jack is connected to the KEYBOARD, top, DB-25 connector.

Approximately three inches to the left of the 26-pin keyboard jack are the B/W video 2-pin molex connector and the 10-pin jack for the RGB output. The rear pin of the 2-pin molex connector is ground. The wire from the top BNC connector (B/W) is connected here. If you should remove this wire for any reason, make certain the shield or bare copper wire is connected to the rear pin when reconnecting this wire.

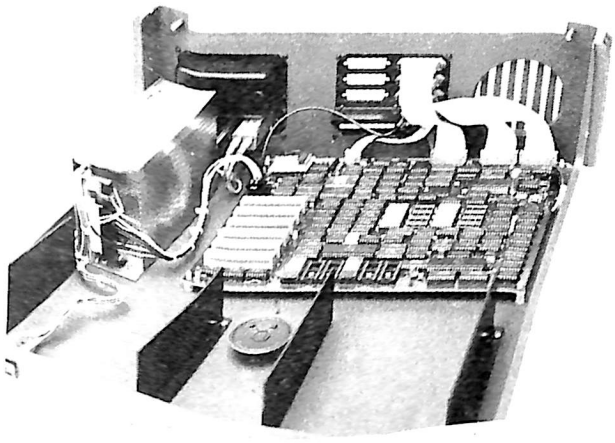


Figure 2-4. Internal Connections, Right View

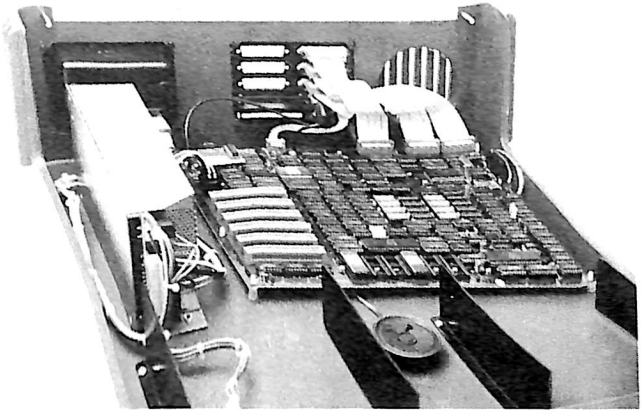


Figure 2-5. Internal Connections, Left View

Internal Connections (Continued)

The three connectors located on the left rear side of the computerboard are shown in Figure 2-6. The 4-pin molex connector provides the NTSC color video output. The wire from the bottom BNC connector is connected to the two rear pins of this 4-pin molex connector. Most standard video modulators can be directly connected to this output. If you should remove the wire from the BNC connector to this jack for any reason, make certain the shield or bare copper wire is connected to the rear or ground pin when reconnecting this wire.

The GAME I/O connector is located between the 4-pin molex jack and the computerboard POWER connector. Game paddles, a joystick, or other devices requiring this interface can be connected here.

The POWER connector is used to connect the power supply to the computerboard. This connector is keyed, allowing the power jack to be connected only in the proper way.

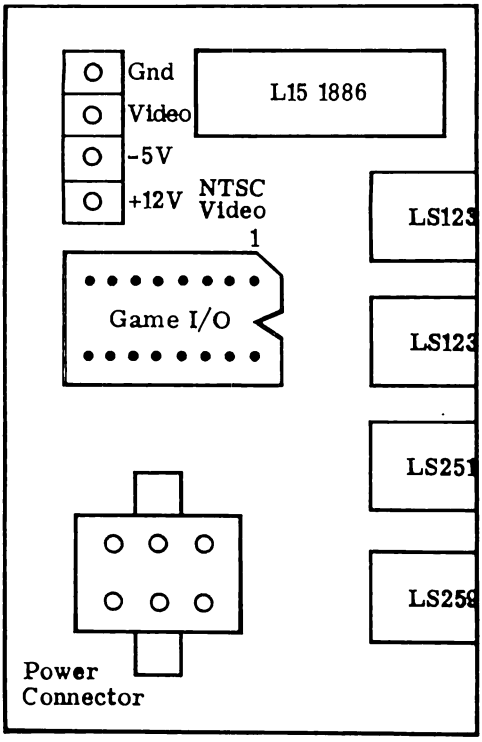


Figure 2-6. NTSC, External I/O, and Power Connectors

DISK DRIVES

Two 5 1/4" drives can be mounted in the computer chassis. The brackets shown in Figure 2-1 are used for this purpose.

Compatibility

Drives compatible with Apple™ formatted diskettes, or other drives with an appropriate controller can be used with the system. The provided brackets are configured for mounting drives using the standard Shugart SA-400 side mounting hole dimensions and placement.

Mounting Drives

If you received your BASIS 108 system without drives, you can mount your drives in the provided brackets. If your drives have cabinets or cases it is necessary to remove the cabinets or cases before they can be mounted in the brackets. Figures 2-7 and 2-8 show typical locations of screws that must be removed in order to remove the drive cabinet. Figure 2-7 shows a drive with screws on the sides. Not all drives have side screws. Figure 2-8 shows screws on the bottom of a drive. Most drives have four or more screws on the bottom of the cabinet. Some drives have a bottom plate which should also be removed after the wrap-around portion of the cabinet is removed.

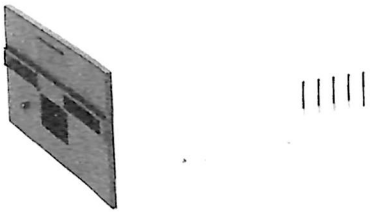


Figure 2-7. Drive Side View

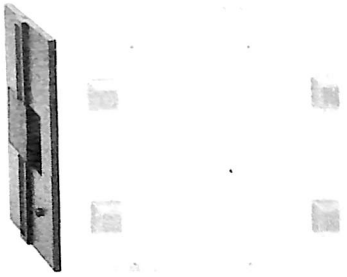


Figure 2-8. Drive Bottom View

Mounting Drives (Continued)

Place the drive between a set of brackets and insert screws in the four mounting holes. Lightly tighten the four screws. If necessary, loosen the mounting bracket screws and adjust the front-to-back position of the drive and brackets so that the front panel of the drive is flush with the front of the BASIS 108 cover when it is in place. Move the drive up or down in the bracket as necessary so that it is centered in the cover opening. Tighten all screws.

After mounting the drives, connect them to the controller card. Your drive manual should provide the information necessary for this connection. Normally the controller card for these two drives is located in Slot 6. Figure 2-9 shows a controller card placed in this Slot 6.

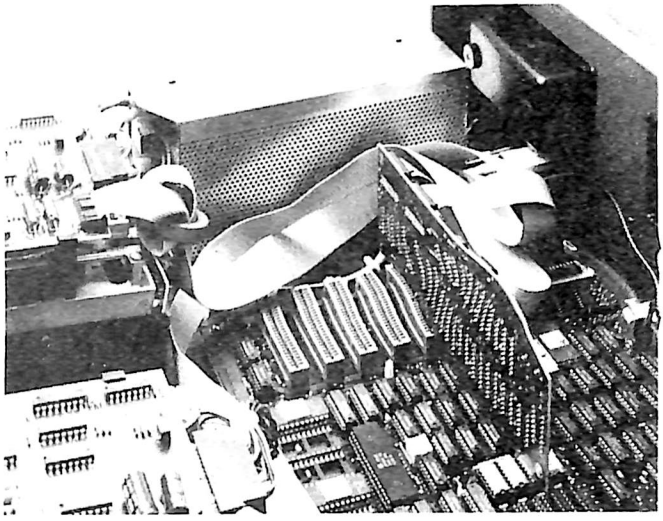


Figure 2-9. Drive Controller Card

Drive Checkout

Follow the drive manufacturer's instructions provided in your drive manual for familiarization and checkout of your disk drives.

SECTION 3

BASIS 108 OPERATION

BASIS 108 OVERVIEW

The BASIS 108 is a dual processor computer. Both the 6502 and Z-80 microprocessors are supported for full, compatible operation in one computer. All circuitry except the power supply is located on a single computerboard. In this section each of the blocks shown on Figure 3-1, BASIS 108 Block Diagram, are discussed. Detail on system operation is provided. The BASIS 108 schematic and board layout in Appendix K should be referenced while reading this section. This section will also refer to other sections and appendices in the manual where further detail can be obtained.

TIMING

System timing is derived from the 14.318 MHz crystal located at A8 on the computerboard. This signal is divided and further used to generate all remaining timing signals except for the 3.5795 MHz Color Reference signal for the NTSC video output. The 3.5795 MHz signal is obtained from the crystal located at L17.

Table 3-1. Timing Signals

SIGNAL	DESCRIPTION
14M	Master oscillator output. Used to derive other timing signals.
7M	7.159 Mhz timing signal.
ø0	1.023 MHz Phase 0 system clock. Compliment to ø1. Sometimes referred to as ø 2 in other literature.
ø1	1.023 MHz Phase 1 system clock. Compliment to ø0.
Q3	General purpose timing signal. Twice the frequency of the system clocks, but asymmetrical.

Video control and addressing signal are also generated by this circuitry. Video generation consists of 192 scan lines on the video screen. These are grouped into 24 lines of eight scan lines each. Each scan line can display up to 80 bytes of memory. Fifteen synchronization signals are used. They consist of "H" (horizontal) and "V" (vertical) groups of signals. The H0 through H5 signals are used to define displayed position on a line. The V0 through V4 signals are used to define the vertical line position on the screen. The VA through VC signals are used to define the vertical scan line position within the vertical screen line.

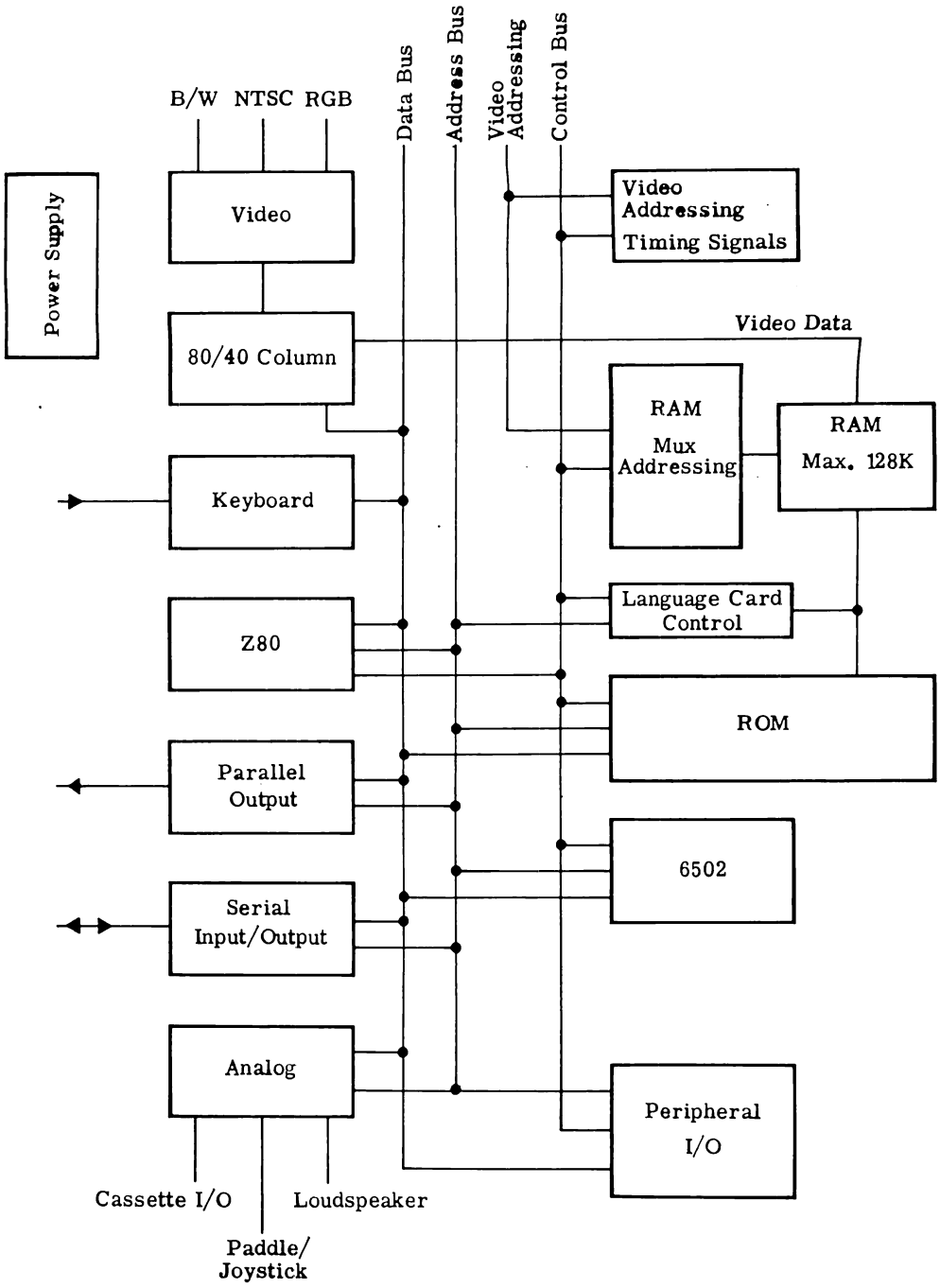


Figure 3-1. BASIS 108 Block Diagram

Language Card Area (Continued

There are two additional soft switches provided for use with this memory. These switches are used to prohibit the switching between LC RAM and ROM. Effectively they make the system appear to have no language card area. Writing to $C00F causes the system not to recognize the $C08X soft switches. If the write operation is performed when the $D000 - $FFFF RAM range is enabled, switching back to ROM is impossible until a write is performed to $C00E. Writing to $C00E or resetting the system causes the $C08X soft switches to be recognized again.

ROM/EPROM Selection

There are six ROM sockets on the board between locations E1 and J1. These can be configured to accept 2716 EPROMs or 2513 ROMs (the type supplied by Apple). As delivered, the system has two 2716s on the computerboard, located in the sockets F8 and E0. Thus, the board is configured for use with 2716s. Either all 2716s or all 2513s must be used. The types cannot be mixed. The Jumpers used to specify the type of device are located at positions D4 and E1. Figure 3-3 shows the jumper configurations for 2716 and 2513 operation.

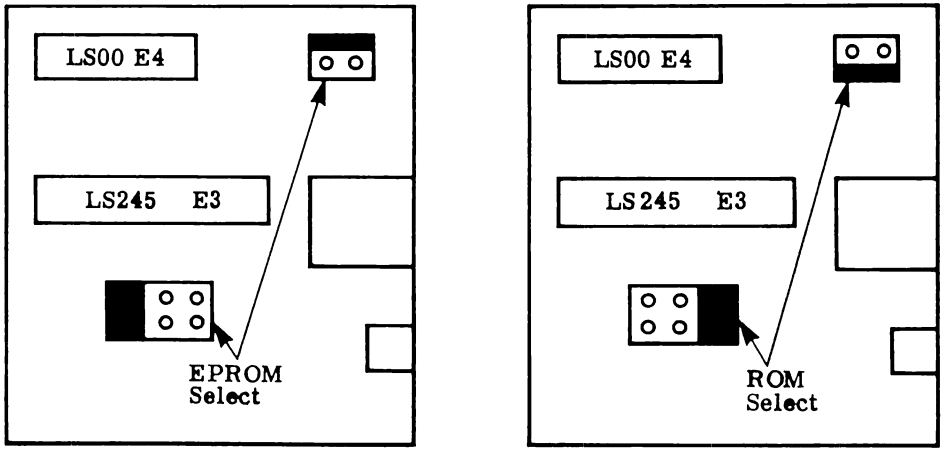


Figure 3-3. ROM/EPROM Jumper Positions

6502

The 6502 is a dynamic microprocessor operating at 1.023 MHz clock rate. It uses the address and data busses only when the ø0 signal is high or active. When ø0 is low, the microprocessor is doing internal operations and does not need the address and data busses. For further information about the 6502 refer to Appendix E.

Z-80

The Z-80 microprocessor is located on the computerboard and is buffered to the Address and Data Buses. The Z-80 is configured to allow direct execution of 8080 and Z-80 programs and operate under the CP/M Operating system. The computer supports the 56k CP/M configuration. Further information about the Z-80 is located in Appendix G.

Timing

The Z-80 is synchronized and phase locked to the 6502 clocks. During the video refresh period, ø1, the 7 MHz clock is divided into three half clock periods of 135 nanoseconds. The first half-clock is high, the second is low, and the third is high. At the end of the third half clock the signal goes low and remains low until the start of the next ø1. Thus, the Z-80 clock is low during all of ø0 and a small part of ø1. The fourth half clock is normally 563 nsec long. It is stretched by 69 nsec at the end of each video line. This timing scheme creates an effective Z-80 clock rate of 2.041 MHz.

Each type of machine cycle contains one memory access period during ø0. The read/write signal is synchronized ensuring that the write can only go low during the time the Z-80 clock is high. Because all address transitions from the Z-80 occur when the clock is high, they must all occur during ø1 while the video update accesses are occurring. Thus, each ø0 has stable addresses on the bus for the entire duration of the cycle.

Control

The Z-80 is controlled by write commands to the area of memory that normally contains peripheral ROM. It is necessary to use write commands to ensure that the 6502 will not perform two accesses in succession, which would prevent switching back to the 6502.

When the BASIS 108 power is turned on, the RESET signal forces the Z-80 to the off state. The RESET signal is synchronized to the system clock to ensure that a write operation cannot be interrupted. The Z-80 is immediately placed in the WAIT mode and remains in this mode until activated.

Upon receipt of a write to the $C100-$C1FF area of memory, the Z-80 is enabled. The Z-80 remains in the WAIT mode until one memory cycle occurs with Z-80 address information on the bus. At this point the Z-80 is allowed to run with no further wait cycles required. Receipt of another write to the $C100-$C1FF area of memory, by the Z-80, will return the Z-80 to the WAIT mode.

Address Bus Interface

The Z-80 Address Bus interface is constructed such that memory conflicts that exist between the 6502 based system and the conventions used by the Z-80 microprocessor and CP/M are resolved. There are four switches located on the computerboard at location G6. Switch 1 when in the OFF position adds $1000 to the Z-80 addressing. This effectively shifts the Z-80 interrupt addresses and CP/M starting addresses out of the 6502 zero page memory. In addition, addresses in the $C000-$EFFF memory range are shifted to allow apparent contiguous memory for CP/M operation. Table 3-4 shows how memory appears to the Z-80 versus normal 6502 configuration when Switch 1 is in the OFF position.

Table 3-4. Z-80 vs. 6502 Addressing

Z-80 Addresses	6502 Addresses
$0000-$0FFF	$1000-$1FFF
$1000-$1FFF	$2000-$2FFF
$2000-$2FFF	$3000-$3FFF
$3000-$3FFF	$4000-$4FFF
$4000-$4FFF	$5000-$5FFF
$5000-$5FFF	$6000-$6FFF
$6000-$6FFF	$7000-$7FFF
$7000-$7FFF	$8000-$8FFF
$8000-$8FFF	$9000-$9FFF
$9000-$9FFF	$A000-$AFFF
$A000-$AFFF	$B000-$BFFF
$B000-$BFFF	$D000-$DFFF
$C000-$CFFF	$E000-$EFFF
$D000-$DFFF	$F000-$FFFF
$E000-$EFFF	$C000-$CFFF
$F000-$FFFF	$0000-$0FFF

Note that the Z-80 can address contiguous memory from $0000-$DFFF, without accessing the 6502's zero page of memory or the peripheral I/O memory area.

When Switch 1 is turned ON, the Z-80 addresses memory in an unaltered form. The same as memory appears to the 6502.

DMA

The Z-80 is included in and supports full BASIS 108 DMA daisy chain operation. When Switch 2 is ON a higher priority DMA device can cause the Z-80 to relinquish control of the bus. When Switch 2 is OFF the Z-80 will not relinquish control of the bus. The Z-80 is always the lowest priority device.

Interrupts

Interrupts can be recognized by the Z-80 as well as the 6502. When Switch 4 is ON, the Z-80 will respond to interrupts occuring in the BASIS 108. The interrupt handler routine should not attempt to service the interrupt. Control should be passed back to the 6502 for processing. The 6502 sees all interrupts. Passing interrupt servicing back to the 6502 allows the 6502 to clear itself of the interrupt status.

Switch 3 performs the same function for the non-maskable interrupt.

PERIPHERAL I/O

Along the front left side of the board are six Peripheral I/O connectors. These connectors are used for Peripheral Interface boards designed for the BASIS 108 or the Apple II computer. Peripheral Board I/O space has been set aside for use with each of the six slots and there is also a 2k common area for use by any of the boards installed.

Each of the slots is numbered, with the frontmost slot being "Slot 2". There are no slots 0 and 1. Built-in features (language card, parallel printer port, and serial interface) use the memory space allocated for slots 0 and 1.

Each of the six 50-pin connectors is individually selected by control circuitry. Table 3-5 provides detail on the signals available at these connectors. The load drive capability specified is for each slot. The pinout is shown in Figure 3-4.

Table 3-5. Peripheral I/O Signal Descriptions

PIN	SIGNAL	DESCRIPTION
1	I/O SELECT	This signal is normally high. It becomes low during ø0 when a reference is made to $CnXX, where n is the slot number. This signal will drive 10 LSTTL loads.
2-17	A0 - A15	The buffered address bus. The address on these pins becomes valid during ø1 and remains valid through ø0. These signals will each drive 5 LSTTL loads.
18	R/W	Buffered Read/Write signal. This signal becomes valid the same time the address bus does, and goes high during a read cycle and low during a write cycle. This signal will drive 2 LSTTL loads.
20	I/O STROBE	Normally high, this line goes low during ø0 when the address bus contains an address between $C800 and $CFFF. This signal will drive 4 LSTTL loads.
21	RDY	The 6502's RDY input. This signal going low during ø1 halts the 6502 with the address bus holding the address of the current location being fetched.

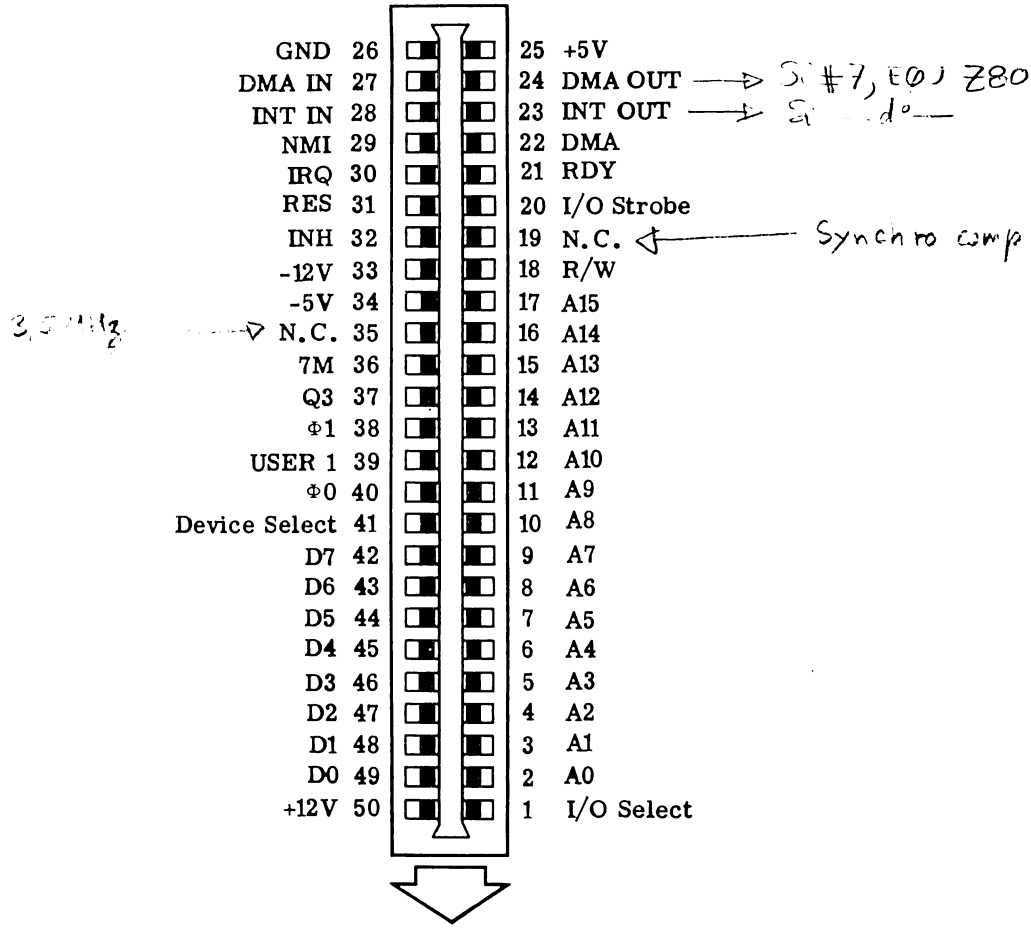


Figure 3-4. Peripheral Connector Pinout

Table 3-5. Peripheral I/O Signal Descriptions (Continued)

PIN	SIGNAL	DESCRIPTION
22	DMA	Pulling this line low disables the 6502's address bus and halts the 6502. The line is held high by a 1 kOhm resistor to +5v.
23	INT OUT	Daisy-chain interrupt output to lower priority devices. This pin is usually connected to pin 28 (INT OUT). INT OUT for slot 7 is connected to the Z-80.
24	DMA OUT	Daisy-chain DMA output to lower priority devices. This pin is usually connected to pin 27 (DMA IN). DMA OUT for slot 7 is connected to the Z-80.
25	+5 volts	+5 volt power supply. 3 Amps are available for all peripheral slots.
26	Ground	System electrical ground.
27	DMA IN	Daisy-chain DMA input from higher priority devices. Usually connected to pin 24 (DMA OUT).
28	INT IN	Daisy-chain Interrupt input from higher priority devices. Usually connected to pin 23 (INT OUT). INT IN for slot 2 is connected to the Keyboard Interrupt circuitry.
29	NMI	Non-Maskable Interrupt. When this line is pulled low the BASIS 108 begins an interrupt cycle and jumps to the interrupt handling routine location $3FB.
30	IRQ	Interrupt Request. When this line is pulled low the BASIS 108 begins an interrupt cycle if the 6502's Interrupt disable (I) flag is not set. If executed the 6502 will jump to the interrupt handling routine whose address is stored in addresses $3FE and $3FF.
31	RES	When this line is pulled low the 6502 begins a RESET cycle.
32	INH	When this line is pulled low the 12k address range of $D000 - $FFFF is disabled on the computerboard. This line is held high by a 1 kOhm resistor to +5 volts.
33	-12 volts	-12 volt power supply, 0.5 Amps is available for all peripheral slots.
34	-5 volts	-5 volt power supply, 0.5 Amps is available for all peripheral slots.
36	7M	7 MHz clock. This signal will drive 2 LSTTL loads.

Table 3-5. Peripheral I/O Signal Descriptions (Continued)

PIN	SIGNAL	DESCRIPTION
37	Q3	Asymmetrical 2 MHz clock. This signal will drive 2 LSTTL loads.
38	ø1	Phase 1 clock. This signal will drive 2 LSTTL loads.
39	USER 1	When this line is pulled low, all $CXXX address decoding is inhibited.
40	ø0	Phase 0 clock. This signal will drive 2 LSTTL loads.
41	DEVICE SELECT	This signal becomes active, low, on a connector when the address bus holds an address between $C0n0 and $C0nF, where n is the slot number plus $8. This signal will drive 10 LSTTL loads.
42-49	D0 - D7	Buffered bidirectional data bus. The data on this line becomes valid 300 nsec into ø0 on a write cycle, and should be stable no less than 100 nsec before the end of ø0 on a read cycle.
50	+12 volts	+12 volt power supply, 2.5 Amps is available for all peripheral slots.

Card I/O Space

Each slot is allocated sixteen address locations for its use. Each card can use these locations as it needs them. Whenever the DEVICE SELECT signal on pin 41 of a peripheral connector goes low, one of the allocated addresses is being referenced. The card in that slot can look at the lower four address lines to determine which of the addresses is being referenced. Table 3-6 list the address allocations.

Table 3-6. I/O Space Allocations

SLOT	ADDRESSES
2	$C0A0 - $C0AF
3	$C0B0 - $C0BF
4	$C0C0 - $C0CF
5	$C0D0 - $C0DF
6	$C0E0 - $C0EF
7	$C0F0 - $C0FF

I/O ROM Space

Each peripheral slot also has one page (256 addresses) reserved for use by its ROM. The memory page reserved is $Cn, where n is the slot number. Thus, slot 2 has the address range $C200 - $C2FF for its use. The I/O SELECT signal for a slot goes low when a memory reference is made to an address within a slot's reserved space. Peripheral cards can use this signal to enable their ROMs, and use the lower eight address lines to address each byte of the 256 bytes.

I/O Expansion ROM

The 2k memory range from $C800 - $CFFF is reserved for a 2k ROM on a peripheral card. This space is available to all peripheral slots, and more than one card can have an expansion ROM. However, only one expansion ROM can be active at any time.

When a peripheral card is enabled and a reference is made to the $C800 - $CFFF memory range the I/O STROBE signal on pin 20 of the card goes low. At this time the card can place its expansion ROM on line. The driving routine should reference address $CFFF as a signal to other cards to relinquish control and turn off.

I/O Scratchpad

Each of the slots has eight bytes of allocated RAM. The card in that slot can use these addresses for temporary storage of data. The addresses are listed in Table 3-7.

Table 3-7. I/O Scratchpad Addresses

SLOT 2	SLOT 3	SLOT 4	SLOT 5	SLOT 6	SLOT 7
$047A	$047B	$047C	$047D	$047E	$047F
$04FA	$04FB	$04FC	$04FD	$04FE	$04FF
$057A	$057B	$057C	$057D	$057E	$057F
$05FA	$05FB	$05FC	$05FD	$05FE	$05FF
$067A	$067B	$067C	$067D	$067E	$067F
$06FA	$06FB	$06FC	$06FD	$06FE	$06FF
$077A	$077B	$077C	$077D	$077E	$077F
$07FA	$07FB	$07FC	$07FD	$07FE	$07FF

POWER SUPPLY

The BASIS 108 power supply is a switching type supply. It is located in the left rear of the chassis. Built-in circuitry provides overvoltage and undervoltage as well as overload protection. Specifications on the supply are located in Appendix I.

POWER SUPPLY (Continued)

The supply is connected to the computerboard with a keyed connector. The connector pinout is shown in Figure 3-5.

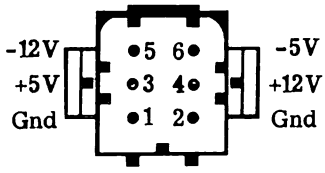


Figure 3-5. Power Supply Connector

INPUT/OUTPUT

The BASIS 108 has many Input/Output features. With the exception of the External I/O located at computerboard location L13, they are available at the rear panel. Space is also provided to extend peripheral I/O card capability to the rear panel. Figure 3-6 shows the portion of the rear panel devoted to I/O connections. The pinout of each connector is shown.

Section 6 of this manual is devoted to the detail of these devices and their capabilities.

VIDEO

Section 5 of this manual is devoted to the capabilities of the video section. Figure 3-6 shows the location of the video output connectors.

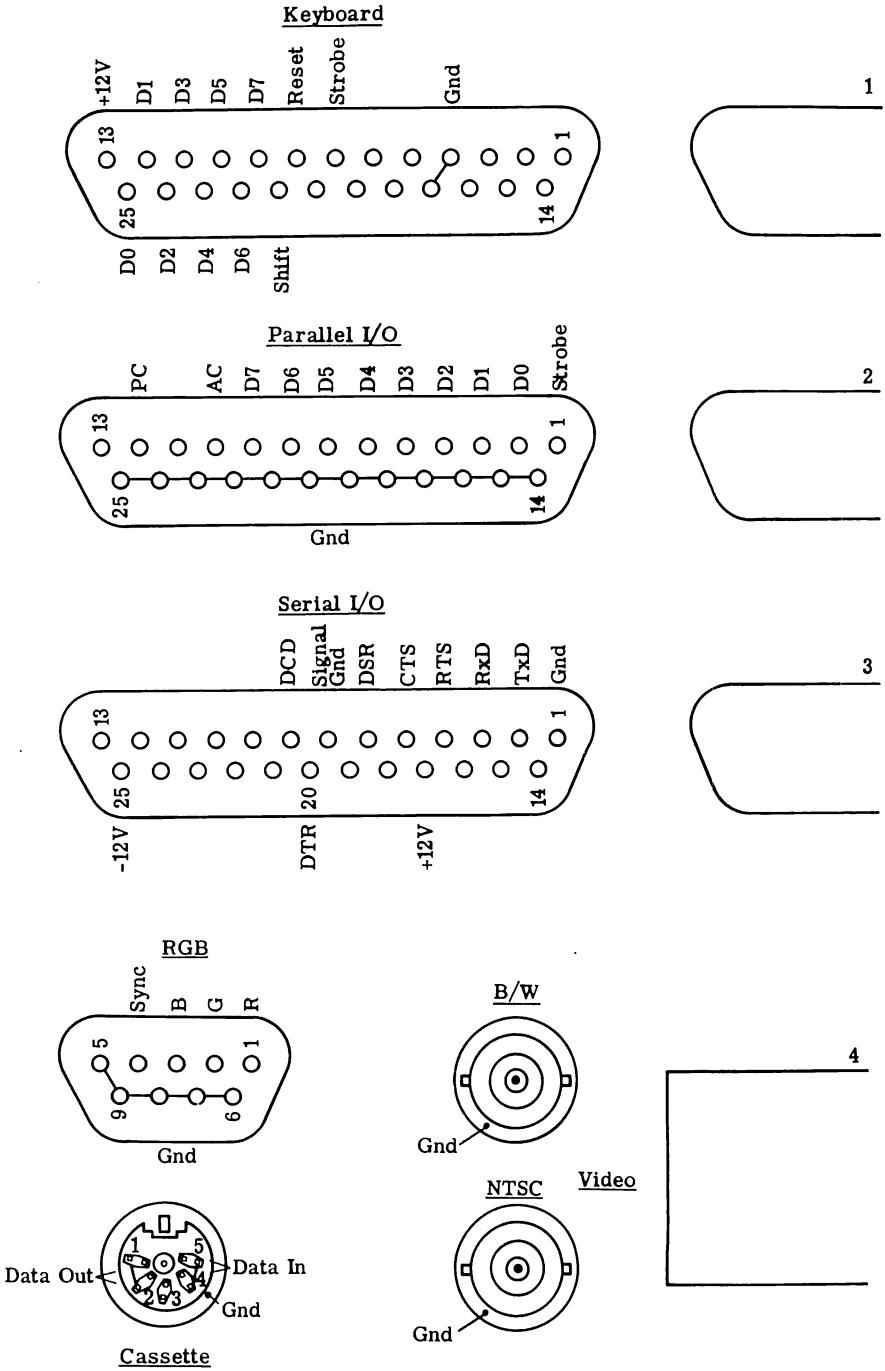


Figure 3-6. Rear Panel Connectors

SECTION 4

SOFTWARE

OPERATING SYSTEMS

The BASIS 108 runs under the operating systems originally designed for use with the Apple II computer. Disk Operating System (DOS) 3.3, Apple Pascal, Pascal IV, and CP/M are the major operating systems that can be used with the BASIS 108. Basically, any operating system designed for use with the Apple II and those designed for use with the BASIS 108 can be used.

This section further defines the use of these operating systems and languages.

Compatibility

The BASIS 108 will work with the operating systems and languages designed for use with the Apple II computer.

Considerations

Most existing operating systems do not know how to handle the built-in and advanced features of the BASIS 108 system. To allow full use of these features it is necessary to modify some operating system and language parameters.

Modifications

Operating system and language modifications are accomplished with the BASIS BOOTER DISKETTE. This diskette running under the Pascal Operating System performs upgrades to existing software to take advantage of all BASIS 108 features. Full instructions on its use are included in Appendix A. This diskette modifies Apple Pascal, CP/M, Applesoft, and Integer Basic. Pascal IV as provided by BASIS is fully capable of using all BASIS 108 features.

DOS 3.3

The DOS PATCH program is provided on the BASIS UTILITY DISKETTE. This program provides an easy method to modify DOS to allow it to accept lowercase entry and the ability to accept IN#9 and PR#9 commands.

PASCAL

All Apple Pascal changes are made using the ZAP program on the BASIS BOOTER DISKETTE. Full documentation is contained in Appendix A.

CP/M

All Microsoft CP/M changes are made using the ZAP program on the BASIS BOOTER DISKETTE. Full documentation is contained in Appendix A.

SECTION 5

VIDEO

VIDEO MODES

The BASIS 108 displays text, and Low Resolution (Lo-Res), Medium Resolution (Me-Res) and High Resolution (Hi-Res) graphics. When Lo-Res, Me-Res or Hi-Res color graphics are show on a monochrome monitor they are shown in gray scale. There are 16 levels of gray scale from black to white. Two memory regions are used for the display of text, Lo-Res, and Me-Res graphics. Two additional regions are used for the display of Hi-Res graphics.

The text screen can display either an 80, or a 40 column by 24 line character display, depending on the mode chosen. The same memory regions are also used for the Lo-Res and Me-Res graphics. In the Lo-Res graphics mode a 40 X 48 block matrix is displayed. In the Me-Res graphics mode an 80 X 48 block matrix is displayed. Lo-Res and Hi-Res graphics are displayed in 16 colors.

Additional memory is allocated for two Hi-Res graphics pages with a resolution of 280 X 192 dots. High Resolution graphics is displayed in six colors.

The system is provided with a color generation ROM which provides NTSC colors as listed in Tables 5-2, 5-3, 5-6, and 5-7. The RGB color output is not changeable.

Video Outputs

The BASIS 108 has three video outputs, Monochrome, NTSC Color, and RGB. Any combination or all of these outputs can be used simultaneously. Each of these outputs is described in the follow sections. The rear panel connectors are shown in Figure 5-1.

Monochrome Video Output

This output should be used with high resolution monochrome monitors with a video input. This signal is an EIA RS-170 video signal. It does not contain the NTSC color burst signal which would appear as noise which causes video degradation on a high quality monochrome monitor. This output is available on the rear panel from the top BNC connector (B/W).

Color Video Output

This output provides an NTSC color video output for color a video monitor or a color TV with a video input. This output is available on the rear panel from the bottom BNC connector (NTSC).

RGB Video Output

This output provides analog Red, Green, Blue (RGB) and a composite synchronization signal. When used with an RGB monitor this output provides the best color signal available. The RGB output is available on the rear panel at the DE-9 connector. The R, G, and B signals are positive, and the synchronization signal is negative.

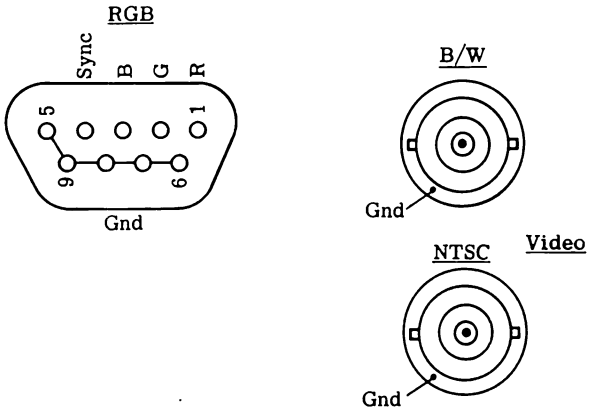


Figure 5-1. Rear Panel Video Connectors

TEXT MODE

The text screens are located in memory between addresses $0400 and $0BFF. This memory area is divided between the two text pages. PAGE 1 is from $0400 to $07FF and PAGE 2 is from $0800 to $0BFF. Selection of the PAGE to be displayed (1 or 2) is made by means of the soft switches $C054 (PAGE 1) and $C055 (PAGE 2). The normal BASIS 108 display is 80 columns.

80 Column Operation

A very simple method has been implemented for 80 column display. Static RAM is placed in parallel with the dynamic RAM on the computerboard. The static RAM and dynamic RAM are referenced at the same addresses. Soft switches are used to select which of the RAMs is referenced. $C00C activates the dynamic RAM and $C00D activates the static RAM.

These two soft switches are used to switch back and forth between the different RAMs. This technique requires that all characters with an even column number be located in dynamic RAM and that all odd column characters be located in static RAM. An additional soft switch must be used to produce the actual 80 character display on the screen. $C00B switches the 80 column display on and $C00A switches it off. The static RAM can still be referenced, write or read, independently of the 80 column on and off switches.

80 Column Operation (Continued)

The dynamic memory can be read only during the time when the microprocessor is not accessing memory. This occurs when the microprocessor's clock pulse is low (logical 0). At this time screen characters are read from memory. Normally, in order to display 80 columns on a line, a screen character would also have to be read while the microprocessor is referencing memory. Since this is very difficult to accomplish, the BASIS 108 reads two characters simultaneously. One character is read from dynamic RAM and one is read from static RAM. These characters are placed in a buffer where they can be further processed independently of the microprocessor's clock timing.

Software routines in the monitor support normal 80 column operation. Thus, normal 80 column operation is transparent to the operating system.

40 Column Operation

Generation of a 40 column display is provided to allow use of a TV and for compatibility with existing software which cannot properly use an 80 column display. Three options are available for 40 column display. First, the FP80, TEXT 40 command can be used with the 25-120 monitor. Second, the language card area of RAM can be loaded with a 40 column version (25-46) of the system monitor. Third, a text handling routine outside of the monitor can be implemented. Any of these methods can be used to allow full 40 column operation of your BASIS 108.

Figure 5-2 is a memory map of the text display area. The map shows the 40 columns and 24 rows as they are displayed in 40 column display mode. To obtain the address of any byte, add the row and column of the byte in question. In the 80 column display mode each of the boxes on this map represents two characters, one in dynamic memory and one in static memory. The map shows page 1 addresses, for page 2 addresses add 1024.

Text Soft Switches

Table 5-1 lists the soft switches normally used with the text display. Soft switches are provided for page selection, 40 or 80 column operation, and selection of dynamic and static RAM.

Table 5-1. Text Soft Switches

HEX ADDRESS	FUNCTION
$C051	Select text mode
$C054	Select Text Page 1
$C055	Select Text Page 2
$C00A	80 character display OFF
$C00B	80 character display ON
$C00C	Select Dynamic RAM
$C00D	Select Static RAM

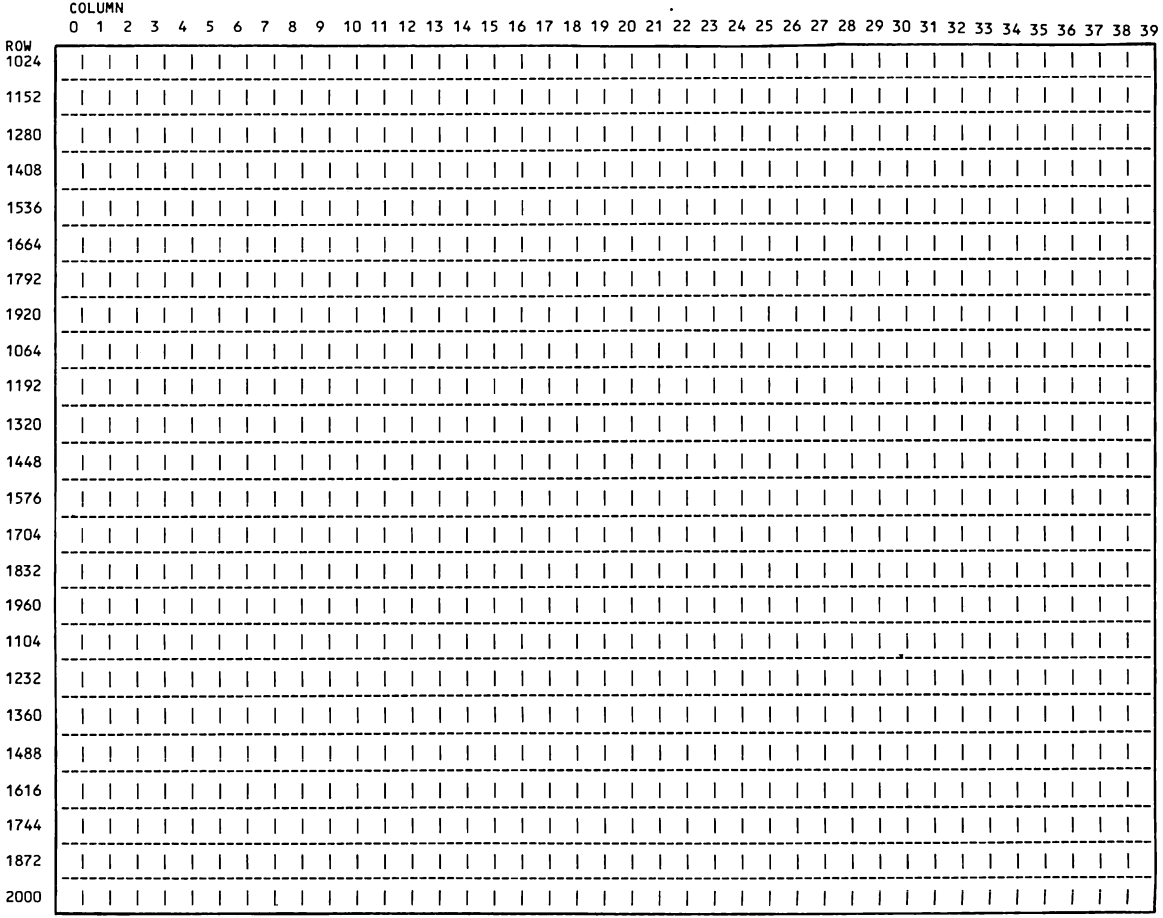


Figure 5-2. Text Memory Map

GRAPHICS MODES

Three forms of graphics are supported: Lo-Res, Me-Res and Hi-Res. These graphics modes and their various options are discussed in the following sections. Color output is obtained from the NTSC and RGB outputs.

HIGH RESOLUTION GRAPHICS

When the BASIS 108 is in the High Resolution mode it displays dots in a matrix of 280 dots (width) by 192 dots (height). There are two separate areas or pages that can be displayed. They are located in memory in the address ranges $2000 to $3FFF and $4000 to $5FFF. High Resolution graphics can be displayed in two modes: Full screen graphics (280 X 192 dots) and mixed graphics (280 X 160 dots). When in mixed graphics, four lines of text are displayed on the bottom four lines of the display. These four lines of text may be either 40 or 80 columns as described in the section under text.

Each dot in the display represents one bit from the display memory. Seven of the eight bits in each byte are displayed on the screen, the eighth bit specifies the bit colors, the color of each dot in that byte. On a monochrome display, each display bit is on when it is a "1" and off when it is a "0".

When displayed on a color screen the position of each display bit is important to the color being displayed. When a display bit is "1" in an even bit position, starting with the leftmost edge of the screen being column 0, it can be displayed as either color 1 or 5 as shown in Table 5-2 or 5-3. When a display bit is "1" in an odd bit position, with the rightmost edge of the screen being column 279, it can be displayed as either color 2 or 6 as shown in Table 5-2 or 5-3.

Colors 3 and 7 in Table 5-2 or 5-3 are obtained when two adjacent bits, and even and odd, are both "1". Colors 0 through 3 are displayed when the eigth or undisplayed bit is "0". Colors 4 through 7 are displayed when the eigth bit is a "1". Thus, the undisplayed bit of a byte changes the color selection of a byte. The colors described in Tables 5-2 and 5-3 may vary depending upon the type and quality of video monitor used.

Table 5-2. High Resolution NTSC Colors

COLOR NUMBER	COLOR
0	Black
1	Orange
2	Green
3	Yellow
4	Black
5	Red
6	Blue
7	Lilac

Table 5-3. High Resolution RGB Colors

COLOR NUMBER	COLOR
0	Black
1	Green
2	Red
3	Gold
4	Black
5	Light Blue
6	Magenta
7	Lilac

High Resolution Memory Mapping

Each High Resolution graphics page has 8192 bytes. Each row displays 280 bits or 7 bits from each of the 40 bytes which make up each row. The least significant bit of the first byte in each row is displayed on the left edge of the screen, followed by the second bit, third bit, etc. The eighth bit, most significant bit, of each byte is not displayed. Each row shown in Figure 5-4 contains eight lines as shown in Figure 5-3. Thus, 24 rows with eight lines each, represent the 192 lines on the display screen. Figures 5-3 and 5-4 show a memory map of High Resolution graphics page 1. These tables and the instructions will allow you to locate any address within the mapped area.

To locate the address of any byte in the High Resolution graphics area:

1. Locate the column and row in Figure 5-3.
2. Locate the line in Figure 5-4.
3. Add together the column address, row address, and line address.
4. The address obtained is that of the byte being located on page 1.
5. If the byte being located is on page 2, add 8192.

High Resolution Soft Switches

The soft switches listed in Table 5-4 are used to select the display options available in the High Resolution graphics mode.

Table 5-4. High Resolution Soft Switches

HEX ADDRESS	FUNCTION
$C050	Select graphics mode
$C052	Select full graphics display
$C053	Select mixed graphics display
$C054	Select Page 1
$C055	Select Page 2
$C057	Select High Resolution graphics

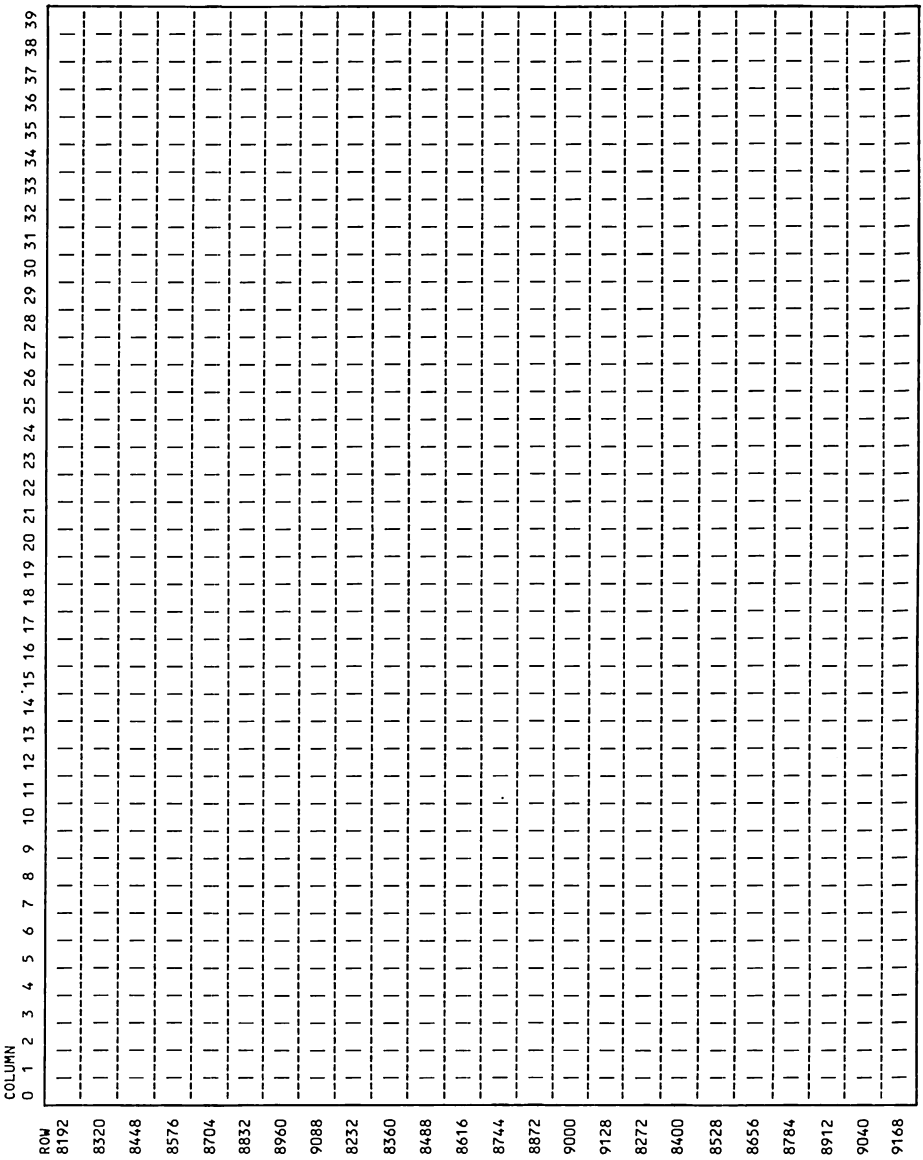


Figure 5-3. High Resolution Graphics Map - Part 1

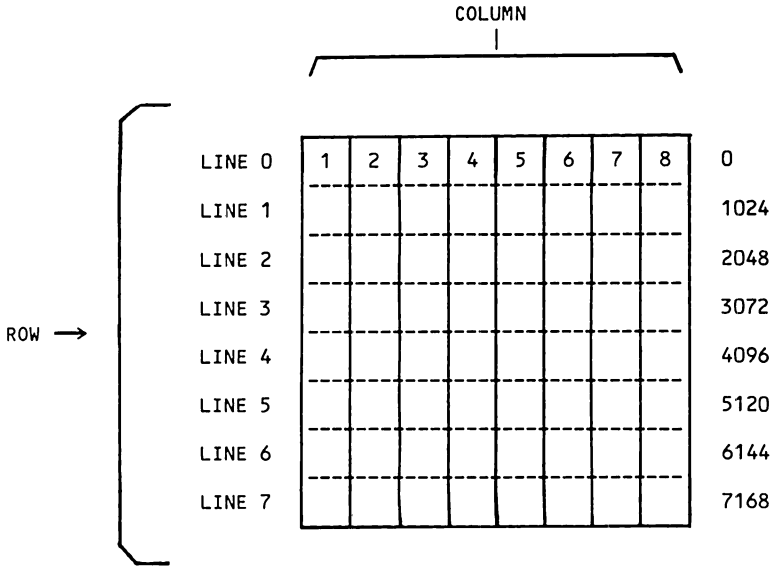


Figure 5-4. High Resolution Graphics Map - Part 2

LOW RESOLUTION GRAPHICS

In the Low Resolution (Lo-Res) graphics mode the BASIS 108 uses the same locations in memory used by the Text mode. A 40 X 48 matrix or 40 X 40 matrix with four lines of 40 column text can be displayed.

Each graphic memory byte is displayed as two colored blocks, not as an ASCII character. These two blocks are stacked one on top of the other (vertically). Each half of the byte or nibble is capable of displaying any of the 16 Lo-Res colors. The lower nibble, least significant 4 bits of the byte, is used to display the upper graphic block and the value in the upper nibble is used to display the lower graphic block. Refer to Figure 5-2, Text Memory Map, for mapping of the Lo-Res graphic area, the addresses used for both are the same.

The soft switches used with Lo-Res graphics are listed in Table 5-5. Tables 5-6 and 5-7 list the colors available for use in both Lo-Res and Me-Res graphics.

Table 5-5. Low Resolution Soft Switches

HEX ADDRESS	FUNCTION
$C050	Select a graphics mode
$C052	Select full graphics display
$C053	Select mixed graphics display
$C054	Select Page 1
$C055	Select Page 2
$C056	Select Low Resolution graphics

Table 5-6. Low Resolution NTSC Colors

COLOR NUMBER	COLOR
0	Black
1	Magenta
2	Dark Blue
3	Purple
4	Dark Green
5	Grey
6	Medium Blue
7	Light Blue
8	Brown
9	Orange
10	Red
11	Pink
12	Light Green
13	Yellow
14	Aquamarine
15	White

Table 5-7. Low Resolution RGB Colors

COLOR NUMBER	COLOR
0	Black
1	Dark Green
2	Medium Green
3	Light Green
4	Red
5	Orange
6	Dark Yellow
7	Light Yellow
8	Royal Blue
9	Medium Blue
10	Light Blue
11	Aquamarine
12	Magenta
13	Pink
14	Light Pink
15	White

MEDIUM RESOLUTION GRAPHICS

Medium Resolution (Me-Res) graphics displays 80 X 48 or 80 X 40 blocks in 16 colors. It has the same capabilities as Lo-Res graphics, except that the Static RAM is used for the 80 block wide display. The same conditions exist for soft switches $C00A through $C00D as when they are used in the Text mode.

Refer to the Text Memory Map in Figure 5-2 to locate the address of any byte in the display. Remember, each column now represents two columns, one each for the static and dynamic memories being used.

Refer to Tables 5-6 and 5-7 for the colors displayed in Me-Res graphics. They are the same as Lo-Res graphics colors.

Table 5-8. Medium Resolution Soft Switches

HEX ADDRESS	FUNCTION
$C00A	80 character display OFF
$C00B	80 character display ON
$C00C	Select Dynamic RAM
$C00D	Select Dynamic RAM
$C050	Select a graphics mode
$C052	Select full graphics display
$C053	Select mixed graphics
$C054	Select Page 1
$C055	Select Page 2
$C056	Select Me-Res graphics (must be in 80 columns)

CHARACTER GENERATION

Character generation in the BASIS 108 system is based on a 2532 EPROM. This EPROM can hold up to five character sets. The character set enabled for display, is selected by means of soft switches. Four soft switches designated SW0 through SW3 in Table 5-9 are used for selection of the five character sets. Refer to this table for the switch combinations necessary for the selection of the available character sets.

Character sets one through three can also be displayed as normal, and inverse or blinking characters. As shown in Table 5-9, the soft switch designated SW3 can be used to provide the inverse or blinking characteristics of these character sets. Character set four can display 128 normal, 64 inverse and 64 flashing characters.

Table 5-10 lists the actual address of the soft switches (SW0 through SW3) used to select the character sets and to change inverse and blinking characteristics.

Table 5-9. Character Set Switches

CHARACTER SET	CHARACTERS	SW0	SW1	SW2	SW3
Set 0	64	0	0	0	0 *
Set 1	128	0	0	1	X
Set 2	128	0	1	0	X
Set 3	128	0	1	1	X
Set 4	128+	1	0	0	X *

* This soft switch, activated with this character mode, reverses inverse and blinking characteristics.

Table 5-10. Character Set Soft Switches

HEX ADDRESS	FUNCTION
$C000	SW3 Inverse
$C001	SW3 Blinking
$C002	SW2 Off (0)
$C003	SW2 On (1)
$C004	SW1 Off (0)
$C005	SW1 On (1)
$C006	SW0 Off (0)
$C007	SW0 On (1)

Character Sets

The BASIS 108 has been delivered with two different character generators. The 2532 EPROM in computerboard position I14 is identified as 46-3 or 46-4 depending upon the position of it's character sets. Tables 5-11 lists the character sets and their positions.

Table 5-11. Character Sets

CHARACTER SET	46-3	46-4
Set 0	Standard Apple II	Standard Apple II
Set 1	German ASCII	ASCII
Set 2	ASCII	German ASCII
Set 3	APL	APL
Set 4	Full German ASCII	Full ASCII

Character generator 46-3 is installed in systems delivered with the 25-86 System Monitor. On power on, Character Set 2, ASCII, is the selected set.

Character generator 46-4 is installed in systems delivered with the 25-120 System Monitor. On power on, Character Set 1, ASCII, is the selected set.

If either system configuration is used with a version of the BASIS BOOTER DISKETTE that is different from the one delivered with the system, the German ASCII character set is selected on power on.

SECTION 6

INPUT/OUTPUT

SYSTEM I/O

The BASIS 108 has several built-in Input and Output (I/O) capabilities. Detail about these various BASIS 108 I/O features is covered in the remainder of this section.

- Keyboard interface
- Parallel printer interface
- Serial RS-232C interface
- Cassette recorder interface
- External I/O interface
- Three TTL inputs
- Four TTL outputs
- Loudspeaker output

KEYBOARD

The keyboard provides the user with a direct interface to the computer. This man-machine interface provides for human interaction with the computer. Operations requiring data entry or other program control functions are supported.

The keyboard supplied with the BASIS 108 provides many features normally only available with the addition of add-ons or "EXTRAS". The keyboard contains 100 keys divided into four functional areas. These areas are: A 58 key extended typewriter keyboard, 9 cursor control keys, an 18 key, full function, numeric keypad, and 15 function keys. The keyboard layout and, character or code generated for each key are shown in Figure 6-1.

The keyboard sends the computer information on which key was pressed, as well as detailed status information about the keyboard. The keyboard is connected to the computer by a 13 conductor cable terminated with a DB-25 connector on the computer end.

The keyboard contains a microprocessor which sends key position and status information to the computer. Circuitry in the computer converts the key position information to American Standard Code for Information Interchange (ASCII) codes and function codes. As each key is capable of generating four distinct outputs (output conditions consisting of lowercase, shift, control, or control and shift along with the pressed key). Thus, all codes from 0 to 255 can be generated. This allows generation of 128 ASCII codes and up to 128 additional codes. Normally the keyboard is mapped to generate 128 ASCII codes and 70 function codes (seven of the cursor control keys also generate function codes). All keys are autorepeat. When a key is depressed and held down for 0.6 seconds, the code for that key is continuously repeated at a rate of approximately 18 times per second, as long as the key is held down.

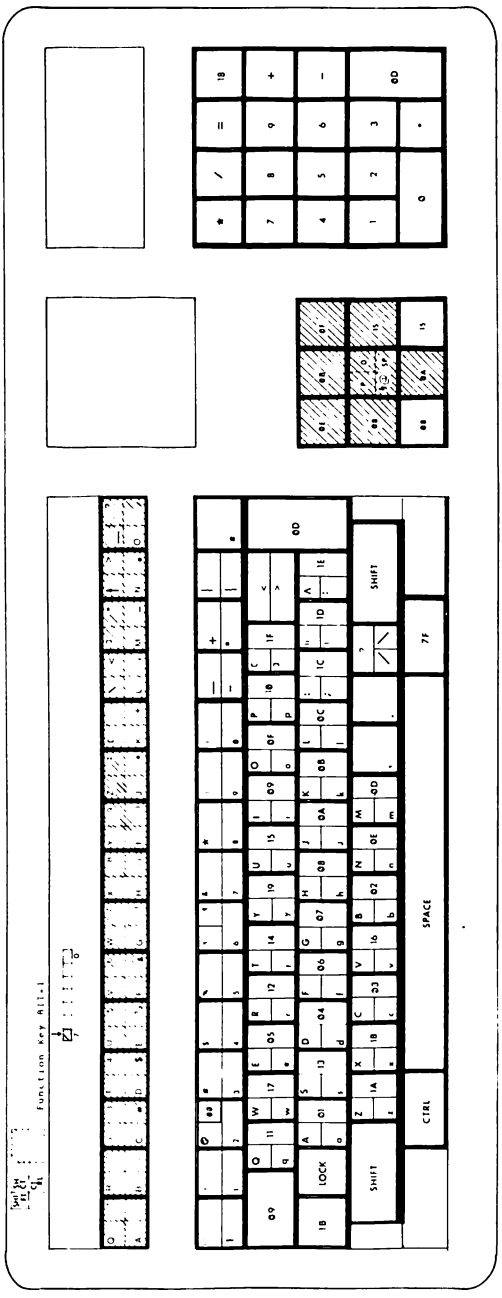


Figure 6-1. Keyboard

Table 6-1. Standard 7-bit ASCII Codes

BIT 3 2 1 0	6 5 4 COL / ROW	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
		0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Since ASCII is a 7-bit code, it has 128 characters (2^7). Some of the characters are printable, while others are control characters. The definitions for these codes are listed in Table 6-2. Many of the original meanings are lost to antiquity in the everchanging computer and communications fields.

Table 6-2. ASCII Code Definitions

MNEMONIC	MEANING	MNEMONIC	MEANING
SOH	Start of Heading	DC1	Device Control 1
STX	Start of Text	DC2	Device Control 2
ETX	End of Text	DC3	Device Control 3
EOT	End of Transmission	DC4	Device Control 4
ENQ	Enquiry	NAK	Negative Acknowledge
ACK	Acknowledge	SYN	Synchronous Idle
BEL	Bell	ETB	End of Transmission Block
BS	Back Space	CAN	Cancel
HT	Horizontal Tabulation	EM	End of Medium
LF	Linefeed	SUB	Substitute
VT	Vertical Tabulation	ESC	Escape
FF	Formfeed	FS	File Separator
CR	Carriage Return	GS	Group Separator
SO	Shift Out	RS	Record Separator
SI	Shift In	US	Unit Separator
DLE	Data Link Escape	DEL	Delete

Table 6-3. Keyboard Keystrokes

HEX	DEC	ASCII	KEYS	HEX	DEC	ASCII	KEYS
$00	0	NUL	Control Shift 2	$18	24	CAN	Control X
$01	1	SOH	Control A	$19	25	EM	Control Y
$02	2	STX	Control B	$1A	26	SUB	Control Z
$03	3	ETX	Control C	$1B	27	ESC	ESC
$04	4	EOT	Control D	$1C	28	FS	Control ;
$05	5	ENQ	Control E	$1D	29	GS	Control '
$06	6	ACK	Control F	$1E	30	RS	Control :
$07	7	BEL	Control G	$1F	31	US	Control]
$08	8	BS	←				
$09	9	HT	TAB	$20	32	SPACE	SPACE
$0A	10	LF	Control J	$21	33	!	Shift 1
$0B	11	VT	Control K	$22	34	"	Shift '
$0C	12	FF	Control L	$23	35	#	#
$0D	13	CR	Control M	$24	36	$	Shift 4
$0E	14	SO	Control N	$25	37	%	Shift 5
$0F	15	SI	Control O	$26	38	&	Shift 7
				$27	39	'	'
$10	16	DLE	Control P	$28	40	(	Shift 9
$11	17	DC1	Control Q	$29	41	)	Shift 0
$12	18	DC2	Control R	$2A	42	*	Shift 8
$13	19	DC3	Control S	$2B	43	+	Shift =
$14	20	DC4	Control T	$2C	44	,	,
$15	21	NAK	→	$2D	45	-	-
$16	22	SYN	Control V	$2E	46	.	.
$17	23	ETB	Control W	$2F	47	/	/

Table 6-3. Keyboard Keystrokes (Continued)

HEX	DEC	ASCII	KEYS
$30	48	0	0
$31	49	1	1
$32	50	2	2
$33	51	3	3
$34	52	4	4
$35	53	5	5
$36	54	6	6
$37	55	7	7
$38	56	8	8
$39	57	9	9
$3A	58	:	:
$3B	59	;	;
$3C	60	<	Shift <
$3D	61	=	=
$3E	62	>	>
$3F	63	?	Shift /
$40	64	@	Shift 2
$41	65	A	Shift A
$42	66	B	Shift B
$43	67	C	Shift C
$44	68	D	Shift D
$45	69	E	Shift E
$46	70	F	Shift F
$47	71	G	Shift G
$48	72	H	Shift H
$49	73	I	Shift I
$4A	74	J	Shift J
$4B	75	K	Shift K
$4C	76	L	Shift L
$4D	77	M	Shift M
$4E	78	N	Shift N
$4F	79	O	Shift O
$50	80	P	Shift P
$51	81	Q	Shift Q
$52	82	R	Shift R
$53	83	S	Shift S
$54	84	T	Shift T
$55	85	U	Shift U
$56	86	V	Shift V
$57	87	W	Shift W
$58	88	X	Shift X
$59	89	Y	Shift Y
$5A	90	Z	Shift Z
$5B	91	[	Shift]
$5C	92	\	Control /
$5D	93	]	]
$5E	94	^	Shift :
$5F	95	_	Shift -
$60	96	`	Shift Control 6
$61	97	a	A
$62	98	b	B
$63	99	c	C
$64	100	d	D
$65	101	e	E
$66	102	f	F
$67	103	g	G
$68	104	h	H
$69	105	i	I
$6A	106	j	J
$6B	107	k	K
$6C	108	l	L
$6D	109	m	M
$6E	110	n	N
$6F	110	o	O
$70	112	p	P
$71	113	q	Q
$72	114	r	R
$73	115	s	S
$74	116	t	T
$75	117	u	U
$76	118	v	V
$77	119	w	W
$78	120	x	X
$79	121	y	Y
$7A	122	z	Z
$7B	123	{	{
$7C	124	|	|
$7D	125	}	Shift {
$7E	126	~	Shift |
$7F	127	DEL	DELETE

Reset Function

There is no RESET key as such on the BASIS 108. The RESET function is started when both SHIFT keys and the CONTROL key are depressed together (simultaneously). When any one of these three keys is released the microprocessor begins a reset cycle.

Alphalock and Shiftlock

The keyboard LOCK key performs two functions. It places the keyboard in the Alphalock condition (alpha keys in uppercase) if it is pressed together with the CONTROL key. If it is pressed by itself it places the keyboard in the Shiftlock condition (all keys uppercase). The red LED in the LOCK key is illuminated when the key is active, in a lock condition. If the shift key is pressed while either lock condition exists, it causes the lowercase character to be generated for keys that would otherwise produce an uppercase character. Either lock condition is removed by pressing the LOCK key.

Interrupt Mode

The keyboard can be used in an interrupt mode. When this feature is enabled the depression of any key causes the generation of an interrupt (IRQ). The keyboard interrupt mode is enabled and disabled by writing to soft switches, one to turn the mode on and another to turn it off. The use of this feature requires a user software routine. Table 6-4 lists these locations.

Table 6-4. Keyboard Interrupt Mode Soft Switches

ADDRESS	FUNCTION
$C008	Keyboard Interrupt OFF
$C009	Keyboard Interrupt ON

Keyboard Mapping

The data sent from the keyboard to computer for depression of any key represents which key was pressed and keyboard status. This key position information, along with keyboard status, is used to convert the keyboard input to an ASCII code, 0 to 127 decimal, or a function code.

Using this type of keyboard encoding scheme allows the keyboard to be easily remapped by simply inserting a new keyboard conversion ROM. The ROM converts keyboard input to codes contained in the ROM. Table 6-5 lists information about this scheme.

Keyboard status consists of information related to the depression of keys. The Control key, Shift status, any function key, and any key depressed cause the generation of status information about the keyboard. This information can be obtained by reading the byte at any of the locations $C008 through $C00F and looking at specific bits of the input byte. Three other computer status

Table 6-5. Keyboard ROM

The first two digits of the ROM address are shown in the left column. The last digit of the address is shown across the top of the chart. The keys are listed in each row with the codes they produce beneath them.

The lines in each group of addresses relate to the following conditions:

xx CONTROL
xx SHIFT and CONTROL
xx LOWERCASE
xx SHIFT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	
07	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	00
17	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	00
0F	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	00
1F	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	00
	!	1	2	3	4	5	6	7	8	9	0	-	=	)	#	
06	7C	31	32	33	34	35	36	37	38	39	30	2D	3D	7B	23	00
16	7E	21	00	23	24	25	60	26	2A	28	29	1F	2B	7D	23	00
0E	7C	31	32	33	34	35	36	37	38	39	30	2D	3D	7B	23	00
1E	7E	21	40	23	24	25	27	26	2A	28	29	5F	2B	7D	23	00
		Q	W	E	R	T	Y	U	I	O	P	]	>	CR		
05	00	11	17	05	12	14	19	15	09	0F	10	1F	3E	0D	00	00
15	00	51	57	45	52	54	59	55	49	4E	50	5B	3C	0D	00	00
0D	00	71	77	65	72	74	79	75	69	6F	70	5D	3E	0D	00	00
1D	00	51	57	45	52	54	59	55	49	4F	50	5B	3C	0D	00	00
	ESC	LOCK	A	S	D	F	G	H	J	K	L	;	'	:		
04	1B	FF	01	13	04	06	07	08	0A	0B	0C	1C	1D	1E	00	00
14	1B	FF	01	13	04	06	07	08	0A	0B	0C	1C	1D	1E	00	00
0C	1B	FF	61	73	64	66	67	68	6A	6B	6C	3B	27	3A	00	00
1C	1B	FF	41	53	44	46	47	48	4A	4B	4C	3A	22	5E	00	00
	Z	X	C	V	B	N	M	,	.	/	TAB	SPC	DEL			
03	1A	18	03	16	02	0E	0D	2C	2E	5C	09	20	7F	00	00	00
13	1A	18	03	16	02	0E	0D	2C	2E	3F	09	20	7F	00	00	00
0B	7A	78	63	76	62	6E	6D	2C	2E	2F	09	20	7F	00	00	00
1B	5A	58	43	56	42	4E	4D	2C	2E	3F	09	20	7F	00	00	00
	| cursor row 1 | keypad row 3 | keypad row 2 |															
02	8E	8B	8F	34	35	36	2D	37	38	39	2B	00	00	00	00	00
12	8E	8B	8F	34	35	36	2D	37	38	39	2B	00	00	00	00	00
0A	8E	8B	8F	34	35	36	2D	37	38	39	2B	00	00	00	00	00
1A	8E	8B	8F	34	35	36	2D	37	38	39	2B	00	00	00	00	00
	| cursor row 2 | keypad row 4 | keypad row 1 |															
01	88	A0	95	31	32	33	0D	2A	2F	3D	18	00	00	00	00	00
11	88	B0	95	31	32	33	0D	2A	2F	3D	18	00	00	00	00	00
09	88	C0	95	31	32	33	0D	2A	2F	3D	18	00	00	00	00	00
19	88	D0	95	31	32	33	0D	2A	2F	3D	18	00	00	00	00	00
	| cursor row 3 | keypad row 5 |															
00	08	8A	15	00	30	2E	00	00	00	00	00	00	00	00	00	00
10	08	8A	15	00	30	2E	00	00	00	00	00	00	00	00	00	00
08	08	8A	15	00	30	2E	00	00	00	00	00	00	00	00	00	00
18	08	8A	15	00	30	2E	00	00	00	00	00	00	00	00	00	00

Keyboard Mapping (Continued)

conditions are also available at these addresses. They are the Horizontal Blanking Signal, Video Synchronization Signal, and Printer Active Status. The bit positions of these status signals are listed in Table 6-6.

Table 6-6. Status Byte

BIT	STATUS FUNCTION
0	Printer Active
1	Video Synchronization Signal (VSS)
2	Horizontal Blanking Signal (HBL)
3	NOT USED
4	NOT USED
5	Control key depressed
6	Shift status
7	Function key depressed

Keyboard Interface

The keyboard is connected to the chassis by means of a DB-25 connector. The top DB-25 connector on the rear panel is used for this purpose. Figure 6-2 shows the pinout of this connector.

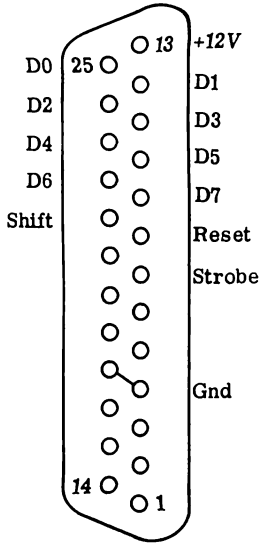


Figure 6-2. Keyboard Interface Connector

PARALLEL PRINTER INTERFACE

The Printer Interface is a Centronics compatible interface which generates all signals necessary for controlling the printer. Output data are written to the output addresses $C090 through $C097. Upon receipt of data the STROBE signal is automatically generated. The status of the printer's ACKNOWLEDGE can be checked in the highest order bit of address $C1C1. The driver routine for the interface is located in ROM and begins at address $C100. The PR#1 command activates this interface. The ROM listing is located in Appendix D. Figure 6-3 shows the pinout for the interface connector on the rear panel.

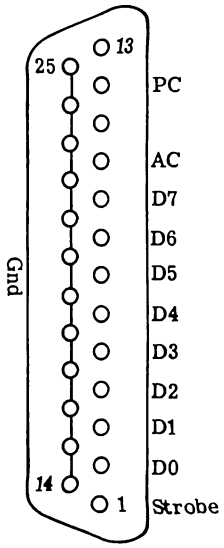


Figure 6-3. Parallel Printer Interface Connector

RS-232C SERIAL INTERFACE

The serial interface consists of a 6551 Asynchronous Communications Interface Adapter (ACIA) and the circuitry necessary, including line drivers and receivers, to allow full operation of the ACIA. The ACIA provides a program controlled interface between the system and serial communications equipment.

A driver routine is contained in ROM starting at address $C100. When called, the driver routine initializes the serial port to 9600 baud, 8 bits with 2 stop bits, and no parity. These values can be changed by program at any time. The PR#9 command is used from DOS to activate this interface.

6551 Operation

The BASIS 108 is capable of transmitting at 15 program selectable rates between 50 baud and 19,200 baud. Programmable word lengths of 5, 6, 7 or 8 bits; even, odd or no parity; 1, 1 1/2 or 2 stop bits are possible. A separate command register and a separate control register allow easy selection of operating modes and an easy means to check data, parameters and status. External clock operation is not supported. The four addresses listed in Table 6-7 are used to provide the computer's internal interface with the ACIA.

Table 6-7. 6551 Addresses

ADDRESS	WRITE	READ
$C098	Transmit Data Register	Receive Data Register
$C099	Program Reset *	Status Register
$C09A	Command Register	Command Register
$C09B	Control Register	Control Register

Note that only the Command and Control registers are read/write. The Programmed Reset operation does not cause any data transfer, but is used to clear bits 0 through 4 in the Command Register and Bit 2 in the Status Register. The Programmed Reset is slightly different from the Hardware Reset; these differences are shown with each of the registers. External clock operation is not supported. Refer to Appendix F for detailed information on the 6551 ACIA.

6551 Registers

The Control register selects the desired baud rate, frequency source, word length, and the number of stop bits. The Command Register controls specific modes and functions. The Status Register reports the status of various 6551 functions. The Receive and Transmit data registers are used as temporary data storage for the Transmit and Receive circuits. The Transmit Data Register is characterized as follows:

- Bit 0 is the leading bit to be transmitted.
- Unused data bits are the high-order bits and it does not matter what they are during transmission.

The Receiver Data Register is characterized in a similar fashion:

- Bit 0 is the leading bit received.
- Unused data bits are the high-order bits and are "0" for the receiver.
- Parity bits are not contained in the Receive Data Register, but are stripped-off after being used for external parity checking. Parity and all unused high-order bits are "0".

Parameter Addresses

Table 6-8 lists the Serial RS232 parameter addresses for various operating systems. These addresses are supported by the Printer ROM.

Table 6-8. RS232 Parameter Addresses

FUNCTION	DOS BASIC	UCSD II.1.1 (6502 addressing)	UCSD IV.0	CP/M (Z-80 addressing)
6551 Control Register (baudrate, word length, stopbits)	$06F9	$FFCE	$0271	$F280
6551 Command Register (Parity, RTS, DTR)	$0779	$FFCF	$0270	$F282
Input:	IN#9 *	REMIN: #7:	REMIN: #7:	UR1:
Output:	PR#9	REMOUT: #8:	REMOUT: #8:	UL1: UP1:

* POKE 41153,10 enables the use of the PR#9 and IN#9 commands with DOS.

Serial Interface Connector

The Serial interface DB-25 connector on the rear panel uses standard RS-232C pin locations. These are identified in Figure 6-4.

CASSETTE INTERFACE

A cassette recorder can be connected to the BASIS 108 by means of the DIN connector on the rear panel. This connector is shown in Figure 6-5. In this way a standard cassette tape recorder can be used to save information to the tape and read it back again.

Address $C020 is used to write to the recorder and address $C060 is used to read information from the recorder. Referencing the soft switch for the output causes the output to switch or toggle from zero volts to 25 millivolts, or return from 25 millivolts to zero volts. When applied to a recorder this transition can be used to record information. When driven under program control this output provides a convenient way to store data. A write operation to the soft switch causes two outputs, a read operation one output.

The input takes a 1 volt (peak-to-peak) signal from the input and converts it to digital information. Each time the input signal changes from positive to negative or from negative to positive the input at $C060 changes state. By reading this address the changing state of the input can be obtained. A

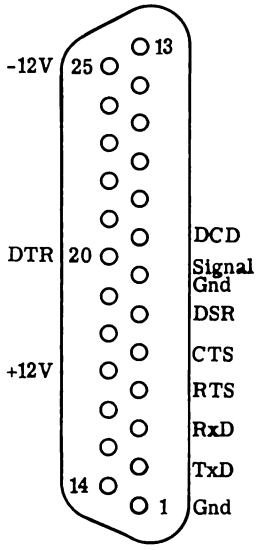


Figure 6-4. Serial Interface Connector

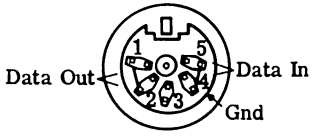


Figure 6-5. Cassette Interface Connector

value less than 128 is said to be a zero. A value equal to or greater than 128 is said to be a one.

Programs using this interface should be written in machine language. Basic or other programs are usually to slow to see all of the changes and properly decode them. Driver routines are located in the 40 column monitor for use of this interface.

EXTERNAL I/O

The External I/O connector on the computerboard provides a convenient connector for control paddles, joysticks or other external devices. This connector provides interface to four analog inputs, three TTL inputs, four TTL outputs, and a strobe output. Figure 6-6 shows the pinout of this connector and Table 6-9 describes these signals.

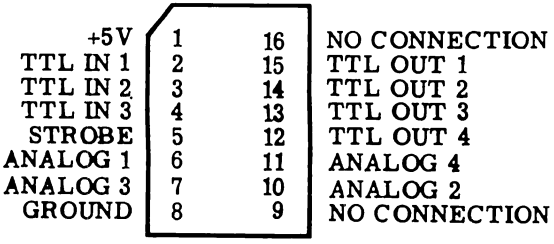


Figure 6-6. External I/O Connector

Table 6-9. External I/O Signal Descriptions

NAME	DESCRIPTION
+5V	+5 volt power supply
TTL IN	Standard 74LS series TTL inputs
STROBE	Normally high. Goes low during a read or write to $C040 through $C04F. This is a standard 74LS TTL output.
ANALOG	These inputs should be connected through a 150 kOhm variable resistor to +5V.
GROUND	System ground
TTL OUT	Standard 74LS series TTL outputs

TTL Inputs

The three TTL inputs can each be connected to an external device or to a pushbutton switch. The three one-bit inputs can be read at addresses $C061 through $C063 for inputs 1 through 3 respectively.

TTL Outputs

Each TTL Output can be used to interface to external device or circuit. Each output is controlled by a soft switch. The "off" state sets an output to approximately 0 volts. The "on" state sets an output to approximately 5 volts. Table 6-10 lists the TTL Output soft switch addresses.

Table 6-10. TTL Output Soft Switches

TTL OUT	STATE	ADDRESS
1	OFF	$C058
	ON	$C059
2	OFF	$C05A
	ON	$C05B
3	OFF	$C05C
	ON	$C05D
4	OFF	$C05E
	ON	$C05F

Analog Inputs

The Analog inputs should be connected to 150 kOhm variable resistors or potentiometers. The resistance between each input and the +5 volts is used to provide a reference voltage to a timing circuit. As the input varies, the characteristics of the timing circuit are varied. This input is measured and provided as a numerical value which can be read.

Before an input is read the timing circuits must be reset. This is done by referencing address $C070. This causes the reference address associated with each input to become greater than 128. After approximately 3 milliseconds the reference address should contain a reading proportional to the respective input. The reference addresses for the Analog inputs are $C064 through $C067 for Analog Inputs 1 through 4 respectively.

Strobe

The Strobe output is normally +5 volts. It will drop to 0 volts for a duration of approximately one-half microsecond when address $C04X is referenced. If a write is performed to this address the output will be toggled twice.

SPEAKER OUTPUT

The speaker is mounted between the disk drive mounting brackets. It is connected to circuitry which allows it to produce a large range of sounds. The speaker output connector is located on the rear right side of the computerboard. The same connector is shared with cassette interface.

A soft switch is used to control the output to the speaker. Each time the soft switch at address $C030 is referenced, the speaker produces a click. If a write is performed to this address the output to the speaker is toggled twice. The pitch and duration of the speaker output can be varied under software control.

SECTION 7

SYSTEM MONITOR

FOREWARD

The System Monitor is a program located on the computerboard in the IC at position I1. This location is also known as IC socket 25. The different variants of the monitor are described by a 25 followed by an identifying number. Currently there are three variants of the monitor. They are: 25-46, 40 character screen driver and cassette recorder interface; 25-86, 80 column screen driver, and 25-120, which contains a switchable screen driver for 40 or 80 column operation. As currently delivered the BASIS 108 contains the 25-120 monitor and the BASIS BOOTER diskette contains the 25-46 and 25-120 versions of the monitor. Earlier systems were delivered with the 25-86 version monitor on the computerboard and 25-46 and 25-86 version monitors on the diskette. Listings for all three versions of the monitor are contained in Appendix D.

The monitor both controls the system and allows control of many of the system features by the use of subroutines (programs) it contains. The monitor is written in 6502 machine language. The capabilities covered in this appendix refer to the 6502 and its registers, and not the Z-80. One of its many features is the "autoboot" or locate a disk controller and load from diskette upon power on. The monitor provides many user routines. This appendix explains these user features.

ENTERING THE MONITOR

The Monitor is entered at monitor location $FF69. This is done from Basic by executing a CALL to 65385 or -151. The monitor prompt * (an asterisk) appears at the left edge of the screen. The monitor accepts input command lines up to 255 characters long. It takes no action on the input until you press RETURN. When you are finished using the monitor and wish to return to the language previously being used a Q <RETURN> or System RESET is used. When DOS 3.3 is being used the 3DOG <RETURN> command can be used.

Data and Addresses

Working in the monitor is similar to working with any other program: input is entered through the keyboard, followed by a RETURN. The monitor then checks the input and acts upon valid commands. You supply the monitor with three different types information: commands, addresses, and data. Addresses and data are are given to the monitor in hexadecimal notation consisting of the ten digits 0-9 and the letters A-F. When data is entered the monitor is looking for a value of $0 to $FF. When an address is entered the monitor is looking for a value of $0 to $FFFF. If there are fewer than four digits in an address, leading zeros are added. If there are more than four digits, the input is truncated and the last four digits are used. The same procedure is used with two digit inputs.

MONITOR COMMANDS

Each of the monitor commands is described in detail. Examples for each command and variations of a command are shown. Many of the examples use other commands to show the memory contents before and after the command which is being demonstrated. The **bold face** portion of an example is the portion entered from the keyboard.

Examining Memory

Enter the address of the location you wish to examine. The monitor replies with the address you entered and the contents (value) of that address. Depending on the use of a 40 or 80 column screen display, the actual display will show eight or sixteen locations across the screen. The examples are shown using an 80 column display.

```
*20 <RETURN>
 0020: 00
*
```

Examining several memory address is done by using the . (period). This causes the memory from the first address entered to the address following the period to be displayed.

```
*0 <RETURN>
 0000: 04
*.11 <RETURN>
 0001: C6 00 0A 1B 18 18 00 00 FF 4C FF FF 22 00 6B
 0010: 00 00
*
```

Another way to perform this operation is:

```
*30.40 <RETURN>
 0030: FF 00 FF AA 05 00 BD 9E 81 9E FF FF 36 00 41 00
 0040: 30 00
*
```

Just pressing RETURN also causes the contents of memory to be listed.

```
*5 <RETURN>
 0005: 18
*<RETURN>
 18 00 00 FF 4C FF FF 22 00 6B
*<RETURN>
 0010: 00 00 00 00 04 00 FF 00 FF FF FF FF FF FF FF FF
*
```

Changing Memory

Changing the contents of memory is accomplished by typing a value after a : (colon). The last address examined is the address changed.

```
*0 <RETURN>
 0000: 04
*:3C <RETURN>
*0 <RETURN>
 0000: 3C
*
```

An address may be changed directly by using the following format.

```
*10:33 <RETURN>
*10 <RETURN>
 0010: 33
*
```

To change the contents of consecutive address locations, several values are entered following the :. These values are each separated by a space as they are entered. The following example shows the addresses 0 through 7 being changed.

```
*0.7 <RETURN>
 0000: 5F C6 00 0A 1B 18 18 00
*0: 6F 3A 1 B 1A 16 11 07 <RETURN>
*0.7 <RETURN>
 0000: 6F 3A 01 0B 1A 16 11 07
*
```

Moving Memory

The MOVE command copies a memory range to another place in memory. This operation copies the memory range specified to a new destination. The original memory area specified is not altered. The form of the command is:

(destination) < (start) . (end)M

The destination, start, and end are the addresses specifying where the copy is to begin, what address to start with, and the last address to copy. The letter **M** specifies the operation to perform. The example shows the memory range from addresses 0 - 7 being copied to addresses 100 - 107.

```
*100.107 <RETURN>
 0100: FF FF FF FF FF FF FF FF
*100<0.7M <RETURN>
*100.107 <RETURN>
 0100: 6F 3A 01 0B 1A 16 11 07
*
```

Comparing Memory

The VERIFY command is used to compare two ranges of memory. The format of the command is similar to the MOVE command.

(destination) < (start) . (end)V

The monitor compares the range specified with the range beginning at the destination address. If there are any differences, the monitor displays the address at which the difference was found and the values compared.

```
*100<0.7 <RETURN>
*100<0.8 <RETURN>
 0008: 00 FF
*
```

The first line of the example compares memory range 100 - 107 to the memory range 0 - 7, and found no differences. The second line of the example compares the memory range 100 - 108 to the memory range 0 - 8, and found a difference between address 8 and 108.

Program Execution

The GO command is used to begin execution of a machine language program. The entry is the address followed by the letter G. The following example enters a program and then uses the GO command to execute the program. Program execution begins at the address specified. The program displays the values 0 through 9 on the screen when it is executed.

```
*0:A9 B0 20 ED FD 18 69 1 C9 BA D0 F6 60 <RETURN>
*0.D <RETURN>
 0000: A9 B0 20 ED FD 18 69 01 C9 BA D0 F6 60 00
*0G <RETURN>
 0123456789
*
```

Program Listing

The LIST command shows a disassembled range of memory. The listing is in assembly language form and shows the address, its contents, and a three-letter mnemonic for each instruction. The disassembler takes one, two, or three bytes and displays them in an instruction format that is much easier to understand than the hexadecimal format seen in a memory listing. The command is entered in the form:

(start) . (end)L

The display is continuous, starting at the first address entered and ending at the second address given. The 6502 mnemonics are shown in Appendix E. The example used is the program entered in **Program Execution.**

Program Listing (Continued)

```
*0.DL <RETURN>
 0000:   A9 B0       LDA   #$B0
 0002:   20 ED FD    JSR   $FDED
 0005:   18          CLC
 0006:   69 01       ADC   #$01
 0008:   C9 BA       CMP   #$BA
 000A:   D0 F6       BNE   $0002
 000C:   60          RTS
 000D:   00          BRK
*
```

Examining and Changing Registers

The contents of the 6502's registers can be examined and changed. The monitor EXAMINE command ?, instructs the monitor to display the contents of the registers. The five registers: A, X, Y, P (processor status register), and S (stack pointer) are displayed.

To change the values in the registers, enter a colon and the values separated by spaces. The values are actually moved to the 6502 registers the next time a GO command is executed.

```
*? <RETURN>
 A=88 X=13 Y=D8 P=00 S=B7
*:A B
*? <RETURN>
 A=0A X=0B Y=D8 P=B0 S=F8
*
```

Input Vector

The command xK, where x is the slot from 2 through 7 or 9, accepts the input from the device in that slot. CTRL-OK accepts input from the keyboard. Slot 9 is the serial interface input.

Output Vector

The command xP, where x is the slot from 1 through 7 or 9, sends output to the device in that slot. Slot 9 is the serial output interface.

Inverse Mode

The command I causes video to be displayed in inverse.

Normal Mode

The command N causes video to be displayed in the normal (not inverse) mode.

Execute $3F8

The USER command U causes a jump to the machine language routine at location $3F8. The following example loads $3F8 with a jump instruction to the monitor routine that causes the speaker to "BEEP".

```
*3F8: 4C 3A FF <RETURN>
*3F8L <RETURN>
 03F8: 4C 3A FF     JMP $FF3A
*U <RETURN>          ***** beep *****
*
```

Cassette Commands

These commands are only contained in the 25-46 monitor. They are used read and write a range of memory to tape. The first command, WRITE, lets you save the contents of one to 65,536 memory address to tape. To save the memory range, the starting and ending addresses followed by W is entered in the following format:

(start) . (end)W

The tape recorder should be placed in the record mode prior to pressing RETURN on the input line. Turn the recorder on, let it run for a few seconds and then press RETURN. The monitor then writes a ten second header on the tape, followed by the data and a checksum value. When the monitor has completed the operation you will hear a "BEEP" and see the prompt return to the screen. At this time you should stop the recorder and write a description of the data saved. Additionally, if your recorder has a counter you should also write the counter start and stop numbers to further identify where the data is located on the tape. The following example writes a memory range to tape.

```
*0.14W <RETURN>
*
```

A memory range can be loaded from tape using the READ command. Memory is loaded to the addresses specified. The tape data can be loaded to any similarly sized memory range. The form of the command is similar to WRITE, with an R instead of the W. The format is:

(start) . (end)R

Locate the beginning of the tape segment to be loaded. Place the recorder in play and wait a few seconds before pressing RETURN. This allows time for the recorder's output to settle down and be acceptable at the computer's input. Only three seconds of the header tone is necessary for the computer to lock on to the input frequency. After the monitor has read and stored the data values on the tape, it reads the checksum value. It then checks the computed check-

sum for the values read and the checksum read from the tape. If the two checksum values are different, the "ERR" message is displayed and a "BEEP" is heard. This tells you that the values loaded to memory are not the same as the values on the tape. If all was proper the monitor prompt is displayed. The example loads the memory range saved above with the WRITE command.

```
*0.14R <RETURN>
*
```

MORE ABOUT THE MONITOR

You can place as many monitor commands as you wish on a single line, as long as the commands are separated with a space and the line length is not longer that 254 characters. You can mix all commands freely accept the STORE (:) command. The monitor takes all values after a STORE command and places them in consecutive memory locations. The last value in a STORE command must be a letter command. The NORMAL command (N) is ideal for this since it no effect other than to return the display to normal video.

If the monitor finds a character which it does not recognize as a hexadecimal digit or a valid command, it executes all commands in the input line to this point. It then causes a "BEEP" for the invalid input, returns a prompt to the screen, and ignores the remainder of the input line.

The MOVE (M) command can be used to duplicate a pattern of values throughout a memory range. First store the pattern in its first position in the memory range. Remember the number of values in the pattern. Then use the following arrangement of the MOVE command:

```
(start + n) < (start) . (end - n)M        where n is the number of values in
                                          the pattern
```

The following example shows the use of this variation of the command.

```
*0:1 2 3 <RETURN>
*3<0.CM <RETURN>
*0.10 <RETURN>
 0000: 01 02 03 01 02 03 01 02 03 01 02 03 01 02 03 01
 0010: 00
*
```

A command line which will repeat all or part of itself indefinitely can be entered. Beginning the command line with a letter command, such as N, and ending it with the sequence 34:n, where n is a hexadecimal number specifying the character position of the command which begins the loop; for the first character of the line, n=0, causes a never ending loop. The value for N must be followed by a space. The only way to stop the loop is RESET. Try the following example.

```
*N 0 2 34:N <RETURN>
 0000: 01
 0002: 03
 0000: 01
 0002: 03
 0000: 01
 0002: 03
 .
 .
 .
 <RESET>          You must enter RESET to stop the loop.
```

MONITOR COMMAND SUMMARY

A summary of the commands listed in this section follows.

Examining Memory

(address)	Display the contents of one memory address.
(start).(end)	Display the contents of the memory address between the start address and the end address.
RETURN	Display up to 16 locations following the last used address.

Changing Memory

(address):(contents)	Store the new value(s) in consecutive addresses starting at (address).
:(contents) (contents)	Store the new values in memory starting at the last address referenced.

Moving and Comparing Memory

(dest.)<(start).(end)M	Copy the values located in the range (start).(end) to the range beginning at (destination).
(dest.)<(start).(end)V	Compare the values located in the range (start).(end) to the range beginning at (destination).

MONITOR COMMAND SUMMARY (Continued)

Cassette Commands

(start).(end)W	Write the values in the memory range specified by (start).(end) to tape. Start with a ten second leader and follow the data with a checksum.
(start).(end)R	Read the values from tape. Store them in memory beginning at (start) and stop at (end). Print "ERR" if the checksum is not correct.

Running and Listing Programs

(address)G	Execute the program beginning at (address).
(start).(end)L	Disassemble and display the instructions beginning at (start) and ending at (end).

Miscellaneous Commands

?	Display the contents of the 6502's registers.
I	Set normal video display.
N	Set inverse video display.
CTRL-B	Enter the language installed in ROM.
Q	Quit the monitor. Jump to the address in locations $3F2, $3F3.
nP	Accept input from the device in slot n. If slot equals 0, accept input from the keyboard.
nK	Send output to the device in slot n. If slot equals 0, send output to the video screen.

MONITOR COMPARISON

The 80 column system monitor has many features that are different from the Apple monitor. Many of the routines in the BASIS 40 column monitor are the same as the 80 column monitor. Table 7-1 is a comparison of the monitors.

Table 7-1. Monitor Comparison

	APPLE	BASIS 80	BASIS 40
Screen:	24 X 40	24 X 80	24 X 40
	ESC-@	HOME	HOME
	ESC-E	Upper left corner of cursor control	Upper left corner of cursor control
	ESC-F	Upperright corner of cursor control	Upper right corner of cursor control
	ESC-I	Up arrow	Up arrow
	ESC-J	Left Arrow	Left arrow
	ESC-K	Right arrow	Right arrow
	ESC-M	Down arrow	Down arrow
	Left arrow	Large left arrow	Large left arrow
	Right arrow	Large right arrow	Large right arrow
Cassette:			
	nnnn.mmmmR	---	nnnn.mmmmR
	nnnn.mmmmW	---	nnnn.mmmmW
BASIC cold-start (without disk):			
	CTRL-B	CTRL-B	CTRL-B
BASIC warm-start (3DOG):			
	CTRL-C	Q	Q
Disassembler:			
	nnnnL	nnnn.mmmmL	nnnn.mmmmL

Table 7-1. Monitor Comparison (Continued)

	APPLE	BASIS 80	BASIS 40
Input Vector:			
	nCTRL-K	nK	nK
Output Vector:			
	nCTRL-P	nP	nK
Show 6502 registers:			
	CTRL-E	?	?
User program:			
	CTRL-Y	U	U
Input:	Uppercase letters only	Uppercase and lowercase letters	Uppercase and lowercase letters
Execute 6502 program:			
	nnnnG	nnnnG	nnnnG
Move:	nnnn.aaaa<bbbbM	nnnn.aaaa<bbbbM	nnnn.aaaa<bbbbM
Verify:	nnnn.aaaa<bbbbV	nnnn.aaaa<bbbbV	nnnn.aaaa<bbbbV
Display:	nnnn.mmmm	nnnn.mmmm with 16 bytes per line displayed	nnnn.mmmm with 8 bytes per line displayed

MONITOR SUBROUTINES

A list of some of the more useful subroutines in the monitor is provided. The subroutines are shown in their assembly language format in Appendix D.

$F800 PLOT - Place a block on the Lo-Res or Me-Res screen.

This subroutine plots a block on the screen in the prespecified color. The vertical position is sent to the accumulator, and its horizontal position to the Y register. Upon return the accumulator contents are changed.

$F819 HLINE - Draw a horizontal line of blocks

This subroutine draws a horizontal line of blocks of the predetermined color on the Lo-Res or Me-Res screen. On entry HLINE requires the vertical coordinate of the line in the accumulator, the leftmost horizontal coordinate in the Y register, and the rightmost horizontal coordinate in the location $2C. HLINE returns with A and Y values changed, X intact.

$F828 VLINE - Draw a vertical line of blocks.

This subroutine draws a vertical line of blocks of the predetermined color on the Lo-Res or Me-Res screen. On entry VLINE requires the horizontal coordinate of the line in the Y register, the top vertical coordinate in the accumulator, and the bottom vertical coordinate in location $2D. VLINE returns with the accumulator value changed.

$F832 CLRSCR (CLeaR SCReen) - Clear the Lo-Res or Me-Res screen.

This subroutine clears the screen. If CLRSCR is used while in TEXT mode, it fills the screen with inverse "@" characters. CLRSCR changes the contents of the A and Y registers.

$F836 CLRTOP (CLear TOP of screen) - Clear the top 40 lines of the Lo-Res or ME-Res screen.

CLRTOP performs the same function as CLRSCR, except that it clears only the top 40 rows of the screen. If CLRTOP is used while in TEXT mode, it fills the top 20 lines of the screen with inverse "@" characters.

$F85F NEXTCOL (NEXT COLor) - Increment color by 3.

This subroutine adds 3 to the current color used for Lo-Res or Me-Res Graphics.

$F864 SETCOL (SET COLor) - Set Lo-Res or Me-Res color.

This subroutine sets the color used for plotting on the Lo-Res or Me-Res screen to the color passed in the accumulator.

$F871 SCRN (read SCReeN) - Read the Lo-Res or Me-Res screen.

This subroutine returns the color of a single block on the Lo-Res or Me-Res screen. The block's vertical position is passed in the accumulator, its horizontal position in the Y register. The color of the block is returned in the accumulator. No other registers are changed.

$F941 PRNTAX (PRiNT A and X in hexadecimal) - Print A and X.

PRNTAX outputs the contents of the A and X registers as a four-digit hexadecimal value. The accumulator contains the first byte output, the X register contains the second byte output. The contents of the accumulator is changed.

$F948 PRBLNK (PRint 3 BLaNKs) - Print 3 spaces.

This subroutine outputs three blank spaces to the output device. On exit, the accumulator contains $A0 and the X register contains 0.

$F94A PRBL2 (PRint BLanks 2) - Print a lot of spaces.

This subroutine outputs from 1 to 256 blanks to the output device. Upon entry the X register should contain the number of blanks to be output. If X=$00, 256 blanks are output.

$FB1E PREAD - Read external analog device.

This subroutine returns a number which represents the analog input. The number of the analog input to be read is passed in the X register. Analog 1=0, 2=1, 3=2, and 4=3. PREAD returns with a number between $00 and $FF in the Y register. The accumulator is changed.

$FBDD BELL1 - Make a two-tone beep on the speaker.

BELL1 outputs a two-tone signal on the BASIS 108 speaker. The contents of the A and Y registers are changed.

$FD0C RDKEY (ReaD input KEY) - Get an input character.

RDKEY is the standard character input routine. It places a blinking cursor on the screen at the position of the output cursor and jumps to the input routine whose address is stored in KSW ($38 and $39) which is usually KEYIN.

$FD1B KEYIN (get KEYboard INput) - Read the keyboard.

This is the keyboard input subroutine. It reads the keyboard, waits for a key being depressed, and randomizes the random number seed. The keycode is returned in the accumulator. If a function key was pressed $C000 bit 7=1, otherwise bit 7=0.

$FD35 RDCHAR (ReaD input CHARacter) - Get an input character.

This is an alternate input subroutine which gets characters from the input device. It also interprets the cursor block keys.

$FD6A GETLN (GET input LiNe) - Get an input line with prompt.

This subroutine gathers input lines. Your programs can call GETLN with the proper prompt character in location $33. The subroutine returns with the input line in the input buffer (beginning at location $200) and the X register holding the length of the input line.

$FD67 GETLNZ - Get an input line.

GETLNZ is an alternate entry point for GETLN. It sends a carriage return to the output before executing GETLN.

$FD8E CROUT (Carriage Routine OUT) - Output a carriage return.

This subroutine sends a RETURN character to the output device.

$FDDA PRBYTE (PRint a hexadecimal BYTE) - Print a byte.

PRBYTE outputs the contents of the accumulator in hexadecimal on the current output device. The contents of the accumulator are changed.

$FDE3 PRHEX (PRint a HEXadecimal digit) - Print a digit.

PRHEX outputs the contents of the A and X registers as a four-digit hexadecimal value. The accumulator contains the first byte output, the X register contains the second. The contents of the accumulator are usually changed.

$FDED COUT (Character OUTput) - Output a character.

COUT is the standard character output routine. The character to be output is placed in the accumulator (A register). COUT calls the character output routine whose address is stored in CSW (locations $36 and $37) which is normally COUT1.

$FDF0 COUT1 (Character OUTput 1) - Output to screen.

COUT1 displays the character in the accumulator on the video screen at the current output cursor position and advances the output cursor. It places the character using the setting of the Normal/Inverse location. It handles the control character RETURN, LINEFEED, and BELL. The subroutine returns with all registers intact.

$FE80 SETINV (SET INVerse) - Set inverse mode.

SETINV sets the Inverse video mode for COUT1. All output characters are displayed as black dots on a white background. The Y register is set to $3F. All other registers are unchanged.

$FE84 SETNORM (SET NORMal) - Set normal mode.

SETNORM sets the Normal video mode for COUT1. All output characters are displayed as white dots on a black background. The Y register is set to $FF. All other registers are unchanged.

$FCA8 WAIT - Delay

This subroutine delays for a specific amount of time, then returns to the program that called it. The duration of the delay is specified by the value in the accumulator. The delay is approximately 13 + 12A + 5A*A useconds, where A is the value in the accumulator. Wait returns with the accumulator equal to 0 and the X and Y register unchanged.

$FF3A BELL - Output a 'bell' character.

BELL is the routine that that sends a bell character (Control G) to the current output device. It leave the accumulator holding $87.

$FF3F RESTORE - Restore registers.

This subroutine loads the 6502's registers from locations $45 through $48. The S (stack) register is not restored.

$FF4A SAVE - Save registers.

This subroutine saves the 6502's registers to locations $45 through $49 in the order A, X, Y, P, S. The contents of A and X are changed. The microprocessor decimal mode is cleared.

SPECIAL MONITOR ADDRESSES

Table 7-2 contains a list of locations used with the monitor.

Table 7-2. Special Monitor Addresses

DECIMAL ADDRESS	HEX ADDRESS	USE
32	$20	Contains the left edge of the screen.
33	$21	Contains the width o the screen.
34	$22	Contains the top edge of the screen.
35	$23	Contains the bottom edge of the screen.
50	$32	Contains the Normal/Inverse value used by COUT. $FF - display normal $3F - display inverse $7F - display flashing
1008	$3F0	The monitor uses this location as the BRK instruction
1009	$3F1	interrupt address. Normally contains $FA59.
1010	$3F2	Warmstart entry Address.
1011	$3F3	Monitor 'Q' jump address.
1012	$3F4	Power-up byte.
1013	$3F5	Holds the JMP instruction for the FPBASIC '&' command.
1014	$3F6	Normally contains $4C $58 $FF.
1015	$3F7	
1016	$3F8	Holds the JMP instruction for the USER 'U' command.
1017	$3F9	
1018	$3FA	
1019	$3FB	Holds the Non-Maskable (NMI) JMP instruction.
1020	$3FC	
1021	$3FD	
1022	$3FE	IRQ Interrupt Address.
1023	$3FF	
1273	$4F9	When = 0 - 40 column display enabled. When ≠ 0 - 80 column display enabled. *** only used with monitor 120 ***

APPENDIX A

BASIS BOOTER DISKETTE

GENERAL INFORMATION

In order to perform the procedures in this Appendix, you are required to have or purchase the following items:

- Apple Pascal 1.1
- Apple DOS 3.3 Basics Diskette
- Apple DOS 3.3 System Master Diskette
- Blank Diskettes

Two BASIS diskettes are supplied with the BASIS 108 system. They are the BASIS BOOTER and the BASIS UTILITY diskettes. The BASIS BOOTER diskette contains a program which adapts Apple Pascal 1.1, Applesoft, Integer Basic, and CP/M to use the built-in features of the BASIS 108, as well as a fast boot routine which allows rapid language loading for the system. Information about the BASIS UTILITY diskette is in Appendix B.

There are two versions of the diskette. The difference is in the files contained on the diskette. If the diskette has the file 25-86 it is called OLD. If it contains the file 25-120 it is called NEW. The NEW diskette can be identified by the gold label. As provided the BASIS BOOTER contains only one bootable file, A: VC.16. This file is booted when VisiCalc is to be used. The following sections contain information and procedures for this diskette. **USING THE BASIS BOOTER DISKETTE** describes the use of the BASIS BOOTER diskette for fast loading and system configuration.

An Apple Pascal APPLE 1 diskette is required to perform the procedures for adaptation of the Pascal Operating System, as well as the other operating systems. An Apple DOS 3.3 BASICS diskette (NOT the DOS 3.3 System Master), is required for the adaptation of Applesoft and Integer Basic. A Microsoft SoftCard 16-sector CP/M diskette is required for the adaptation of CP/M. The Apple Pascal APPLE 3 diskette is required if you have no other way to make copies of the APPLE 1 and BASIS BOOTER diskettes.

You should make copies of your diskette before performing any procedures. Section A.2 guides you through the copying of your diskette. Once you have performed the Pascal and Basic adaptation procedures, you can run the COPYA program on the DOS 3.3 SYSTEM MASTER diskette to make copies of Pascal, DOS 3.3, and CP/M diskettes. Refer to Apple's "THE DOS MANUAL" for instructions on the use of COPYA.

If other than normal indications are received during execution of the procedures in this appendix refer to the "APPLE PASCAL OPERATING SYSTEM MANUAL" for information on the operation of the Pascal system.

COPYING DISKETTES

The procedure in this section provides a method to make copies of your BASIS BOOTER and Pascal APPLE 1 diskettes. Steps 1 through 8 are used to format new diskettes. Steps 9 through 12 are used to copy the BASIS BOOTER and APPLE 1 diskettes on the newly formatted diskettes.

If you have already made copies of the BASIS BOOTER and APPLE 1 diskettes proceed to **APPLE PASCAL 1.1 ADAPTATION.**

Table A-1. Copying Diskettes

STEP

1. Place APPLE 1 (APPLE1:) in drive 1, the boot drive (the one that the red light comes on when the power is turned on). Place APPLE 3 (APPLE3:) in drive 2.

2. Turn the 108's POWER switch **ON**. The Pascal system will boot. Press **<RETURN>** and the screen will clear and the command line

Command: E(dit,R(un,F(ile,C(omp,L(ink,X(ecute, A(ssem, D(ebug ? [1.1]

will appear on the top line of the screen.

3. Run the Apple Pascal Formatter program.

   Type the letter **X** and respond to the prompt.

   Execute what file? **APPLE3:FORMATTER**

4. The system will then execute the formatter program and display the following message:

   APPLE DISK FORMATTER PROGRAM

   FORMAT WHICH DISK (4, 5, 9..12) ?

5. Remove APPLE3: from drive 2 (device #5) and insert a blank diskette.

6. Type the number **5** and press the **<RETURN>** key.

   The program checks to make sure you are not trying to reformat a previously formatted diskette. If you left APPLE3: in the drive you will be warned by the question

   DESTROY DIRECTORY OF APPLE3 ?

   Type the letter **N** for No, you will again be asked

   FORMAT WHICH DISK (4, 5, 9..12) ? **(return to Step 5.)**

Table A-1. Copying Diskettes (Continued)

STEP

7. If everything is proper, the drive whirrs and clacks, and the message

   NOW FORMATTING DISKETTE IN DRIVE 5

   appears on the monitor.

8. When formatting is complete, you will be prompted to specify the the next diskette to be formatted with the message

   FORMAT WHICH DISK (4, 5, 9..12) ?

   Remove the formatted diskette from drive 2, place a new diskette in drive 2, and return to Step 6. for as many diskettes as you wish to format.

   You should format a minimum of five diskettes. When you have formatted five or more diskettes, press **<RETURN>**, which causes Pascal to return to the command level. The message

   THAT'S ALL FOLKS...

   appears on the monitor.

   NOTE: If the program has trouble formatting a diskette, the following message is displayed:

   ERROR: UNABLE TO FORMAT DISK.
   DISKETTE WRITE PROTECTED,
   BAD MEDIA, OR BAD DRIVE.

   FORMAT WHICH DISK (4, 5, 9..12) ?

   Check the obvious causes, no diskette in drive, write protect tab on diskette, or improper insertion of the diskette.

   Sometimes this message occurs when an old diskette is reformated. Try formatting the diskette again. If the diskette being formatted still causes an error, it may be bad. If the problem always exists, check your drive.

9. You will now make a copies of the BASIS BOOTER and APPLE1: diskettes using the Pascal Filer. From the Pascal command level do the following:

   Type the letter F.

   The Filer is loaded and the command line

   Filer: G, S, N, L, R, C, T, D, Q [1.1]

   is displayed, thus enabling use of the Filer's diskette handling capabilities.

Table A-1. Copying Diskettes (Continued)

STEP

10. Place the BASIS BOOTER diskette in drive 1 (device #4). Place a formatted diskette in drive 2 (device #5).

    Type the letter **T** and respond to the prompts.

    Transfer ? **#4**
    To where ? **#5**
    Transfer 280 blocks ? (Y/N) **Y**
    Destroy BLANK: ? **Y**

    The Filer will return the message

    ZAP: --> BLANK:

    when the transfer is complete. Remove and label the diskette in drive 2 as a BASIS BOOTER copy.

11. Place the APPLE1: diskette in drive 1 (device #4). Place a formatted diskette in drive 2 (device #5).

    Type the letter T and respond to the prompts.

    Transfer ? **#4**
    To where ? **#5**
    Transfer 280 blocks ? (Y/N) **Y**
    Destroy BLANK: ? **Y**

    The Filer will return the message

    APPLE1: --> BLANK:

    when the transfer is complete. Remove and label the diskette in drive 2 as an APPLE1: copy.

12. You may make additional copies by following either Step 11 or 12. You only need one copy of each diskette to begin. Two copies of each diskette would be better.

APPLE PASCAL 1.1 ADAPTATION

This adaptation allows the Apple Pascal system to use 80 column display, uppercase/lowercase characters, keyboard features, and the printer and serial RS232 interfaces.

Follow the steps exactly as outlined in Table A-2 to perform the adaptation.

Table A-2. Apple Pascal 1.1 Adaptation

STEP

1. This is the actual beginning of the Pascal adaptation. You will begin by copying the file SYSTEM.APPLE from your APPLE1: diskette to the BASIS BOOTER (ZAP: to Pascal) diskette.

   If you came here from Table A-1, Step 12, press **<RETURN>** and proceed to Step 2. If you came here from Step 1, type the letter **F** and proceed to Step 2.

2. Type the letter T and respond to the following prompts.

   Transfer ? **APPLE1:SYSTEM.APPLE**
   To where ? **ZAP:SYSTEM.APPLE**

   The response

   APPLE1:SYSTEM.APPLE
    --> ZAP:SYSTEM.APPLE

   will appear at normal completion of the command.

3. Type the letter **Q**. This returns you to the Pascal command level.

4. Type the letter **X** and respond to the following prompts.

   Execute what file ? **ZAP:ZAP**

   The prompt - COMMAND 'console:' - appears at approximately the middle of the screen.

5. Type the following command.

   COMMAND 'ZAP:PASCAL'

   The ZAP program will now modify the file ZAP:SYSTEM.APPLE and produce a file named ZAP:NEW.APPLE. Normal completion of this operation is shown by the display of

   Normal end after line 87 (OLD BOOTER) or 126 (NEW BOOTER).

   *** Remember the line number you will need it in step 9.

6. You will now copy the file ZAP:NEW.APPLE to the APPLE1: diskette.

   Press **<RETURN>** to clear the screen.

   Type the letter F to enter the Filer.

Table A-2. Apple Pascal 1.1 Adaptation (Continued)

STEP

7. You will now transfer the adapted SYSTEM.APPLE to the APPLE1: diskette.

   Type the letter **T** and respond to the prompts.

   ```
   Transfer ?  ZAP:NEW.APPLE
   To where ?  APPLE1:SYSTEM.APPLE
   Remove old APPLE1:SYSTEM.APPLE ? Y
   ```

   The normal completion of this command is

   ```
   ZAP:NEW.APPLE
    --> APPLE1:SYSTEM.APPLE
   ```

8. You will now transfer the file ZAP:108.MISCINFO to the APPLE1 diskette.

   Type the letter **T** and respond to the prompts.

   ```
   Transfer ? ZAP:108.MISCINFO
   To where ? APPLE1:SYSTEM.MISCINFO
   Remove old APPLE1:SYSTEM.MISCINFO ? Y
   ```

   The normal completion of the command is

   ```
   ZAP:108.MISCINFO
    --> APPLE1:SYSTEM.MISCINFO
   ```

9. In Step 5 if the Normal end was after 87, (OLD BOOTER) perform this step. If it was not, proceed to step 10. The ZAP:NEW.APPLE file has to be renamed on the ZAP: diskette.

   Press **<RETURN>** to clear the screen.

   Type the letter C and respond to the prompts.

   ```
   Change ? ZAP:NEW.APPLE
   Change to what ? ZAP:PCODE
   ```

   The normal completion of the command is

   ```
   ZAP:NEW.APPLE     -->PCODE
   ```

10. Now you will remove the .APPLE files from the ZAP: diskette.

    Press **<RETURN>** to clear the screen.

    Type the letter R and respond to the prompts.

Table A-2. Apple Pascal 1.1 Adaptation (Continued)

STEP

10. (Continued)

    Remove ? **ZAP:=.APPLE**

    The normal response to this command is

    ```
    ZAP:SYSTEM.APPLE          -->REMOVED
    ZAP:NEW.APPLE             -->REMOVED     *** only on NEW BOOTER
    ```

    Update directory ? **Y**

11. This step is required only for the OLD BOOTER diskette. For proper operation of the ZAP: diskette as the BASIS BOOTER, the diskette must be "Crunched". This operation moves all available disk space to one contiguous block.

    Press **<RETURN>** to clear the screen.

    Type the letter **K** and respond to the prompts.

    ```
    Crunch ? ZAP:
    From end of disk, Block 280 ? (Y/N) Y
    ```

    The normal response to this command is

    ```
    Moving forward PCODE
    ZAP: Crunched
    ```

12. Remove the diskette in drive 1 (device #4) and label it BASIS 108 APPLE1:. Remove the diskette in drive 2 (device #5) and label it **WORKING BASIS BOOTER.**

13. Turn the 108's power switch **OFF.**

14. Place the diskette labeled BASIS 108 APPLE1: in drive 1 (device #4).

    Turn the 108's power switch **ON.**

    The Pascal system should boot and display in full (normal) 80 column mode.

    Perform the procedure in Table A-1 to make at least one copy of your BASIS 108 APPLE1: diskette. Label the copy accordingly.

Pascal Capabilities

The adaptation you just completed allows the Pascal operating system to use the 80 column display, upper/lower case characters, keyboard features, and the printer and serial RS232 interfaces. The small up, down, left and right arrow keys are recognized by the editor for cursor control. The large left and right arrows keys are recognized as the Apple left and right arrow keys. The 'HOME' key is recognized as the Editor-Accept in place of CTRL-C.

In addition, if you used the OLD Version of the BASIS BOOTER diskette Function key 'F15' is used for the BREAK in place of CTRL-@.

With the NEW version of the BASIS BOOTER diskette Shift-CTRL-F15 is used for BREAK in place of CTRL-@. Shift-F1 is for STOP EXECUTION in place of CTRL-S. Shift-F2 is used for FLUSH OUTPUT in place of CTRL-F.

APPLESOFT AND INTEGER BASIC ADAPTATION

Performing the procedure in Table A-3 adapts Applesoft and Integer Basic to use the full capabilities of the BASIS 108. Follow the steps exactly.

Table A-3. Applesoft and Integer Basic Adaptation

STEP

1. The Pascal operating system is used to perform this procedure. The BASIS BOOTER and APPLE1: diskette adapted in Table A-2 are used. Place the APPLE1: diskette in drive 1 (device #4) and the BASIS BOOTER diskette in drive 2 (device #5). Turn the 108's power switch **ON** to boot the system.

2. You will now copy the FPBAS.DATA and INTBAS.DATA files on the DOS 3.3 BASICS diskette **(NOT the DOS 3.3 SYSTEM MASTER)** to the BASIS BOOTER diskette.

   Type the letter **F**.

   Place the DOS 3.3 BASICS diskette (BASICS:) in drive 1 (device #4).

   Type the letter **T** and respond to the prompts.

   ```
   Transfer ?  BASICS:=BAS.DATA
   To where ?  ZAP:$
   ```

   The normal response to the command is

   ```
   BASICS:INTBAS.DATA
    --> ZAP:INTBAS.DATA
   BASICS:FPBAS.DATA
    --> ZAP:FPBAS.DATA
   ```

   Place the APPLE1: diskette in drive 1 (device #4) and type the letter Q to return to the Pascal command level.

Table A-3. Applesoft and Integer Basic Adaptation (Continued)

STEP

3. Now you will execute the file ZAP to adapt both basics.

   Type the letter **X** and respond to the prompts.

   Execute what file ? **ZAP:ZAP**

4. The prompt - COMMAND 'console:' - appears in approximately the middle of the screen.

5. Type the following command.

   COMMAND 'ZAP:BASIC'

   The ZAP program will now modify the files and produce the files FP40, FP80, and INT40 (and INT80 with the NEW BASIS BOOTER diskette). The display at the normal completion of this command is

   Normal end at line 174 (OLD BOOTER) or 173 (NEW BOOTER)

6. The BASIS BOOTER diskette is now configured for fast loading of the Basic languages.

Basic Capabilities

This adaptation creates FP40 and FP80 which have greatly enhanced capabilities, and corrects several Applesoft problems. The following table lists these capabilities.

Table A-4. FP Enhancements

1. The INPUT command will accept both upper and lowercase input.
2. The GET command can read the FUNCTION keys.
3. Full 80 column operation is supported in text and mixed mode graphics.
4. Keywords and variables can be entered in lowercase.
5. The problem FOR I= S TO P is no longer interpreted as FOR I=STOP.
6. TAB(..), SPC(..), and HTAB always remain in the window (refer to the Applesoft Manual, page 129).

Basic Capabilities (Continued)

Additionally, if the NEW BASIS BOOTER was used, FP40 and FP80 have an extended command set. New commands have been added to the FP languages to provide ease in character set selection, 80/40 column switching, and use of flashing and inverse characters. INT80 is an 80 column version of INT40.

The commands LEFT$ (A$,0) and RIGHT$ (A$,0) return a 'null' string and generate no error message.

The following tables list the new commands and a short explanation of each. Run the NEW FP DEMO on the UTILITY diskette for a tutorial demonstration of the commands.

Table A-5. New FP80 Commands

1. TEXT 40 - switches to 40 column display
2. TEXT 80 - switches to 80 column display

Table A-6. New FP80 and FP40 Commands

1. NORMAL 0 - select character set 0 (Apple)
2. NORMAL 1 - select character set 1 (ASCII)
3. NORMAL 2 - select character set 2 (German)
4. NORMAL 3 - select character set 3 (APL)
5. NORMAL 4 - select character set 4 (Full ASCII)
6. FLASH! - used with character set 4
7. INVERSE! - used with character set 4

NOTES: FLASH and INVERSE are used with sets other than character set 4. NORMAL removes the FLASH!, INVERSE!, FLASH or INVERSE condition and leaves you in the selected character set.

CP/M ADAPTATION

This adaptation allows CP/M to use the 80 column screen, printer interface and the serial RS232 interface. Perform the procedure in Table A-7 to adapt your 16-Sector CP/M Diskette.

Table A-7. CP/M Adaptation

STEP

1. You will now adapt CP/M for enhanced use in the BASIS 108. The Pascal operating system is used to perform this procedure. Place your adapted APPLE1: diskette in drive 1 (device #4) and the BASIS BOOTER diskette in drive 2 (device #5). Boot the system.

2. You will now run the ZAP program to modify your CP/M diskette.

   Type the letter **X** and respond to the prompts.

   Execute what file ? **ZAP:ZAP**

   The prompt - COMMAND 'console:' - appears in approximately the middle of the monitor screen.

3. Remove the APPLE1: diskette from drive 1 (device #4) and place a copy of your CP/M diskette in drive 1 (device #4 to Pascal and drive A: to CP/M).

4. Enter the following command.

   COMMAND 'ZAP:CPM'

   The normal response to the command is

   Insert CP/M disk in drive "A:" (#4: or S6, D1) and press <RETURN>

   Press **<RETURN>**.

   After several seconds when the operation is completed the message

   Press <RESET> for CP/M test or <RETURN> for Pascal!

5. If you desire to return to Pascal, place APPLE1: in drive 1 (device #4) and press return. Otherwise, <RESET> the system to run CP/M. (To RESET the system, press the shift and control keys with your left hand and the shift key with your right hand and release any one or all of the keys.)

CP/M Capabilities

The logical device LST: can be assigned as LPT: (parallel printer interface) or UL1: (serial RS232 output). The PUN: can be assigned UP1: and the RDR: can be assigned UR1: for serial communications. The serial interface is initialized for 9600 baud, 8 data bits, 2 stop bits and no parity.

The serial interface baudrate can be changed by changing the value in address $F280. Digital Research's DDT (CP/M DYNAMIC DEBUGGING TOOL) can be used to change the value of address $F280. Table A-8 lists the HEX number and corresponding baudrate. Table A-9 contains a procedure that can be used to change this address.

Table A-8. CP/M Serial Baudrate

HEX NUMBER	BAUDRATE
$91	50
$92	75
$93	110
$94	134.5
$95	150
$96	300
$97	600
$98	1200
$99	1800
$9A	2400
$9B	3600
$9C	4800
$9D	7200
$9E	9600
$9F	19200

The DDT S (Set) command allows memory locations to be examined and optionally altered. The form of the command is

Sx

where x is the hexadecimal starting address for examination and alteration of memory. DDT responds with a numeric prompt, giving the memory location, along with the data currently held in the memory location. If you type a carriage return, then the data is not altered. If a byte value is typed, then the value is stored in the prompted address. In either case, DDT continues to prompt with successive addresses and values until either a period (.) is typed, or an invalid input value is detected.

Table A-9 contains a procedure to change the serial interface baudrate byte at address $F280. This procedure uses DDT.

Table A-9. CP/M Address Contents Change

STEP

1. From the CP/M command level enter **DDT <RETURN>**

2. The screen prompt - appears.

3. Enter **SF280 <RETURN>**.

4. The display now shows the address and it's contents. The display is now similar to:

```
-SF280
F280 00
```

5. Enter the value corresponding to the baudrate required from Table A-8. $98 for 1200 baud in this example.

   F280 00 **98 <RETURN>**

6. The display is then updated. It appears similar to:

```
-SF280
F280 00 98
F281 00
```

7. Enter **. <RETURN>** (a period and <RETURN>). This returns you to the DDT command level prompt -.

8. A **CTRL-C** at this time returns you to the CP/M command level.

USING THE BASIS BOOTER DISKETTE

When the BASIS BOOTER diskette is booted it displays a menu. The menu will appear similiar to:

```
   Interpreter files:

A: VC.16
B: INTBAS.DATA      *** PCODE on OLD BOOTER
C: FPBAS.DATA
D: FP40
E: FP80
F: INT40
G: INT80            *** not available on OLD BOOTER

     Select (A..G):
```

The exact sequence, type and number of files in the menu depends upon the BASIS BOOTER diskette used. The file VC.16 is always present. Select the letter representing the file you wish to load. When the file is loaded a screen prompt will appear.

Except for PCODE the prompt "INSERT YOUR 16 SECTOR DISK AND PRESS <RETURN>" appears. Insert the appropriate diskette at this time and press the RETURN key. Table A-10 briefly describes the files.

Table A-10. BOOTER File Descriptions

NAME	DESCRIPTION
VC.16	Booted when Visicalc is to be used.
PCODE	The adapted SYSTEM.APPLE *** OLD BOOTER only
INTBAS.DATA	Unchanged Integer Basic.
FPBAS.DATA	Unchanged Applesoft.
FP40	Adapted Applesoft, 40 column operation.
FP80	Adapted Applesoft, 80 column operation.
INT40	Adapted Integer Basic, 40 column operation.
INT80	Adapted Integer Basic, 80 column operation. *** NEW BOOTER only

The VC.16 file is used to configure the BASIS 108 for operation with Visicalc. The other files can be loaded as required.

APPENDIX B

BASIS UTILITY DISKETTE

BASIS UTILITY DISKETTE

The BASIS UTILITY diskette contains programs for Pascal 1.1, Floating Point Basic (FP), and CP/M. A catalog or directory in each respective language shows the programs available for that particular language. This diskette is not bootable. A copy of the diskette should be used. A short synopsis is provided for each program. Listings of the FP programs are included.

Pascal 1.1 Utilities

There are three programs that can be executed. A short synopsis of each program follows.

DISPLAY

When executed this program places a display of the character sets on the screen. It allows selection of the character set to be used and allows a blinking, or inverse cursor. The screen is self prompting.

DISPLAY.A2

The same as display above for use with older revision (A2) computerboards.

FORMAT40

This program formats diskettes. It will format 35 and 40 track diskettes for Apple Pascal, BASIS Test, and Softech (Pascal IV.0) diskettes. The program is self prompting.

Floating Point (FP) Basic Utilities

There are nine programs listed on the diskette. A short synopsis of each program follows.

CHAIN UPDATE

This program updates the CHAIN program on the DOS 3.3 System Master diskette to use features of the BASIS 108. Place a copy of CHAIN on a diskette and place it in drive 1. Run CHAIN UPDATE.

Floating Point (FP) Basic Utilities (Continued)

CHRGEN

When executed this program cycles through the character sets and displays them on the screen. Pressing the space bar stops the program with the displayed character set enabled.

COLOR DEM0108

Shows off the color capabilities of the 80 column Medium Resolution screen. Try it!

DOS PATCH

This program modifies DOS 3.3. It enables the IN#9 and PR#9 commands, and allows the use of lowercase command entry to DOS.

NEW FP DEMO

This program takes you through a prompted demonstration of the commands added to FP80 for use with the 25-120 monitor.

PRINTER/RS232
Printer/RS232 Part 1
Printer/RS232 Part 2

These programs are used to configure the serial interface. They are self prompting and display a list of pokes that can be included in your program. Run the PRINTER/RS232 program. It determines if Part 1 or Part 2 should be run, and runs the appropriate program.

THE FOLLOWING PROGRAM VERSIONS ONLY ARE FOUND ON OLDER UTILITY DISKETTES

Check the PRINTER/RS232 program. If it has a line 11 GOTO 100, delete line 11 and save the program.

***** PRINTER/RS232 and PRINTER/RS232 V2.0 *****

When executed these programs configure the serial interface. The program also displays the pokes required for inclusion in a program. Either one of these programs may be executed. They are cross linked so that the proper version is executed dependent upon machine configuration.

CP/M Utilities

There are four programs that can be executed. A short synopsis of each follows.

ASCII.COM

Changes the display to the ASCII character set.

DEUTSCH.COM

Changes the display to the German ASCII character set.

APL.COM

Changes the display to the APL character set.

REBOOT.COM

When this program is executed it allows another language to be booted when the system is RESET. It is not necessary to power the system down to change languages.

PROGRAM LISTINGS

The remainder of this Appendix contains directories, catalogs, and listings for the BASIS UTILITY DISKETTE.

```
UT108:
RESERVED            6  4-Aug-82   10 Data
DOS_T2              8  4-Aug-82   16 Bad
CP/M               64  4-Aug-82   24 Bad
DOS_3.3            56  4-Aug-82   88 Bad
DISPLAY.TEXT        4  4-May-82  144 Text
FORMAT40.6502       3 29-May-82  148 Data
FORMAT40.CODE       8  1-Jul-82  151 Code
FORMAT40.BOOT       6  1-Jul-82  159 Data
DISPLAY.A2.CODE     2 15-Feb-82  165 Code
DISPLAY.A2.TEXT     4 15-Feb-82  167 Text
DISPLAY.CODE        2 10-May-82  171 Code
6551.TEXT           6 16-Feb-82  173 Text
< UNUSED >        101            179
12/12 files, 101 unused, 101 in largest
```

```
dir b:
B: APL      COM : DEUTSCH  COM : ASCII    COM : REBOOT   COM
```

```
DISK VOLUME 001

 A 002 CHAIN UPDATE
 A 002 CHRGEN
 A 009 COLOR DEMO108
 A 003 DOS PATCH
 A 004 NEW FP DEMO
 A 003 PRINTER/RS232
 A 008 Printer/RS232 Part 1
 A 008 Printer/RS232 Part 2
 A 003 RENUMBER UPDATE
```

CHAIN UPDATE

```
10    PRINT "DOS 3.3 Chain Update"
20    PRINT CHR$ (4)"BLOAD CHAIN,A$1208"
30    FOR A = 4826 TO 4831
40      POKE A, PEEK (A) - 128
50    NEXT
60    PRINT CHR$ (4)"UNLOCK CHAIN"
65    PRINT CHR$ (4)"BSAVE CHAIN,A$1208,L$1C8"
70    PRINT CHR$ (4)"LOCK CHAIN"
80    END
```

CHRGEN

```
10    TEXT
    : HOME
15    INVERSE
20    FOR L = 0 TO 7
25      IF L > 3 THEN NORMAL
30      PRINT SPC( 66);
    :   LA = PEEK (40) + 256 * PEEK (41)
50      FOR C = 0 TO 31
    :     POKE LA + C + 1,L * 32 + C
    :   NEXT C
60      PRINT
70    NEXT L
80    NORMAL
100   FOR L = 0 TO 1
110     FOR H = 0 TO 1
120       POKE 49154 + L,0
    :   POKE 49156 + H,0
140     FOR T = 0 TO 300
    :     IF PEEK (49152) > 127 THEN GET A$
  * :     END
150     NEXT T
160   NEXT H,L
170   GOTO 100
```

```
COLOR DEMO108

30    U = 9
    : D = 3
    : E = - 16384
    : F = - 16368
    : G = - 16287
    : H = - 16286
    : Z = 127
100   POKE 768,173
    : POKE 769,48
    : POKE 770,192
    : POKE 771,136
    : POKE 772,208
    : POKE 773,4
    : POKE 774,198
    : POKE 775,7
110   POKE 776,240
    : POKE 777,8
    : POKE 778,202
    : POKE 779,208
    : POKE 780,246
    : POKE 781,166
    : POKE 782,6
    : POKE 783,76
    : POKE 784,0
    : POKE 785,03
    : POKE 786,96
200   TEXT
    : HOME
    : PRINT
    : POKE F,0
220   A$ = "108   DEMONSTRATION PROGRAMS"
    : GOSUB 2000
    : PRINT
230   PRINT "    TO OPERATE A DEMONSTRATION,  TYPE"
240   PRINT "ITS NUMBER.  THEN TYPE THE KEY MARKED"
250   PRINT "'RETURN' AT THE RIGHT EDGE OF THE KEY-"
260   PRINT "BOARD.  TYPE THE 'RETURN' KEY TO STOP"
270   PRINT "ANY DEMONSTRATION."
    : PRINT
    : PRINT
290   PRINT "1.";
300   A$ = "STANDARD COLOR NAMES"
    : GOSUB 2000
310   PRINT "2.";
320   A$ = "STANDARD COLOR NUMBERS"
    : GOSUB 2000
330   PRINT "3.";
340   A$ = "KALEIDOSCOPE"
    : GOSUB 2000
350   PRINT "4.";
360   A$ = "SKETCHING SCREEN"
    : GOSUB 2000
```

```
370   PRINT
    : PRINT
380   INPUT "WHICH WOULD YOU LIKE?";A$
390   IF LEN (A$) = 0 THEN GOTO 200
395   N = ASC (A$) - 48
400   IF N < 1 OR N > 4 THEN GOTO 200
410   ON N GOTO 500,600,700,800
450   IF PEEK (E) < Z THEN 450
460   GOTO 200
500   GOSUB 3000
510   A$ = "BMDPDGMLBOGPLYAW"
    : GOSUB 4000
    : PRINT
520   A$ = "LGBUGRBBRRRIGEQI"
    : GOSUB 4000
    : PRINT
530   A$ = "ATLRRELLWNENRLUT"
    : GOSUB 4000
    : PRINT
540   A$ = "KAUPNYUUNGYKNOAE"
    : GOSUB 4000
550   GOTO 450
600   GOSUB 3000
610   FOR I = 0 TO 14 STEP 2
620     HTAB 2 * I + 4
    :   PRINT I;
630   NEXT I
    : PRINT
640   FOR I = 1 TO 15 STEP 2
650     HTAB 2 * I + 4
    :   PRINT I;
660   NEXT I
    : PRINT
    : PRINT
    : GOTO 450
700   GR
    : HOME
    : FOR W = 3 TO 50
    :   FOR I = 1 TO 19
    :     FOR J = 0 TO 19
    :       K = I + J
    :       IF PEEK (E) > Z THEN 200
750         COLOR= J * 3 / (I + 3) + I * W / 12
751         PLOT I,K
    :       PLOT K,I
    :       PLOT 40 - I,40 - K
    :       PLOT 40 - K,40 - I
752         PLOT K,40 - I
    :       PLOT 40 - I,K
    :       PLOT I,40 - K
    :       PLOT 40 - K,I
760         COLOR= J * 3 / (I + 3) + I * W / 12 + 5
761         PLOT I + 39,K
    :       PLOT K + 39,I
```

```
    :       PLOT 80 - I,40 - K
    :       PLOT 80 - K,40 - I
762         PLOT K + 39,40 - I
    :       PLOT 80 - I,K
    :       PLOT I + 39,40 - K
    :       PLOT 80 - K,I
780     NEXT J,I,W
    : GOTO 450
800   HOME
    : A$ = "USE THE GAME CONTROLS AND BUTTONS."
    : GOSUB 2000
810   A$ = "THE BUTTONS CLEAR THE TV AND CHANGE"
    : GOSUB 2000
820   A$ = "THE DISPLAYED COLOR.  TRY ONE NOW."
    : GOSUB 2000
900   IF PEEK (G) > Z OR PEEK (H) > Z THEN 930
910   IF PEEK (E) > Z THEN 200
920   GOTO 900
930   HOME
    : GR
940   P = PDL (0)
    : X = INT (P / 6)
    : IF X > 39 THEN X = 39
950   I = PDL (1)
    : Y = INT (I / 6)
    : IF Y > 39 THEN Y = 39
960   P = INT ((I + P) / 2 + 9)
    : IF P > 255 THEN P = 255
970   GOSUB 1500
980   VTAB 21
    : PRINT "X IS ";X;" "
    : PRINT "Y IS ";Y;" "
990   IF PEEK (E) > Z THEN 200
1000  IF PEEK (G) > Z THEN GR
1010  IF PEEK (H) > Z THEN U = (U + 1)
1015  IF U = 16 THEN U = 0
1020  COLOR= U
    : PLOT X,Y
    : GOTO 940
1500  POKE 6,P
    : POKE 7,D
    : CALL 768
    : RETURN
2000  HTAB ((40 - LEN (A$)) / 2 + 1)
    : PRINT A$
    : PRINT
    : RETURN
3000  GR
    : FOR I = 0 TO 31
    :   COLOR= I / 2
    :   VLIN 0,39 AT I + 3
    : NEXT I
3010  COLOR= 5
    : VLIN 0,39 AT 2
```

```
    : VLIN 0,39 AT 35
    : HLIN 2,35 AT 0
    : HLIN 2,35 AT 39
    : RETURN
4000  FOR I = 1 TO LEN (A$)
4010    HTAB 2 * I + 2
    :   PRINT MID$ (A$,I,1);
4020  NEXT I
    : RETURN
```

DOS PATCH

```
10    REM  DOS PATCHES
20    POKE 42569,255
    : REM  Lower case input from TEXT files is legal!
30    POKE 41153,10
    : REM  PR#9 and IN#9 are now legal!
40    POKE 41380,32
    : POKE 41381,120
    : POKE 41382,157
    : REM  JSR $9D78
50    FOR I = 40312 TO 40323
    :   READ P
    :   POKE I,P
    : NEXT I
    : REM  lowercase DOS commands are now legal!
60    DATA 32,147,161,240,6,201,224,144,2,41,223,96
999   PRINT "DOS 3.3 is patched !"
1000  END
```

```
NEW FP DEMO

10    ONERR GOTO 1000
11    HOME
    : PRINT "Press any key to advance the display at each pause.  ";
    : GET A$
15    TEXT 80
    : HOME
20    TEXT 40
    : LIST 20
    : GET A$
    : REM 40 Column display
30    TEXT 80
    : LIST 30
    : GET A$
    : REM 80 Column display
90    HOME
    : GOSUB 10000
100   NORMAL 0
    : LIST 100
    : GET A$
    : REM APPLE
110   NORMAL 1
    : LIST 110
    : GET A$
    : REM ASCII
120   NORMAL 2
    : LIST 120
    : GET A$
    : REM National
130   NORMAL 3
    : LIST 130
    : GET A$
    : REM apl
140   NORMAL 1
150   HOME
    : GOSUB 10000
160   INVERSE
    : GOSUB 10000
    : NORMAL
170   INVERSE
    : LIST 170
    : GET A$
    : REM  96 normal + 96 inverse
180   FLASH
    : LIST 180
    : GET A$
    : REM 96 normal + 96 blinking
200   HOME
    : NORMAL
    : GOSUB 10000
210   INVERSE !
    : GOSUB 10100
220   FLASH !
```

```
    : GOSUB 10100
230   NORMAL 4
    : LIST 200,240
    : GET A$
240   REM 96 normal ASCII + 64 inverse + 64 blinking
300   PRINT
    : PRINT "QUIT ?   y/n   ";
310   GET A$
    : IF A$ = "Y" OR A$ = "y" THEN PRINT
  * : END
320   GOTO 11
1000  ER = PEEK (222)
    : EL = PEEK (218) + PEEK (219) * 256
1020  IF ER = 16 AND EL = 15 THEN PRINT "Please use FP80 !"
  * : STOP
  * : END
1030  POKE 216,0
    : RESUME
10000 FOR C = 32 TO 127 STEP 32
    :   FOR CC = C TO C + 31
    :     PRINT CHR$ (CC);
    :   NEXT
    :   PRINT
    : NEXT
    : PRINT
10010 RETURN
10100 FOR C = 32 TO 95 STEP 32
    :   FOR CC = C TO C + 31
    :     PRINT CHR$ (CC);
    :   NEXT
    :   PRINT
    : NEXT
    : PRINT
10110 RETURN
```

PRINTER/RS232

```
1     REM This program uses Printer/RS232 Part 1 or 2, and DOS PATCH.
10    TEXT
    : HOME
    : PRINT "RS232 SET-UP  -  ";
20    X = PEEK (49414)
    : X$ = ""
30    IF X = 0 THEN X$ = "0.0"
31    IF X = 16 THEN X$ = "1.0"
32    IF X = 32 THEN X$ = "2.0"
33    IF X = 33 THEN X$ = "2.1"
34    IF X$ = "" GOTO 60
35    PRINT "ROM Version ";X$
40    IF X = 32 OR X = 33 THEN PRINT "(cD)RUN Printer/RS232 Part 2"
50    PRINT "(cD)RUN Printer/RS232 Part 1"
60    PRINT "Unknown ROM version !"
```

```
100   DIM BR$(16)
110   PRINT
    : PRINT "Pressing <RETURN> to the following questions produces:"
    : PRINT
120   FOR ZZ = 1 TO 48
    :   READ ZZ$
122     IF ZZ = 1 THEN PRINT ZZ$;" ............... YES"
124     IF ZZ = 6 THEN PRINT ZZ$;" ................. YES"
126     IF ZZ = 11 THEN PRINT ZZ$;" ............................... 9600"
128     IF ZZ = 29 THEN PRINT ZZ$;" ............................... 8"
130     IF ZZ = 36 THEN PRINT ZZ$;" ............................ NO"
132     IF ZZ = 48 THEN PRINT ZZ$;" ............................... 2"
134   NEXT
    : RESTORE
140   PRINT
    : PRINT "Enter <RETURN> for preset data  -OR-  for other data:"
    : PRINT "<y>, <n> or <value>, and <RETURN> to the questions."
    : PRINT
200   LF = 0
    : LX = 165
    : CT = 16
    : CM = 8
1000  GOSUB 9000
    : LF = LF + D * 128
    : LX = LX - D * 128
1100  GOSUB 9000
    : LF = LF + D * 64
    : LX = LX + D * 64
1200  GOSUB 9000
    : CT = CT + 1 + D
1300  GOSUB 9000
    : CT = CT + D * 32
    : WL = D
1400  GOSUB 9000
    : CM = CM + D * 32
    : P = D
1500  IF P = 0 THEN FOR I = 1 TO 7
  * :   READ C$
  * : NEXT
  * : GOTO 1600
1510  GOSUB 9000
    : CM = CM + D * 64
1550  IF WL = 0 AND P = 1 THEN FOR I = 1 TO 10
  * :   READ C$
  * : NEXT
  * : GOTO 1700
1600  IF WL = 3 AND P = 0 THEN FOR I = 1 TO 5
  * :   READ C$
  * : NEXT
  * : SK = 1
1610  GOSUB 9000
    : CT = CT + D * 128
    : IF NOT SK THEN FOR I = 1 TO 5
  * :   READ C$
  * : NEXT
```

```
1700  REM
4000  PRINT "poke 1529,";LX
    : POKE 1529,LX
4010  PRINT "poke 1657,";LF
    : POKE 1657,LF
4040  PRINT "poke 1785,";CT
    : POKE 1785,CT
4050  PRINT "poke 1913,";CM
    : POKE 1913,CM
4060  PRINT "poke 2041,";CM + 1
    : POKE 2041,CM + 1
5000  PRINT "(cD)RUN DOS PATCH"
5010  END
9000  REM  MENU
9010  READ Q$
    : I = 0
    : PRINT Q$;" (";
    : HX = PEEK (36)
9020  READ BR$(I)
    : IF BR$(I) = "" THEN 9100
9030  IF I > 0 THEN PRINT ",";
  * : IF PEEK (36) + 15 > PEEK (33) THEN PRINT
  * : POKE 36,HX
9040  PRINT BR$(I);
    : I = I + 1
    : GOTO 9020
9100  READ D
    : INPUT ") ? ";A$
9115  AN$ = ""
    : IF A$ = "" THEN 9200
9120  FOR L = 1 TO LEN (A$)
    :   C$ = MID$ (A$,L,1)
    :   IF "A" < = C$ AND C$ < = "Z" THEN C$ = CHR$ ( ASC (C$) + 32)
9130    AN$ = AN$ + C$
9140  NEXT
9200  FOR I = 0 TO 15
9210    IF BR$(I) = "" THEN I = 999
  * :   GOTO 9290
9220    IF BR$(I) = AN$ THEN D = I
  * :   I = 999
9290  NEXT I
9300  PRINT Q$": "BR$(D)
    : PRINT
9500  RETURN
10000 DATA Printer CR->CR/LF translation,n,y,,1
10010 DATA RS232 CR->CR/LF translation,n,y,,1
10020 DATA RS232 Baudrate
10021 DATA 50,75,110,134.5,150,300,600,1200,1800,2400,3600,4800,7200,9600,19
      200,,13
10030 DATA RS232 Databits,8,7,6,5,,0
10040 DATA RS232 Paritybits,n,y,,0
10050 DATA RS232 Parity,odd,even,mark,space,,3
10100 DATA RS232 Stopbits,1,2,,1
10110 DATA RS232 Stopbits,1,1.5,,1
```

Printer/RS232 Part 2

```
100   DIM BR$(16)
110   PRINT
    : PRINT "Pressing <RETURN> to the following questions produces:"
    : PRINT
120   FOR ZZ = 1 TO 48
    :   READ ZZ$
122     IF ZZ = 1 THEN PRINT ZZ$;" ..... YES"
124     IF ZZ = 6 THEN PRINT ZZ$;" ..... YES"
126     IF ZZ = 11 THEN PRINT ZZ$;" ..... 9600
          "
128     IF ZZ = 29 THEN PRINT ZZ$;" ..... 8"
130     IF ZZ = 36 THEN PRINT ZZ$;" ..... NO"
132     IF ZZ = 48 THEN PRINT ZZ$;" ..... 2"
134   NEXT
    : RESTORE
140   PRINT
    : PRINT "Enter <RETURN> for preset value  -OR-  for other data:"
    : PRINT "<y>, <n>, <value> and <RETURN> to the questions."
    : PRINT
200   LF = 0
    : LX = 165
    : CT = 16
    : CM = 11
1000  GOSUB 9000
    : LF = LF + D * 128
    : LX = LX - D * 128
1100  GOSUB 9000
    : LF = LF + D * 64
    : LX = LX + D * 64
1200  GOSUB 9000
    : CT = CT + 1 + D
1300  GOSUB 9000
    : CT = CT + D * 32
    : WL = D
1400  GOSUB 9000
    : CM = CM + D * 32
    : P = D
1500  IF P = 0 THEN FOR I = 1 TO 7
  * :   READ C$
  * : NEXT
  * : GOTO 1600
1510  GOSUB 9000
    : CM = CM + D * 64
1550  IF WL = 0 AND P = 1 THEN FOR I = 1 TO 10
  * :   READ C$
  * : NEXT
  * : GOTO 1700
1600  IF WL = 3 AND P = 0 THEN FOR I = 1 TO 5
  * :   READ C$
  * : NEXT
  * : SK = 1
1610  GOSUB 9000
    : CT = CT + D * 128
    : IF NOT SK THEN FOR I = 1 TO 5
  * :   READ C$
  * : NEXT
```

```
1700  REM
4000  PRINT "poke 1529,";LX
    : POKE 1529,LX
4010  PRINT "poke 1657,";LF
    : POKE 1657,LF
4040  PRINT "poke 1785,";CT
    : POKE 1785,CT
4050  PRINT "poke 1913,";CM
    : POKE 1913,CM
4060  PRINT "poke 2041,";0
    : POKE 2041,0
5000  PRINT "(cD)RUN DOS PATCH"
5010  END
9000  REM  MENU
9010  READ Q$
    : I = 0
    : PRINT Q$;" (";
    : HX = PEEK (36)
9020  READ BR$(I)
    : IF BR$(I) = "" THEN 9100
9030  IF I > 0 THEN PRINT ",";
  * : IF PEEK (36) + 15 > PEEK (33) THEN PRINT
  * : POKE 36,HX
9040  PRINT BR$(I);
    : I = I + 1
    : GOTO 9020
9100  READ D
    : INPUT ") ? ";A$
9115  AN$ = ""
    : IF A$ = "" THEN 9200
9120  FOR L = 1 TO LEN (A$)
    :   C$ = MID$ (A$,L,1)
    :   IF "A" < = C$ AND C$ < = "Z" THEN C$ = CHR$ ( ASC (C$) + 32)
9130    AN$ = AN$ + C$
9140  NEXT
9200  FOR I = 0 TO 15
9210    IF BR$(I) = "" THEN I = 999
  * :   GOTO 9290
9220    IF BR$(I) = AN$ THEN D = I
  * :   I = 999
9290  NEXT I
9300  PRINT Q$": "BR$(D)
    : PRINT
9500  RETURN
10000 DATA Printer and RS232 CR->CR/LF translation,n,y,,1
10010 DATA Printer and RS232 echo to screen,n,y,,1
10020 DATA RS232 Baudrate
10021 DATA 50,75,110,134.5,150,300,600,1200,1800,2400,3600,4800,7200,9600,19
        200,,13
10030 DATA RS232 Databits,8,7,6,5,,0
10040 DATA RS232 Paritybit,n,y,,0
10050 DATA RS232 Parity,odd,even,mark,space,,3
10100 DATA RS232 Stopbits,1,2,,1
10110 DATA RS232 Stopbits,1,1.5,,1
```

RENUMBER UPDATE

```
1     PRINT "Renumber Update"
10    D$ = CHR$ (4)
15    PRINT D$"MON I,O,C"
20    PRINT D$"OPEN EXEC"
30    PRINT D$"WRITE EXEC"
40    PRINT "LOAD RENUMBER
50    PRINT "CALL-151
60    PRINT "B84LL
70    PRINT "B87:0
80    PRINT "B84LL
90    PRINT "Q
100   PRINT "UNLOCK RENUMBER
110   PRINT "SAVE RENUMBER
120   PRINT "LOCK RENUMBER
125   PRINT "NEW
130   PRINT "DELETE EXEC
140   PRINT D$"CLOSE
150   PRINT D$"EXEC EXEC
```

APPENDIX C

MONITOR LISTINGS

This appendix contains listings for the System Monitors used with the BASIS 108. The 25-120 and 25-46 monitors are the current versions.

Pages C-2 through C-35 list the 25-120 monitor.

Pages C-36 through C-74 list the 25-46 monitor.

Pages C-75 through C-88 list the differences between the 25-46 and 25-86 monitor. The 25-46 listing contains complete detail about both monitor versions. The 25-86 monitor is a conditional assembly of the 25-46 listing.

```
0000:          2 ****************************************
0000:          3 *                                      *
0000:          4 *     BASIS 108 MONITOR                *
0000:          5 *                                      *
0000:          6 *     25-120                           *
0000:          7 *                                      *
0000:          8 *     (C) COPYRIGHT 1982               *
0000:          9 *     BASIS Incorporated               *
0000:         10 *                                      *
0000:         11 *     ALL RIGHTS RESERVED              *
0000:         12 *                                      *
0000:         13 ****************************************
0001:         14 CFONT1  EQU  1          ;1 = use this character set
0000:         15 CFONT2  EQU  0          ;0 = don't use this character set
0000:         16 CFONT3  EQU  0          ;only 1 character set can be selected at a time
0000:         17 ;
0000:         18 LOC0    EQU  0          ;used for bootstrap
0001:         19 LOC1    EQU  1
0020:         20 WNDLFT  EQU  $20        ;text window left
0021:         21 WNDWDTH EQU  $21        ;  "    "    width
0022:         22 WNDTOP  EQU  $22        ;  "    "    top
0023:         23 WNDBTM  EQU  $23        ;  "    "    bottom+1
0024:         24 CH      EQU  $24        ;cursor horizontal
0025:         25 CV      EQU  $25        ;cursor verticle
0026:         26 GBASL   EQU  $26        ;left end of LoRes line
0027:         27 GBASH   EQU  $27
0028:         28 BASL    EQU  $28        ;left end of TEXT line
0029:         29 BASH    EQU  $29
002A:         30 BAS2L   EQU  $2A        ;destination line address during scroll
002B:         31 BAS2H   EQU  $2B
002C:         32 H2      EQU  $2C        ;right endpoint for HLIN PLOT
002C:         33 LMNEM   EQU  $2C        ;lo byte table index for dissassm
002D:         34 V2      EQU  $2D
002D:         35 RMNEM   EQU  $2D        ;table index for dissassm
002E:         36 MASK    EQU  $2E        ;for plot
002E:         37 FORMAT  EQU  $2E        ;dism instruction format code
002F:         38 LASTIN  EQU  $2F        ;tape in work area for RDBIT
002F:         39 LENGTH  EQU  $2F        ;dism instruction length
0030:         40 COLOR   EQU  $30        ;LoRes color
0031:         41 MODE    EQU  $31        ;parsing for command processor
0032:         42 INVFLG  EQU  $32        ;FF=norm, 3F=inverse, 7F=flash
0033:         43 PROMPT  EQU  $33        ;prompt char
0034:         44 YSAV    EQU  $34        ;save Y during command for line scanning (input)
0035:         45 YSAV1   EQU  $35        ;save Y during screenout by COUT1
0036:         46 CSWL    EQU  $36        ;output vector
0037:         47 CSWH    EQU  $37
0038:         48 KSWL    EQU  $38        ;input vector
0039:         49 KSWH    EQU  $39
003A:         50 PCL     EQU  $3A        ;go-, list-command, program counter save
003B:         51 PCH     EQU  $3B
003C:         52 A1L     EQU  $3C        ;monitor workspace
003D:         53 A1H     EQU  $3D
003E:         54 A2L     EQU  $3E
003F:         55 A2H     EQU  $3F
0040:         56 A3L     EQU  $40        ;memory set
```

```
0041:            57 A3H      EQU  $41
0042:            58 A4L      EQU  $42
0043:            59 A4H      EQU  $43
0045:            60 ACC      EQU  $45        ;6502 register save area
0046:            61 XREG     EQU  ACC+1      ;  "      "      "    "
0047:            62 YREG     EQU  ACC+2      ;  "      "      "    "
0048:            63 STATUS   EQU  ACC+3      ;  "      "      "    "
0049:            64 SPNT     EQU  ACC+4      ;  "      "      "    "
004E:            65 RNDL     EQU  $4E        ; random number
004F:            66 RNDH     EQU  $4F
0080:            67 BIT7     EQU  $80        ;mask
00C0:            68 IOPAGE   EQU  $C0
0200:            69 IN       EQU  $200       ;keyboard buffer (page)
03F0:            70 BRKV     EQU  $3F0       ;BRK vector (addr)
03F2:            71 SOFTEV   EQU  $3F2       ;soft RESET vector (addr)
03F4:            72 PWREDUP  EQU  $3F4       ;reboot if not (3F3) XOR $A5
03F5:            73 AMPERV   EQU  $3F5       ;Applesoft & vector (instr)
03F8:            74 USRADR   EQU  $3F8       ;U-command Ctrl-Y vector (instr)
03FB:            75 NMI      EQU  $3FB       ;NMI vector (instr)
03FE:            76 IRQLOC   EQU  $3FE       ;IRQ vecto (addr)
0400:            77 LINE1    EQU  $400       ;first screen line, LoRes pg1
07F8:            78 MSLOT    EQU  $7F8       ;active slot ID ($Cn)
C000:            79 IOARD    EQU  $C000
C000:            80 KBD      EQU  $C000      ;ASCII input
C000:            81 CHRINV   EQU  $C000      ;invers/flash switch
C002:            82 CHRGEN0  EQU  $C002      ;char gen A10
C004:            83 CHRGEN1  EQU  $C004      ;char gen A11
C006:            84 CHRBAS   EQU  $C006      ;64+64+128 set (inverse, flash, normal)
C008:            85 KBDEXTN  EQU  $C008      ;functions key input
C00A:            86 VID40    EQU  $C00A      ;40/80 col switch
C00B:            87 VID80    EQU  $C00B
C00C:            88 VIDBNK   EQU  $C00C      ;video RAM switch C00D=off
C010:            89 KBDSTRB  EQU  $C010      ;for alternate 80 columns
C030:            90 SPKR     EQU  $C030
C050:            91 TXTCLR   EQU  $C050      ;video mode swithes
C052:            92 MIXCLR   EQU  $C052      ;"
C054:            93 LOWSCR   EQU  $C054      ;"
C056:            94 LORES    EQU  $C056      ;"
C058:            95 TTLOUT0  EQU  $C058      ;even:off,low <=0.4V
C05A:            96 TTLOUT1  EQU  $C05A      ; odd:on,high >=2.4V
C05C:            97 TTLOUT2  EQU  $C05C
C05E:            98 TTLOUT3  EQU  $C05E
C064:            99 PADDL0   EQU  $C064
C070:           100 PTRIG    EQU  $C070
CFFF:           101 CLRROM   EQU  $CFFF
E000:           102 BASIC    EQU  $E000      ;cold start address for Basic
E003:           103 BASIC2   EQU  $E003      ;warm start address
0050:           104 WIDTH    EQU  80
0479:           105 CHY      EQU  $479       ;80 column video driver
04F9:           106 SWITCH   EQU  $4F9       ;=0 for 40 column
0000:           107 ;                        <> 0 for 80 column
0000:           108 ****************************************
----- NEXT OBJECT FILE NAME IS MONITOR 120
F800:           109          ORG  $F800
F800:           110 ****************************************
```

```
F800:           111 *
F800:           112 * PLOT A POINT
F800:           113 *
F800:           114 ****************************************
F800:4A         115 PLOT    LSR  A          ;Y coord LSB to carry
F801:08         116         PHP             ;save
F802:20 47 F8   117         JSR  GBASCALC   ;set addr in GBASL,H
F805:28         118         PLP             ;retrieve LSB
F806:A9 0F      119         LDA  #$F        ;mask=$F if even
F808:90 02      120         BCC  PL1
F80A:69 E0      121         ADC  #$E0       ;OR $F0 if odd
F80C:85 2E      122 PL1     STA  MASK
F80E:08         123 PLOT1   PHP              ;save LSB
F80F:20 C9 FC   124         JSR  SELBNK     ;set bank
F812:4C ED FC   125         JMP  PLOT80     ;plot point
F815:00 00 00   126         DFB  0,0,0,0    ;fillers
F818:00
F819:           127 ****************************************
F819:           128 *
F819:           129 * DRAW HORIZONTAL LINE
F819:           130 *
F819:           131 ****************************************
F819:20 00 F8   132 HLINE   JSR  PLOT       ;Basic HLINE,plot a point
F81C:C4 2C      133 HL1     CPY  H2         ;end of line?
F81E:B0 11      134         BCS  RTS1       ;yes return
F820:C8         135         INY             ;no, point to next point in pair
F821:20 0E F8   136         JSR  PLOT1      ;plot it
F824:90 F6      137         BCC  HL1        ;do it again (always)
F826:           138 ****************************************
F826:           139 *
F826:           140 * DRAW VERTICAL LINE
F826:           141 *
F826:           142 ****************************************
F826:69 01      143 VLINEZ  ADC  #1         ;increment Y coord
F828:48         144 VLINE   PHA             ;Basic VLINE,save it
F829:20 00 F8   145         JSR  PLOT       ;plot point
F82C:68         146         PLA             ;retrieve next
F82D:C5 2D      147         CMP  V2         ;end of line ?
F82F:90 F5      148         BCC  VLINEZ     ;no, do it again
F831:60         149 RTS1    RTS             ;yes,return
F832:           150 ****************************************
F832:           151 *
F832:           152 * CLEAR SCREEN
F832:           153 *
F832:           154 ****************************************
F832:A0 2F      155 CLRSCR  LDY  #$2F       ; Y-max,full screen clear
F834:D0 02      156         BNE  CLRSC2     ;always
F836:A0 27      157 CLRTOP  LDY  #$27       ;Y-max,mixed clear
F838:20 EE FF   158 CLRSC2  JSR  CLRSC3     ;get screen width
F83B:EA         159         NOP
F83C:A9 00      160 CL1     LDA  #0         ;starting point
F83E:85 30      161         STA  COLOR      ;set black
F840:20 28 F8   162         JSR  VLINE      ;draw verticle
F843:88         163         DEY             ;reduce count
F844:10 F6      164         BPL  CL1        ;if not done
```

```
F846:60         165         RTS
F847:           166 ****************************************
F847:           167 *
F847:           168 * TRANSLATE LINE # TO BASE ADDRESS
F847:           169 *
F847:           170 ****************************************
F847:48         171 GBASCALC PHA             ;line # bits = 000ABCDE
F848:4A         172         LSR  A
F849:29 03      173         AND  #3
F84B:09 04      174         ORA  #4         ;for LoRes Page 1
F84D:85 27      175         STA  GBASH      ;GBASH=000001CD
F84F:68         176         PLA
F850:29 18      177         AND  #$18
F852:90 02      178         BCC  GB1
F854:09 80      179         ORA  #$80
F856:85 26      180 GB1     STA  GBASL
F858:0A         181         ASL  A
F859:0A         182         ASL  A
F85A:05 26      183         ORA  GBASL
F85C:85 26      184         STA  GBASL      ;GBASL=EABAB000
F85E:60         185         RTS
F85F:           186 ****************************************
F85F:           187 *
F85F:           188 * INCREMENT COLOR
F85F:           189 *
F85F:           190 ****************************************
F85F:A5 30      191 NXTCOL  LDA  COLOR      ;color=color+3
F861:18         192         CLC
F862:69 03      193         ADC  #3
F864:           194 ****************************************
F864:           195 *
F864:           196 * SET COLOR
F864:           197 *
F864:           198 ****************************************
F864:29 0F      199 SETCOL  AND  #$F        ;COLOR=17*A mod 16
F866:85 30      200         STA  COLOR
F868:0A         201         ASL  A          ;both nibbles = color
F869:0A         202         ASL  A
F86A:0A         203         ASL  A
F86B:0A         204         ASL  A
F86C:05 30      205         ORA  COLOR
F86E:85 30      206         STA  COLOR
F870:60         207         RTS
F871:           208 ****************************************
F871:           209 *
F871:           210 * READ SCREEN POINT
F871:           211 *
F871:           212 ****************************************
F871:4A         213 SCRN    LSR  A          ;Y coord LSB into carry
F872:08         214         PHP             ;save on stack
F873:20 C2 FE   215         JSR  SCRN80     ;get point "content"
F876:EA         216         NOP              ;filler
F877:EA         217         NOP
F878:28         218         PLP             ;retrieve LSB
F879:90 04      219 SCRN2   BCC  SCR1       ;use low nibble
```

```
F87B:4A          220          LSR  A          ;or move high nibble down
F87C:4A          221          LSR  A
F87D:4A          222          LSR  A
F87E:4A          223          LSR  A
F87F:29 0F       224 SCR1     AND  #$F        ;drop the top
F881:60          225          RTS
F882:            226 ****************************************
F882:            227 *
F882:            228 * DISASSEMBLER DECODER/ERROR CHECK
F882:            229 *
F882:            230 ****************************************
F882:A6 3A       231 INSDS1   LDX  PCL        ;set pc
F884:A4 3B       232          LDY  PCH
F886:20 96 FD    233          JSR  PRYX2      ;display it
F889:20 48 F9    234          JSR  PRBLNK     ;print a blank
F88C:A1 3A       235 INSDS2   LDA  (PCL,X)    ;set opcode
F88E:A8          236          TAY             ;save in Y
F88F:4A          237          LSR  A
F890:90 09       238          BCC  IEVEN      ;if even, go ahead
F892:6A          239          ROR  A
F893:B0 10       240          BCS  ERR        ;all xxxxxx11 opcodes are illegal
F895:C9 A2       241          CMP  #$A2       ;no STA # operation
F897:F0 0C       242          BEQ  ERR
F899:29 87       243          AND  #$87       ;mask
F89B:4A          244 IEVEN    LSR  A          ;LSB into carry for nybble select
F89C:AA          245          TAX
F89D:BD 62 F9    246          LDA  FMT1,X     ;get format
F8A0:20 79 F8    247          JSR  SCRN2      ;select hi or lo nybble
F8A3:D0 04       248          BNE  GETFMT
F8A5:A0 80       249 ERR      LDY  #$80       ;substitute $80 if opcode invalid
F8A7:A9 00       250          LDA  #0         ;format index=0
F8A9:AA          251 GETFMT   TAX
F8AA:BD A6 F9    252          LDA  FMT2,X     ;get output format
F8AD:85 2E       253          STA  FORMAT
F8AF:29 03       254          AND  #3         ;extract length 1=2 bytes, 2=3 bytes
F8B1:85 2F       255          STA  LENGTH
F8B3:98          256          TYA              ;get opcode back
F8B4:29 8F       257          AND  #$8F       ;mask
F8B6:AA          258          TAX             ;and save
F8B7:98          259          TYA             ;get it  again
F8B8:A0 03       260          LDY  #3         ;calculate index into mnemonic table
F8BA:E0 8A       261          CPX  #$8A       ;"
F8BC:F0 0B       262          BEQ  MNNDX3     ;"
F8BE:4A          263 MNNDX1   LSR  A          ;"
F8BF:90 08       264          BCC  MNNDX3     ;"
F8C1:4A          265          LSR  A          ;"
F8C2:4A          266 MNNDX2   LSR  A          ;"
F8C3:09 20       267          ORA  #$20       ;"
F8C5:88          268          DEY             ;"
F8C6:D0 FA       269          BNE  MNNDX2     ;"
F8C8:C8          270          INY             ;"
F8C9:88          271 MNNDX3   DEY             ;"
F8CA:D0 F2       272          BNE  MNNDX1     ;"
F8CC:60          273          RTS
F8CD:00 00 00    274          DFB  0,0,0
```

```
F8D0:           275 ****************************************
F8D0:           276 *
F8D0:           277 * DISASSEMBLER OUTPUT
F8D0:           278 *
F8D0:           279 ****************************************
F8D0:20 82 F8   280 INSTDSP JSR  INSDS1    ;get FMT, LEN, INDEX, output PC
F8D3:48         281         PHA            ;save index
F8D4:B1 3A      282 PRNTOP  LDA  (PCL),Y   ;output instruction (1-3 byte hex)
F8D6:20 DA FD   283         JSR  PRBYTE
F8D9:A2 01      284         LDX  #1        ;interbyte spacing=2 blanks
F8DB:20 4A F9   285 PRNTBL  JSR  PRBL2
F8DE:C4 2F      286         CPY  LENGTH    ;whole instruction output?
F8E0:C8         287         INY
F8E1:90 F1      288         BCC  PRNTOP    ;no, output next byte
F8E3:A2 03      289         LDX  #3        ;3 char mnemonics
F8E5:C0 04      290         CPY  #4        ;align columns
F8E7:90 F2      291         BCC  PRNTBL
F8E9:68         292         PLA            ;retrieve index
F8EA:A8         293         TAY
F8EB:B9 C0 F9   294         LDA  MNEML,Y   ;retreive mnemonic (3 chars in 2 bytes)
F8EE:85 2C      295         STA  LMNEM
F8F0:B9 00 FA   296         LDA  MNEMR,Y
F8F3:85 2D      297         STA  RMNEM
F8F5:A9 00      298 PR1     LDA  #0
F8F7:A0 05      299         LDY  #5        ;shift 5 BITs in ACC
F8F9:06 2D      300 PR2     ASL  RMNEM
F8FB:26 2C      301         ROL  LMNEM
F8FD:2A         302         ROL  A
F8FE:88         303         DEY
F8FF:D0 F8      304         BNE  PR2
F901:69 BF      305         ADC  #$BF      ;offset for ?
F903:20 ED FD   306         JSR  COUT      ;output a character
F906:CA         307         DEX            ;count it
F907:D0 EC      308         BNE  PR1       ;do next char if not done
F909:20 48 F9   309         JSR  PRBLNK    ;output 3 blanks when done
F90C:A4 2F      310         LDY  LENGTH
F90E:A2 06      311         LDX  #6        ;6 format bits
F910:E0 03      312 PRADR1  CPX  #3        ;3 byte instructins
F912:F0 1C      313         BEQ  PRADR5    ;contans addresses
F914:06 2E      314 PRADR2  ASL  FORMAT
F916:90 0E      315         BCC  PRADR3
F918:BD B3 F9   316         LDA  CHAR1-1,X
F91B:20 ED FD   317         JSR  COUT
F91E:BD B9 F9   318         LDA  CHAR2-1,X
F921:F0 03      319         BEQ  PRADR3    ;no 2nd char
F923:20 ED FD   320         JSR  COUT
F926:CA         321 PRADR3  DEX
F927:D0 E7      322         BNE  PRADR1    ;next format BIT
F929:60         323         RTS
F92A:88         324 PRADR4  DEY
F92B:30 E7      325         BMI  PRADR2
F92D:20 DA FD   326         JSR  PRBYTE
F930:A5 2E      327 PRADR5  LDA  FORMAT
F932:C9 E8      328         CMP  #$E8      ;relative addressing
F934:B1 3A      329         LDA  (PCL),Y
```

```
F936:90 F2     330          BCC  PRADR4
F938:20 56 F9  331 RELADR   JSR  PCADJ3    ;yes, output destination instead of offset
F93B:AA        332          TAX
F93C:E8        333          INX
F93D:D0 01     334          BNE  PRNTYX
F93F:C8        335          INY
F940:98        336 PRNTYX   TYA
F941:20 DA FD  337 PRNTAX   JSR  PRBYTE
F944:8A        338 PRNTX    TXA
F945:4C DA FD  339          JMP  PRBYTE
F948:A2 03     340 PRBLNK   LDX  #3         ;output 3 blanks
F94A:A9 A0     341 PRBL2    LDA  #$A0
F94C:20 ED FD  342 PRBL3    JSR  COUT
F94F:CA        343          DEX
F950:D0 F8     344          BNE  PRBL2
F952:60        345          RTS
F953:38        346 PCADJ    SEC             ;adjust PC to next instruction
F954:A5 2F     347 PCADJ2   LDA  LENGTH     ;length = instr len-1
F956:A4 3B     348 PCADJ3   LDY  PCH
F958:AA        349          TAX
F959:10 01     350          BPL  PCADJ4     ;check for backward branch
F95B:88        351          DEY             ;if so decrement
F95C:65 3A     352 PCADJ4   ADC  PCL
F95E:90 01     353          BCC  RTS2
F960:C8        354          INY
F961:60        355 RTS2     RTS
F962:          356 ****************************************
F962:          357 *
F962:          358 * FMT1: 128 (DEC) 4-BIT pointer to the FMT2 table for
F962:          359 * xxxx xxx0 opcodes
F962:          360 * 16 (DEC) 4-BIT pointer to the FMT2 table for
F962:          361 * xxxx xx01 opcodes
F962:          362 *
F962:          363 * if Y=0 use high nybble
F962:          364 * if Y=1 use low nybble
F962:          365 * X=index
F962:          366 *
F962:          367 ****************************************
F962:04 20 54  368 FMT1     DFB  $04,$20,$54,$30,$0D,$80,$04,$90
F965:30 0D 80
F968:04 90
F96A:03 22 54  369          DFB  $03,$22,$54,$33,$0D,$80,$04,$90
F96D:33 0D 80
F970:04 90
F972:04 20 54  370          DFB  $04,$20,$54,$33,$0D,$80,$04,$90
F975:33 0D 80
F978:04 90
F97A:04 20 54  371          DFB  $04,$20,$54,$3B,$0D,$80,$04,$90
F97D:3B 0D 80
F980:04 90
F982:00 22 44  372          DFB  $00,$22,$44,$33,$0D,$C8,$44,$00
F985:33 0D C8
F988:44 00
F98A:11 22 44  373          DFB  $11,$22,$44,$33,$0D,$C8,$44,$A9
F98D:33 0D C8
```

```
F990:44 A9
F992:01 22 44  374          DFB  $01,$22,$44,$33,$0D,$80,$04,$90
F995:33 0D 80
F998:04 90
F99A:01 22 44  375          DFB  $01,$22,$44,$33,$0D,$80,$04,$90
F99D:33 0D 80
F9A0:04 90
F9A2:          376 * xxxx xx01 class:
F9A2:26 31 87  377          DFB  $26,$31,$87,$9A ; ORA,AND,EOR,ADC,STA,LDA,CMD,SBC
F9A5:9A
F9A6:          378 ****************************************
F9A6:          379 *
F9A6:          380 * FMT2 BIT 0..1 : instruction length-1
F9A6:          381 * FMT2 BIT 7..2 : if BIT[i] then (print chr1[i-2],chr2[i-2])
F9A6:          382 *
F9A6:          383 ****************************************
F9A6:00        384 FMT2     DFB  $00       ;illegal opcode
F9A7:21        385          DFB  $21       ;immediate
F9A8:81        386          DFB  $81       ;zero page
F9A9:82        387          DFB  $82       ;absolute
F9AA:00        388          DFB  $00       ;register
F9AB:00        389          DFB  $00       ;accumulator
F9AC:59        390          DFB  $59       ;(zero page,X)
F9AD:4D        391          DFB  $4D       ;(zero page),Y
F9AE:91        392          DFB  $91       ;zero page,X
F9AF:92        393          DFB  $92       ;absolute,X
F9B0:86        394          DFB  $86       ;absolute,Y
F9B1:4A        395          DFB  $4A       ;(absolute)
F9B2:85        396          DFB  $85       ;zero page,Y
F9B3:9D        397          DFB  $9D       ;relative
F9B4:          398 ; char1/char2 used by mini assambler
F9B4:AC A9 AC  399 CHAR1    DFB  $AC,$A9,$AC,$A3,$A8,$A4 ; ",),#($"
F9B7:A3 A8 A4
F9BA:D9 00 D8  400 CHAR2    DFB  $D9,$00,$D8,$A4,$A4,$00 ; "Y X$$ "
F9BD:A4 A4 00
F9C0:          401 *
F9C0:          402 * BBBBB000  ;class instructions
F9C0:1C        403 MNEML    DFB  $1C       ; BRK
F9C1:8A        404          DFB  $8A       ; PHP
F9C2:1C        405          DFB  $1C       ; BPL
F9C3:23        406          DFB  $23       ; CLC
F9C4:5D        407          DFB  $5D       ; JSR
F9C5:8B        408          DFB  $8B       ; PLP
F9C6:1B        409          DFB  $1B       ; BMI
F9C7:A1        410          DFB  $A1       ; SEC
F9C8:9D        411          DFB  $9D       ; RTI
F9C9:8A        412          DFB  $8A       ; PHA
F9CA:1D        413          DFB  $1D       ; BVC
F9CB:23        414          DFB  $23       ; CLI
F9CC:9D        415          DFB  $9D       ; RTS
F9CD:8B        416          DFB  $8B       ; PLA
F9CE:1D        417          DFB  $1D       ; BVS
F9CF:A1        418          DFB  $A1       ; SEI
F9D0:00        419          DFB  $00       ; ?
F9D1:29        420          DFB  $29       ; DEY
```

```
F9D2:19          421          DFB  $19        ; BCC
F9D3:AE          422          DFB  $AE        ; TYA
F9D4:69          423          DFB  $69        ; LDY
F9D5:A8          424          DFB  $A8        ; TAY
F9D6:19          425          DFB  $19        ; BCS
F9D7:23          426          DFB  $23        ; CLV
F9D8:24          427          DFB  $24        ; CPY
F9D9:53          428          DFB  $53        ; IBY
F9DA:1B          429          DFB  $1B        ; BNE
F9DB:23          430          DFB  $23        ; CLD
F9DC:24          431          DFB  $24        ; CPX
F9DD:53          432          DFB  $53        ; INX
F9DE:19          433          DFB  $19        ; BEQ
F9DF:A1          434          DFB  $A1        ; SED
F9E0:            435 * IBBBxx100:
F9E0:00          436          DFB  $00        ; ?
F9E1:1A          437          DFB  $1A        ; BIT
F9E2:5B          438          DFB  $5B        ; JMP
F9E3:5B          439          DFB  $5B        ; JMP
F9E4:A5          440          DFB  $A5        ; STY
F9E5:69          441          DFB  $69        ; LDY
F9E6:24          442          DFB  $24        ; CPY
F9E7:24          443          DFB  $24        ; CPX
F9E8:            444 ; 1BBB1010:
F9E8:AE          445          DFB  $AE        ; TXA
F9E9:AE          446          DFB  $AE        ; TXS
F9EA:A8          447          DFB  $A8        ; TAX
F9EB:AD          448          DFB  $AD        ; TSX
F9EC:29          449          DFB  $29        ; DEX
F9ED:00          450          DFB  $00        ; ?
F9EE:7C          451          DFB  $7C        ; NOP
F9EF:00          452          DFB  $00        ; ?
F9F0:            453 * OIIxxx10:
F9F0:15          454          DFB  $15        ; ASL
F9F1:9C          455          DFB  $9C        ; ROL
F9F2:6D          456          DFB  $6D        ; LSR
F9F3:9C          457          DFB  $9C        ; ROR
F9F4:            458 * 1IIx0010, 1IIx0110, 1IIx1110:
F9F4:A5          459          DFB  $A5        ; STX
F9F5:69          460          DFB  $69        ; LDX
F9F6:29          461          DFB  $29        ; DEC
F9F7:53          462          DFB  $53        ; INC
F9F8:            463 * IIIxxx01:
F9F8:84          464          DFB  $84        ; ORA
F9F9:13          465          DFB  $13        ; AND
F9FA:34          466          DFB  $34        ; EOR
F9FB:11          467          DFB  $11        ; ADC
F9FC:A5          468          DFB  $A5        ; STA
F9FD:69          469          DFB  $69        ; LDA
F9FE:23          470          DFB  $23        ; CMP
F9FF:A0          471          DFB  $A0        ; SBC
FA00:D8 62 5A    472 MNEMR    DFB  $D8,$62,$5A,$48,$26,$62,$94,$88
FA03:48 26 62
FA06:94 88
FA08:54 44 C8    473          DFB  $54,$44,$C8,$54,$68,$44,$E8,$94
```

```
FA0B:54 68 44
FA0E:E8 94
FA10:00 B4 08  474          DFB  $00,$B4,$08,$84,$74,$B4,$28,$6E
FA13:84 74 B4
FA16:28 6E
FA18:74 F4 CC  475          DFB  $74,$F4,$CC,$4A,$72,$F2,$A4,$8A
FA1B:4A 72 F2
FA1E:A4 8A
FA20:00 AA A2  476          DFB  $00,$AA,$A2,$A2,$74,$74,$74,$72
FA23:A2 74 74
FA26:74 72
FA28:44 68 B2  477          DFB  $44,$68,$B2,$32,$B2,$00,$22,$00
FA2B:32 B2 00
FA2E:22 00
FA30:1A 1A 26  478          DFB  $1A,$1A,$26,$26
FA33:26
FA34:72 72 88  479          DFB  $72,$72,$88,$C8
FA37:C8
FA38:C4 CA 26  480          DFB  $C4,$CA,$26,$48,$44,$44,$A2,$C8
FA3B:48 44 44
FA3E:A2 C8
FA40:          481 ****************************************
FA40:          482 *
FA40:          483 * INTERNAL INTERRUPT HANDLER
FA40:          484 *
FA40:          485 ****************************************
FA40:85 45     486 IRQ     STA  ACC       ;save accumulator
FA42:68        487         PLA            ;get top of stack
FA43:48        488         PHA            ;put it back
FA44:29 10     489         AND  #$10      ; test break flag, BIT 4
FA46:D0 04     490         BNE  BREAK     ;yes
FA48:6C FE 03  491         JMP  (IRQLOC)  ;no,goto interrupt handler
FA4B:00        492         DFB  0
FA4C:          493 ****************************************
FA4C:          494 *
FA4C:          495 * INTERNAL BREAK HANDLER
FA4C:          496 *
FA4C:          497 ****************************************
FA4C:28        498 BREAK   PLP            ;pull status
FA4D:20 4C FF  499         JSR  SAV1      ;save registers
FA50:68        500         PLA            ;save PC
FA51:85 3A     501         STA  PCL
FA53:68        502         PLA
FA54:85 3B     503         STA  PCH
FA56:6C F0 03  504         JMP  (BRKV)    ;goto breakhandler
FA59:          505 ****************************************
FA59:          506 *
FA59:          507 * MONITOR BREAK HANDLER
FA59:          508 *
FA59:          509 ****************************************
FA59:20 82 F8  510 OLDBRK  JSR  INSDS1    ;output PC
FA5C:20 DA FA  511         JSR  RGDSP1    ;and registers
FA5F:4C 65 FF  512         JMP  MON       ;enter monitor
FA62:          513 ****************************************
FA62:          514 *
```

```
FA62:           515 * RESET HANDLER
FA62:           516 *
FA62:           517 ****************************************
FA62:D8         518 RESET   CLD
FA63:20 84 FE   519         JSR  SETNORM
FA66:20 2F FB   520         JSR  INIT
FA69:20 93 FE   521         JSR  SETVID     ;console output
FA6C:20 89 FE   522         JSR  SETKBD     ;console input
FA6F:D8         523 NEWMON  CLD             ;not decimal mode
FA70:20 3A FF   524         JSR  BELL
0001:           525         DO   CFONT1     ;ASCII 46-4
FA73:8D 03 C0   526         STA  CHRGEN0+1  ;turn on char set
FA76:8D 04 C0   527         STA  CHRGEN1
FA79:           528         FIN
FA79:           529 *
0000:           530         DO   CFONT2     ;NATIONAL 46-4
FA79         S  531         STA  CHRGEN0    ;turn on char set
FA79         S  532         STA  CHRGEN1+1
FA79:           533         FIN
FA79:           534 *
0000:           535         DO   CFONT3     ;APL 46-4
FA79         S  536         STA  CHRGEN0+1  ;turn on char set
FA79         S  537         STA  CHRGEN1+1
FA79:           538         FIN
FA79:           539 *
FA79:8D 00 C0   540         STA  CHRINV
FA7C:2C FF CF   541         BIT  CLRROM     ;disable extension ROM
FA7F:2C 10 C0   542         BIT  KBDSTRB    ;clear keyboard strobe
FA82:AD F3 03   543         LDA  SOFTEV+1   ;reset high?
FA85:49 A5      544         EOR  #$A5
FA87:CD F4 03   545         CMP  PWREDUP
FA8A:D0 17      546         BNE  PWRUP      ;do,it's the first time
FA8C:AD F2 03   547         LDA  SOFTEV     ;is it cold start
FA8F:D0 0F      548         BNE  NOFIX      ;no,use reset vector
FA91:A9 E0      549         LDA  #$E0       ;is it basic vector
FA93:CD F3 03   550         CMP  SOFTEV+1
FA96:D0 08      551         BNE  NOFIX      ;no,use reset vector
FA98:A0 03      552 FIXSEV  LDY  #3         ;yes,point to warm start
FA9A:8C F2 03   553         STY  SOFTEV
FA9D:4C 00 E0   554         JMP  BASIC      ;and do cold start
FAA0:6C F2 03   555 NOFIX   JMP  (SOFTEV)   ;goto reset vector
FAA3:           556 ****************************************
FAA3:           557 *
FAA3:           558 * POWER ON START
FAA3:           559 *
FAA3:           560 ****************************************
FAA3:38         561 PWRUP   SEC             ;set 80 column mode
FAA4:6E F9 04   562         ROR  SWITCH
FAA7:20 0F FD   563         JSR  LOG01      ;finish setting up video
FAAA:A2 05      564 SETPG3  LDX  #5         ;set vector for Basic
FAAC:BD FC FA   565 SETPLP  LDA  PWRCON-1,X
FAAF:9D EF 03   566         STA  BRKV-1,X
FAB2:CA         567         DEX
FAB3:D0 F7      568         BNE  SETPLP
FAB5:A9 C8      569         LDA  #$C8
```

```
FAB7:85 01      570          STA  LOC1       ;set PTR H to highest slot+1
FAB9:86 00      571          STX  LOC0       ;PTRL=0
FABB:A0 07      572 SLOOP    LDY  #7         ;Y is byte offset into the slot ROM
FABD:C6 01      573          DEC  LOC1       ;start scanning the slots
FABF:A5 01      574          LDA  LOC1
FAC1:C9 C1      575          CMP  #$C1       ;slot=1?
FAC3:F0 D3      576          BEQ  FIXSEV     ;yes,there isn't any disk
FAC5:8D F8 07   577          STA  MSLOT
FAC8:B1 00      578 SL1      LDA  (LOC0),Y   ;try to read slot ROM
FACA:D9 01 FB   579          CMP  DISKID,Y   ;is it a boot device (floppy, harddisk...) ??
FACD:D0 EC      580          BNE  SLOOP      ;no, test next slot
FACF:88         581          DEY
FAD0:88         582          DEY             ;yes so check next odd byte
FAD1:10 F5      583          BPL  SL1
FAD3:6C 00 00   584          JMP  (LOC0)     ;it is a disk! jump to ROM
FAD6:00         585          DFB  0
FAD7:           586 ****************************************
FAD7:           587 *
FAD7:           588 * DISPLAY REGISTER CONTENTS
FAD7:           589 *
FAD7:           590 ****************************************
FAD7:20 8E FD   591 REGDSP   JSR  CROUT      ;new line
FADA:A9 45      592 RGDSP1   LDA  #ACC       ;get pointer to save area
FADC:85 40      593          STA  A3L
FADE:A9 00      594          LDA  #0
FAE0:85 41      595          STA  A3H
FAE2:A2 FB      596          LDX  #$FB       ;register count
FAE4:A9 A0      597 RG1      LDA  #$A0       ;output a blank
FAE6:20 ED FD   598          JSR  COUT
FAE9:BD 1E FA   599          LDA  RTBL-251,X;get register label
FAEC:20 ED FD   600          JSR  COUT
FAEF:A9 BD      601          LDA  #$BD       ;output=sign
FAF1:20 ED FD   602          JSR  COUT
FAF4:B5 4A      603          LDA  ACC+5,X    ;get register contents>
FAF6:20 DA FD   604          JSR  PRBYTE     ;print it
FAF9:E8         605          INX             ;index to next
FAFA:30 E8      606          BMI  RG1        ;repeat until done
FAFC:60         607          RTS             ;then return
FAFD:           608 ****************************************
FAFD:59 FA      609 PWRCON   DW   OLDBRK
FAFF:00 E0      610          DW   BASIC
FB01:45 20      611 DISKID   EOR  $20         ;opcode (0E0^0A5=45) used for mask!!
FB03:A0 00      612          LDY  #0         ;code never executed,
FB05:A2 03      613          LDX  #3         ;only for disk ID
FB07:86 3C      614          STX  $3C
FB09:08 15 0A   615 LOCCHR   DFB  $08,$15,$0A,$0B,$40,$0E,$0F ; special characters
FB0C:0B 40 0E
FB0F:0F
FB10:D0 01      616 SW1      BNE  SW2        ;if not 0
FB12:0A         617          ASL  A          ;first multiply by 2
FB13:8D 79 04   618 SW2      STA  CHY        ;set index (always)
FB16:4C 09 FF   619          JMP  SCR180     ;scroll
FB19:C1 D8 D9   620 RTBL     DFB  $C1,$D8,$D9,$D0,$D3 ;register labels "AXYPS"
FB1C:D0 D3
FB1E:           621 ****************************************
```

```
FB1E:           622 *
FB1E:           623 * BASIC PDL(n) FUNCTION
FB1E:           624 *
FB1E:           625 ****************************************
FB1E:AD 70 C0   626 PREAD   LDA  PTRIG     ;Trigger read
FB21:A0 00      627         LDY  #0        ;initiate count
FB23:EA         628         NOP            ;timing compensation
FB24:EA         629         NOP
FB25:BD 64 C0   630 PRE1    LDA  PADDL0,X  ;test every 12 usec
FB28:10 04      631         BPL  RTSX      ;exit if read complete
FB2A:C8         632         INY            ;or count loop in Y
FB2B:D0 F8      633         BNE  PRE1      ;and do it again (max 255)
FB2D:88         634         DEY            ;back up to 255 if full count
FB2E:60         635 RTSX    RTS
FB2F:           636 ****************************************
FB2F:           637 *
FB2F:           638 * VIDEO INITIALIZATION
FB2F:           639 *
FB2F:           640 ****************************************
FB2F:A9 04      641 INIT    LDA  #4        ;set I flag!
FB31:85 48      642         STA  STATUS
FB33:2C 56 C0   643         BIT  LORES     ;set screen switches
FB36:2C 54 C0   644         BIT  LOWSCR
FB39:           645 ****************************************
FB39:           646 *
FB39:           647 * SET TEXT MODE, BASIC TEXT
FB39:           648 *
FB39:           649 ****************************************
FB39:2C 51 C0   650 SETTXT  BIT  TXTCLR+1  ;text switch
FB3C:A9 00      651         LDA  #0        ;always skip
FB3E:F0 0B      652         BEQ  SETWND    ;to SETWND
FB40:           653 ****************************************
FB40:           654 *
FB40:           655 * SET GRAPHICS, BASIC GR
FB40:           656 *
FB40:           657 ****************************************
FB40:2C 50 C0   658 SETGR   BIT  TXTCLR
FB43:2C 53 C0   659         BIT  MIXCLR+1  ;set mixed mode
FB46:20 36 F8   660         JSR  CLRTOP    ;clear graphics area
FB49:A9 14      661         LDA  #20       ;top of text window
FB4B:           662 ****************************************
FB4B:           663 *
FB4B:           664 * ESTABLISH WINDOW
FB4B:           665 *
FB4B:           666 ****************************************
FB4B:85 22      667 SETWND  STA  WNDTOP
FB4D:A9 00      668         LDA  #0        ;full width
FB4F:85 20      669         STA  WNDLFT
FB51:A9 50      670         LDA  #WIDTH
FB53:85 21      671         STA  WNDWDTH
FB55:A9 18      672         LDA  #24       ;to last line
FB57:85 23      673         STA  WNDBTM
FB59:A9 17      674         LDA  #23       ;set vertical tab to 23
FB5B:85 25      675 TABV    STA  CV
FB5D:4C 22 FC   676         JMP  VTAB
```

```
FB60:           677 ****************************************
FB60:           678 *
FB60:           679 * PUT "Basis 108" ON TOP OF SCREEN
FB60:           680 *
FB60:           681 ****************************************
FB60:20 58 FC   682 LOGO    JSR  HOME       ;clear the screen
FB63:A0 08      683         LDY  #8         ;title length
FB65:B9 90 FB   684 LO1     LDA  TITLE,Y    ;get a char
FB68:99 00 04   685         STA  LINE1,Y    ;put on the screen
FB6B:88         686         DEY             ;count it
FB6C:10 F7      687         BPL  LO1        ;and continue till done
FB6E:60         688         RTS
FB6F:           689 ****************************************
FB6F:           690 *
FB6F:           691 * SET PAGE 3 POWER UP BYTE
FB6F:           692 *
FB6F:           693 ****************************************
FB6F:AD F3 03   694 SETPWRC LDA  SOFTEV+1
FB72:49 A5      695         EOR  #$A5
FB74:8D F4 03   696         STA  PWREDUP
FB77:60         697         RTS
FB78:           698 ****************************************
FB78:           699 *
FB78:           700 * CHECK KEYBOARD BEFORE VIDEO OUTPUT
FB78:           701 *
FB78:           702 ****************************************
FB78:AC 00 C0   703 VIDWAIT LDY  KBD
FB7B:C0 93      704         CPY  #$93       ;ctrl-S pressed?
FB7D:D0 0F      705         BNE  VI2        ;no, continue
FB7F:2C 10 C0   706         BIT  KBDSTRB    ;clear keyboard strobe
FB82:AC 00 C0   707 VI1     LDY  KBD        ;wait for next keypress
FB85:10 FB      708         BPL  VI1
FB87:C0 83      709         CPY  #$83       ;ctrl-C?
FB89:F0 72      710         BEQ  VIDOUT     ;yes, it is for Basic
FB8B:8D 10 C0   711         STA  KBDSTRB    ;no, clear strobe
FB8E:D0 6D      712 VI2     BNE  VIDOUT     ;display char in accu
FB90:           713 ****************************************
FB90:C2 E1 F3   714 TITLE   DFB  $C2,$E1,$F3,$E9,$F3,$A0,$B1,$B0,$B8 ;Basis 108
FB93:E9 F3 A0
FB96:B1 B0 B8
FB99:0F 3E 65   715 LOCJMP  DFB  $F,$3E,$65,$19,$57,$9B,$41
FB9C:19 57 9B
FB9F:41
FBA0:           716 ****************************************
FBA0:           717 *
FBA0:           718 * AUDIBLE KEYPRESS FEEDBACK TESTS
FBA0:           719 *
FBA0:           720 ****************************************
FBA0:A0 07      721 LOCAL   LDY  #7         ;# of codes in list
FBA2:D9 09 FB   722 LOCA1   CMP  LOCCHR,Y   ;test against each
FBA5:D0 0B      723         BNE  LOCA2      ;loop if not it
FBA7:A9 FC      724         LDA  #$FC       ;if it is, put -4 on stack
FBA9:48         725         PHA
FBAA:B9 99 FB   726         LDA  LOCJMP,Y   ;pickup matching item from LOCJMP table
FBAD:48         727         PHA             ;stack it
```

```
FBAE:A0 18     728          LDY  #$18
FBB0:D0 2D     729          BNE  PIP        ;audible feedback for legel keys
FBB2:88        730 LOCA2    DEY
FBB3:10 ED     731          BPL  LOCA1      ;until last one checked
FBB5:60        732          RTS             ;exit if not on list
FBB6:20 A0 FB  733 JLOCAL   JSR  LOCAL      ;vector
FBB9:20 0C FD  734 RDCHAR1  JSR  RDKEY      ;get keyboard input
FBBC:29 FF     735          AND  #$FF       ;test BIT 7
FBBE:10 F6     736          BPL  JLOCAL     ;if special key
FBC0:60        737          RTS             ;else return
FBC1:          738 ****************************************
FBC1:          739 *
FBC1:          740 * CALCULATE BASE ADDRESS FOR LINE ON SCREEN
FBC1:          741 *
FBC1:          742 ****************************************
FBC1:48        743 BASCALC PHA             ;save line # in form 000ABCDE
FBC2:4A        744          LSR  A
FBC3:29 03     745          AND  #3
FBC5:09 04     746          ORA  #4         ;for text page 1
FBC7:85 29     747          STA  BASH       ;BASH=000001CD
FBC9:68        748          PLA
FBCA:29 18     749          AND  #$18
FBCC:90 02     750          BCC  BA1
FBCE:09 80     751          ORA  #$80
FBD0:85 28     752 BA1      STA  BASL
FBD2:0A        753          ASL  A
FBD3:0A        754          ASL  A
FBD4:05 28     755          ORA  BASL
FBD6:85 28     756          STA  BASL       ;BASL=EABAB000
FBD8:60        757          RTS
FBD9:          758 ****************************************
FBD9:          759 *
FBD9:          760 * 2 TONE BELL
FBD9:          761 *
FBD9:          762 ****************************************
FBD9:C9 87     763 BELL1    CMP  #$87       ;Ctrl-G?
FBDB:D0 10     764          BNE  NOCTRL     ;no, exit
FBDD:A0 70     765          LDY  #$70       ;new sound
FBDF:98        766 PIP      TYA             ;another sound
FBE0:4A        767          LSR  A
FBE1:4A        768          LSR  A
FBE2:09 07     769          ORA  #7         ;set minimum time
FBE4:20 A8 FC  770          JSR  WAIT       ;wait a little while
FBE7:2C 30 C0  771          BIT  SPKR       ;tick speaker
FBEA:88        772          DEY
FBEB:D0 F2     773          BNE  PIP        ;loop until sound over
FBED:60        774 NOCTRL   RTS
FBEE:          775 ****************************************
FBEE:          776 *
FBEE:          777 * PUT CHARACTER IN SCREEN RAM
FBEE:          778 *
FBEE:          779 ****************************************
FBEE:25 32     780 STORINV AND  INVFLG     ;set inverse bit
FBF0:20 CE FE  781 STORADV JSR  STOR80     ;do 80 column output
FBF3:EA        782          NOP
```

```
FBF4:                 783 ****************************************
FBF4:                 784 *
FBF4:                 785 * MOVE TO NEXT SCREEN POSITION
FBF4:                 786 *
FBF4:                 787 ****************************************
FBF4:E6 24            788 ADVANCE INC  CH         ;inc horiz index
FBF6:A4 24            789         LDY  CH
FBF8:C4 21            790         CPY  WNDWDTH    ;wrap at window end
FBFA:B0 66            791         BCS  CR         ;to next line
FBFC:60               792         RTS
FBFD:                 793 ****************************************
FBFD:                 794 *
FBFD:                 795 * TEXT OUTPUT TO VIDEO
FBFD:                 796 *
FBFD:                 797 ****************************************
FBFD:C9 A0            798 VIDOUT  CMP  #$A0       ;ctrl?
FBFF:B0 ED            799         BCS  STORINV    ;no, diPLAy it normal
FC01:A8               800         TAY             ;or
FC02:10 EC            801         BPL  STORADV    ;normal
FC04:C9 8D            802         CMP  #$8D       ;Return?
FC06:F0 5A            803         BEQ  CR         ;yes, do it
FC08:C9 8A            804         CMP  #$8A       ;linefeed?
FC0A:F0 5A            805         BEQ  LF         ;yes,do it
FC0C:C9 88            806         CMP  #$88       ;Ctrl-H?
FC0E:D0 C9            807         BNE  BELL1      ;no, check for Ctrl-G
FC10:                 808 ****************************************
FC10:                 809 *
FC10:                 810 * MOVE TO PREVIOUS SCREEN POSITION
FC10:                 811 *
FC10:                 812 ****************************************
FC10:C6 24            813 BS      DEC  CH         ;yes, decrement Horiz index
FC12:10 16            814         BPL  RTS4       ;if in window,leave
FC14:A5 21            815         LDA  WNDWDTH    ;else go to window right edge
FC16:85 24            816         STA  CH
FC18:C6 24            817         DEC  CH         ;minus one
FC1A:A5 22            818 UP      LDA  WNDTOP     ;if at top line
FC1C:C5 25            819         CMP  CV
FC1E:B0 0A            820         BCS  RTS4       ;exit
FC20:C6 25            821         DEC  CV         ;else move up a line
FC22:                 822 ****************************************
FC22:                 823 *
FC22:                 824 * CALCULATE VTAB SCREEN ADDRESS
FC22:                 825 *
FC22:                 826 ****************************************
FC22:A5 25            827 VTAB    LDA  CV
FC24:20 C1 FB         828 VTABZ   JSR  BASCALC    ;calculate new base address
FC27:4C EB FE         829         JMP  VTAB80     ;set vert pos for 80 column
FC2A:60               830 RTS4    RTS
FC2B:                 831 ****************************************
FC2B:                 832 *
FC2B:                 833 * GET A CHARACTER IN UPPER CASE
FC2B:                 834 *
FC2B:                 835 ****************************************
FC2B:B9 00 02         836 GETUPCS LDA  IN,Y       ;read char from input buffer
FC2E:C8               837         INY             ;point to next char
```

```
FC2F:C9 E0     838 UPPER   CMP  #$E0      ;max uppercase ASCII
FC31:90 02     839         BCC  UP1       ;exit if already uppercase
FC33:29 DF     840         AND  #$DF      ;else shift to uppercase
FC35:60        841 UP1     RTS
FC36:48        842 SW5     PHA            ;save A on stack
FC37:98        843 SW6     TYA            ;multiply index
FC38:4A        844         LSR  A         ;by 2
FC39:8D 0B C0  845         STA  VID80     ;set 80 column display
FC3C:4C CC FC  846         JMP  SELBNK2   ;continue bankswitch
FC3F:4C F4 FB  847         JMP  ADVANCE   ;cursor advance vector
FC42:          848 ****************************************
FC42:          849 *
FC42:          850 * CLEAR TO END OF PAGE
FC42:          851 *
FC42:          852 ****************************************
FC42:A4 24     853 CLREOP  LDY  CH        ;get horiz
FC44:A5 25     854         LDA  CV        ;and vert indices
FC46:48        855 CLEOP1  PHA            ;save vert
FC47:20 24 FC  856         JSR  VTABZ     ;get base addr for current line
FC4A:20 9E FC  857         JSR  CLEOLZ    ;clear to end of line
FC4D:A0 00     858         LDY  #0        ;start at left end of subsequent lines
FC4F:68        859         PLA            ;retrieve vert index
FC50:69 00     860         ADC  #0        ;and inc, carry=1 from cleolz
FC52:C5 23     861         CMP  WNDBTM    ;off windowbottom?
FC54:90 F0     862         BCC  CLEOP1    ;if not,do nextline
FC56:B0 CA     863         BCS  VTAB      ;if so set vert index to it
FC58:          864 ****************************************
FC58:          865 *
FC58:          866 * BASIC "HOME"
FC58:          867 *
FC58:          868 ****************************************
FC58:A5 22     869 HOME    LDA  WNDTOP    ;set vert index to windowtop
FC5A:85 25     870         STA  CV
FC5C:A0 00     871         LDY  #0        ;set horiz index to 0
FC5E:84 24     872         STY  CH
FC60:F0 E4     873         BEQ  CLEOP1    ;clear to end of screen (always)
FC62:          874 ****************************************
FC62:          875 *
FC62:          876 * VIDEO CARRIAGE RETURN
FC62:          877 *
FC62:          878 ****************************************
FC62:A9 00     879 CR      LDA  #0        ;set horiz index to 0
FC64:85 24     880         STA  CH
FC66:E6 25     881 LF      INC  CV        ;increment vert index
FC68:A5 25     882         LDA  CV
FC6A:C5 23     883         CMP  WNDBTM    ;off bottom of window?
FC6C:90 B6     884         BCC  VTABZ     ;no, calc new base addr
FC6E:C6 25     885         DEC  CV        ;yes, back up one
FC70:          886 ****************************************
FC70:          887 *
FC70:          888 * SCROLL TEXT SCREEN
FC70:          889 *
FC70:          890 ****************************************
FC70:A5 22     891 SCROLL  LDA  WNDTOP    ;get top linenumber
FC72:48        892         PHA            ;save it
```

```
FC73:20 24 FC  893          JSR  VTABZ      ;get base addr
FC76:A5 28     894 SC1      LDA  BASL       ;copy it
FC78:85 2A     895          STA  BAS2L
FC7A:A5 29     896          LDA  BASH
FC7C:85 2B     897          STA  BAS2H
FC7E:A4 21     898          LDY  WNDWDTH    ;point to last char in line
FC80:88        899          DEY
FC81:68        900          PLA             ;increment linenumber
FC82:69 01     901          ADC  #1         ;carry=0 from line feed
FC84:C5 23     902          CMP  WNDBTM     ;off bottom?
FC86:B0 0D     903          BCS  SC3        ;yes,finnish
FC88:48        904          PHA             ;no,save it linenumber for next time
FC89:20 24 FC  905          JSR  VTABZ      ;calc base addr
FC8C:          906 *
FC8C:98        907          TYA             ;get index into A
FC8D:AC F9 04  908          LDY  SWITCH     ;and 40/80 SWITCH status in Y
FC90:20 10 FB  909          JSR  SW1        ;on return carry 0, scroll the line
FC93:90 E1     910          BCC  SC1        ;do next line (always)
FC95:A0 00     911 SC3      LDY  #0         ;point to start of line
FC97:20 9E FC  912          JSR  CLEOLZ     ;clear it
FC9A:B0 86     913          BCS  VTAB       ;set (always)
FC9C:          914 ****************************************
FC9C:          915 *
FC9C:          916 * CLEAR TO END OF LINE
FC9C:          917 *
FC9C:          918 ****************************************
FC9C:A4 24     919 CLREOL   LDY  CH
FC9E:          920 *
FC9E:38        921 CLEOLZ   SEC             ;carry=1 after PLP
FC9F:08        922          PHP
FCA0:4C DB FC  923          JMP  CLEOL80
FCA3:00 00 00  924          DFB  0,0,0,0,0 ;fillers
FCA6:00 00
FCA8:          925          CHN  MON120P2
FCA8:            1 ****************************************
FCA8:            2 *
FCA8:            3 * TIME DELAY
FCA8:            4 *
FCA8:            5 ****************************************
FCA8:38          6 WAIT     SEC             ;wait for ord(Accu^2) time
FCA9:48          7 WA1      PHA
FCAA:E9 01       8 WA2      SBC  #1
FCAC:D0 FC       9          BNE  WA2
FCAE:68         10          PLA
FCAF:E9 01      11          SBC  #1
FCB1:D0 F6      12          BNE  WA1
FCB3:60         13          RTS
FCB4:           14 ****************************************
FCB4:           15 *
FCB4:           16 * 16 BIT INCREMENT OF A4
FCB4:           17 *
FCB4:           18 ****************************************
FCB4:E6 42      19 NXTA4    INC  A4L        ;increment low byte
FCB6:D0 02      20          BNE  NXTA1      ;if no overflow, skip ahead
FCB8:E6 43      21          INC  A4H        ;if overflow, increment hi byte
```

```
FCBA:A5 3C      22 NXTA1   LDA  A1L       ;compare A1 to A2 (16 bits)
FCBC:C5 3E      23         CMP  A2L
FCBE:A5 3D      24         LDA  A1H
FCC0:E5 3F      25         SBC  A2H       ;carry set if A1 equals or exceeds A2
FCC2:E6 3C      26         INC  A1L       ;increment A1 (16 bits)
FCC4:D0 02      27         BNE  NX1
FCC6:E6 3D      28         INC  A1H
FCC8:60         29 NX1     RTS
FCC9:           30 ****************************************
FCC9:           31 *
FCC9:           32 * 80 COLUMN SCREEN DRIVER
FCC9:           33 *
FCC9:           34 * BANKSWITCH ON ODD/EVEN CHARACTER POSITION
FCC9:           35 *
FCC9:           36 ****************************************
FCC9:4C FA FC   37 SELBNK  JMP  SW3
FCCC:8D 0C C0   38 SELBNK2 STA  VIDBNK    ;400..BFF: dynamic RAM (default)
FCCF:90 04      39         BCC  SE1       ;skip if Y even
FCD1:78         40         SEI            ;if Y odd, enable interrupts
FCD2:8D 0D C0   41         STA  VIDBNK+1  ;400..BFF: static RAM
FCD5:8C 79 04   42 SE1     STY  CHY       ;save Yreg in active bank!
FCD8:A8         43         TAY            ;for horiz index in active bank
FCD9:68         44         PLA            ;retrieve A
FCDA:60         45         RTS
FCDB:           46 ****************************************
FCDB:           47 *
FCDB:           48 * CLEAR TO END OF LINE
FCDB:           49 *
FCDB:           50 ****************************************
FCDB:20 C9 FC   51 CLEOL80 JSR  SELBNK    ;select bank for this char
FCDE:A9 A0      52         LDA  #$A0      ;a blank
FCE0:91 28      53         STA  (BASL),Y  ;clear this char
FCE2:AC 79 04   54         LDY  CHY       ;last char done?
FCE5:C8         55         INY
FCE6:C4 21      56         CPY  WNDWDTH
FCE8:90 F1      57         BCC  CLEOL80   ;no, do another
FCEA:4C E6 FE   58         JMP  VIDPLP    ;yes,restore status
FCED:           59 ****************************************
FCED:           60 *
FCED:           61 * PLOT A POINT
FCED:           62 *
FCED:           63 ****************************************
FCED:B1 26      64 PLOT80  LDA  (GBASL),Y ;get current point value
FCEF:45 30      65         EOR  COLOR     ;set color
FCF1:25 2E      66         AND  MASK      ;mask off
FCF3:51 26      67         EOR  (GBASL),Y ;put it back
FCF5:91 26      68         STA  (GBASL),Y
FCF7:4C E3 FE   69         JMP  VIDRTS    ;restore status
FCFA:           70 ****************************************
FCFA:48         71 SW3     PHA            ;save A on stack
FCFB:AD F9 04   72         LDA  SWITCH    ;get state of 40/80 column switch
FCFE:F0 03      73         BEQ  SW4       ;if not 0,
FD00:4C 37 FC   74         JMP  SW6       ;do bank select
FD03:68         75 SW4     PLA            ;recover A
FD04:8C 79 04   76         STY  CHY       ;set index
```

```
FD07:8D 0A C0   77          STA  VID40      ;set 40 column display
FD0A:60         78          RTS
FD0B:00         79          DFB  0          ;filler
FD0C:4C 15 FD   80 RDKEY    JMP  RDKEY2     ;vector
FD0F:20 2F FB   81 LOG01    JSR  INIT       ;initialize video status
FD12:4C 60 FB   82          JMP  LOGO       ;display logo
FD15:20 D7 FE   83 RDKEY2   JSR  CURS80     ;and cursor
FD18:6C 38 00   84          JMP  (KSWL)     ;go to User routine
FD1B:           85 ****************************************
FD1B:           86 *
FD1B:           87 * READ KEYBOARD
FD1B:           88 *
FD1B:           89 ****************************************
FD1B:E6 4E      90 KEYIN    INC  RNDL
FD1D:D0 02      91          BNE  KE1        ;cycle random number
FD1F:E6 4F      92          INC  RNDH
FD21:2C 00 C0   93 KE1      BIT  KBD        ;key pressed?
FD24:10 F5      94          BPL  KEYIN      ;no, wait for one
FD26:20 D7 FE   95          JSR  CURS80     ;remove cursor
FD29:AD 08 C0   96          LDA  KBDEXTN    ;read function key BIT
FD2C:29 80      97          AND  #BIT7
FD2E:4D 00 C0   98          EOR  KBD        ;merge with ASCII code
FD31:8D 10 C0   99          STA  KBDSTRB    ;save it
FD34:60        100          RTS
FD35:          101 ****************************************
FD35:          102 *
FD35:          103 * READ INPUT LINE
FD35:          104 *
FD35:          105 ****************************************
FD35:4C B9 FB  106 RDCHAR   JMP  RDCHAR1    ;vector
FD38:00 00 00  107          DFB  0,0,0,0,0  ;fillers
FD3B:00 00
FD3D:A5 32     108 NOTCR    LDA  INVFLG     ;get current INVFLG status
FD3F:48        109          PHA             ;save it
FD40:A9 FF     110          LDA  #$FF       ;set INVFLG off
FD42:85 32     111          STA  INVFLG
FD44:BD 00 02  112          LDA  IN,X       ;get char from input buffer
FD47:20 ED FD  113          JSR  COUT       ;output in normal
FD4A:68        114          PLA             ;restore INVFLG
FD4B:85 32     115          STA  INVFLG
FD4D:BD 00 02  116          LDA  IN,X       ;get char again
FD50:C9 88     117          CMP  #$88       ;ctrl-H (backspace)?
FD52:F0 1D     118          BEQ  BCKSPC     ;yes, back up index
FD54:C9 98     119          CMP  #$98       ;ctrl-X?
FD56:F0 0A     120          BEQ  CANCEL     ;yes, cancel input line
FD58:E0 F8     121          CPX  #$F8       ;end of buffer?
FD5A:90 03     122          BCC  NOTCR1     ;no, skip ahead
FD5C:20 3A FF  123          JSR  BELL       ;yes, ring bell
FD5F:E8        124 NOTCR1   INX             ;point to next char
FD60:D0 13     125          BNE  NXTCHAR    ;get it if not too many
FD62:A9 A3     126 CANCEL   LDA  #$A3       ;or flag cancelled input on screen
FD64:20 ED FD  127          JSR  COUT
FD67:20 8E FD  128 GETLNZ   JSR  CROUT      ;output Return
FD6A:A5 33     129 GETLN    LDA  PROMPT     ;output prompt
FD6C:20 ED FD  130          JSR  COUT
```

```
FD6F:A2 01     131          LDX  #1         ;point to start of buffer
FD71:8A        132 BCKSPC   TXA             ;if backspaced to 0,
FD72:F0 F3     133          BEQ  GETLNZ     ;start new input line
FD74:CA        134          DEX             ;else back up one char
FD75:20 35 FD  135 NXTCHAR  JSR  RDCHAR     ;get a char from keyboard
FD78:C9 95     136          CMP  #$95       ;ctrl-U (right arrow)?
FD7A:D0 08     137          BNE  ADDINP     ;no, skip ahead
FD7C:20 FE FE  138          JSR  GET80      ;get input char from screen
FD7F:EA        139          NOP             ;filler
FD80:EA        140          NOP
FD81:EA        141          NOP
FD82:EA        142          NOP
FD83:EA        143          NOP
FD84:9D 00 02  144 ADDINP   STA  IN,X       ;put in input buffer
FD87:C9 8D     145          CMP  #$8D       ;Return? (end of input)
FD89:D0 B2     146          BNE  NOTCR      ;no, get aother
FD8B:20 9C FC  147          JSR  CLREOL     ;clear rest of line
FD8E:A9 8D     148 CROUT    LDA  #$8D       ;echo Return
FD90:D0 5B     149          BNE  COUT       ;and exit via COUT
FD92:          150 ****************************************
FD92:          151 *
FD92:          152 * OUTPUT (A1)
FD92:          153 *
FD92:          154 ****************************************
FD92:A4 3D     155 PRA1     LDY  A1H        ;get A1 contents (16 bit)
FD94:A6 3C     156          LDX  A1L
FD96:20 FA FD  157 PRYX2    JSR  NEWLN      ;start new output line
FD99:20 40 F9  158          JSR  PRNTYX     ;output contents
FD9C:A0 00     159          LDY  #0         ;index to linestart
FD9E:A9 BA     160          LDA  #$BA       ;output a ":"
FDA0:4C ED FD  161          JMP  COUT       ;exit via COUT
FDA3:          162 ****************************************
FDA3:          163 *
FDA3:          164 * HEX MEMORY DUMP
FDA3:          165 *
FDA3:          166 ****************************************
FDA3:A5 3C     167 XAM8     LDA  A1L        ;set starting addr lo byte
FDA5:09 0F     168          ORA  #$F        ;end at mod8 addr
FDA7:85 3E     169          STA  A2L        ;set ending addr in A2
FDA9:A5 3D     170          LDA  A1H
FDAB:85 3F     171          STA  A2H
FDAD:A5 3C     172 MOD8CHK  LDA  A1L        ;retrieve starting addr (lo byte)
FDAF:29 0F     173          AND  #$F        ;strip hi bit
FDB1:D0 03     174          BNE  DATAOUT    ;if lo=0 (each 8 or 16 bytes)
FDB3:20 92 FD  175 XAM      JSR  PRA1       ;output addr 0
FDB6:A9 A0     176 DATAOUT  LDA  #$A0       ;output a blank
FDB8:20 ED FD  177          JSR  COUT
FDBB:B1 3C     178          LDA  (A1L),Y    ;get  byte
FDBD:20 DA FD  179          JSR  PRBYTE     ;and output it
FDC0:20 BA FC  180          JSR  NXTA1      ;increment pointer and test for end
FDC3:90 E8     181          BCC  MOD8CHK    ;loop if not end
FDC5:60        182          RTS             ;exit if end
FDC6:          183 ****************************************
FDC6:          184 *
FDC6:          185 * MODE CHECK
```

```
FDC6:             186 *
FDC6:             187 ****************************************
FDC6:AD F9 04     188 SW7     LDA  SWITCH    ;check 40/80 column switch
FDC9:F0 04        189         BEQ  SW740     ;if 0, do it for 40 column
FDCB:A5 20        190         LDA  WNDLFT    ;else multiply width
FDCD:4A           191         LSR  A         ;by 2
FDCE:60           192         RTS
FDCF:A9 28        193 SW740   LDA  #40       ;is width < or = to 40
FDD1:C5 21        194         CMP  WNDWDTH
FDD3:B0 02        195         BCS  WDTHOK
FDD5:85 21        196         STA  WNDWDTH   ;if not, clamp it
FDD7:A5 20        197 WDTHOK  LDA  WNDLFT    ;get left edge
FDD9:60           198         RTS
FDDA:48           199 PRBYTE  PHA            ;keeping it on stack
FDDB:4A           200         LSR  A         ;move high nibble down
FDDC:4A           201         LSR  A
FDDD:4A           202         LSR  A
FDDE:4A           203         LSR  A
FDDF:20 E5 FD     204         JSR  PRHEXZ    ;output char
FDE2:68           205         PLA            ;retrieve value again
FDE3:29 0F        206 PRHEX   AND  #$F       ;strip off hi bit
FDE5:09 B0        207 PRHEXZ  ORA  #$B0      ;convert to ASCII
FDE7:C9 BA        208         CMP  #$BA      ;>9?
FDE9:90 02        209         BCC  COUT      ;no, output it
FDEB:69 06        210         ADC  #6        ;"A".."F"
FDED:             211 ****************************************
FDED:             212 *
FDED:             213 * ASCII OUTPUT
FDED:             214 *
FDED:             215 ****************************************
FDED:6C 36 00     216 COUT    JMP  (CSWL)    ;convert to user output routine
FDF0:48           217 COUT1   PHA            ;save A
FDF1:84 35        218         STY  YSAV1     ;and Y
FDF3:20 78 FB     219         JSR  VIDWAIT   ;output it
FDF6:A4 35        220         LDY  YSAV1     ;restore Y
FDF8:68           221         PLA            ;and A
FDF9:60           222         RTS            ;and return
FDFA:             223 ****************************************
FDFA:             224 *
FDFA:             225 * START NEW OUTPUT LINE
FDFA:             226 *
FDFA:             227 ****************************************
FDFA:20 8E FD     228 NEWLN   JSR  CROUT     ;output Return
FDFD:A9 A0        229         LDA  #$A0      ;and blank
FDFF:D0 EC        230         BNE  COUT      ;exit through COUT (always)
FE01:             231 ****************************************
FE01:             232 *
FE01:             233 * MONITOR COMMAND PAGE
FE01:             234 *
FE01:             235 ****************************************
FE01:C6 34        236 BL1     DEC  YSAV      ;if YSAVE=1
FE03:F0 9E        237         BEQ  XAM8      ;do memory dump
FE05:CA           238 BLANK   DEX            ;if X doesn't = 1
FE06:D0 16        239         BNE  SETMDZ    ;set mode
FE08:C9 BA        240         CMP  #$BA      ;if not store mode,
```

```
FE0A:D0 A7     241          BNE  XAM        ;do dump, add, or subtract
FE0C:          242 ****************************************
FE0C:          243 *
FE0C:          244 * MONITOR STORE
FE0C:          245 *
FE0C:          246 ****************************************
FE0C:85 31     247 STOR     STA  MODE       ;set mode
FE0E:A5 3E     248          LDA  A2L        ;get byte
FE10:91 40     249          STA  (A3L),Y    ;store it
FE12:E6 40     250          INC  A3L        ;adjust pointer (16 bits)
FE14:D0 02     251          BNE  ST1
FE16:E6 41     252          INC  A3H
FE18:60        253 ST1      RTS             ;and exit
FE19:A4 34     254 SETMODE  LDY  YSAV       ;retrieve pointer
FE1B:B9 FF 01  255          LDA  IN-1,Y     ;get command (:, +, -, or .)
FE1E:85 31     256 SETMDZ   STA  MODE       ;set mode
FE20:60        257          RTS             ;and exit
FE21:          258 ****************************************
FE21:          259 *
FE21:          260 * COPY A2 TO A4, A5 (16 BITS)
FE21:          261 *
FE21:          262 ****************************************
FE21:A2 01     263 LT       LDX  #1         ;index
FE23:B5 3E     264 LT1      LDA  A2L,X      ;move lo byte
FE25:95 42     265          STA  A4L,X
FE27:CA        266          DEX             ;adjust index
FE28:10 F9     267          BPL  LT1        ;move hi byte
FE2A:60        268          RTS             ;then exit
FE2B:00        269          DFB  0          ;filler
FE2C:          270 ****************************************
FE2C:          271 *
FE2C:          272 * MEMORY BLOCK MOVE
FE2C:          273 *
FE2C:          274 ****************************************
FE2C:B1 3C     275 MOVE     LDA  (A1L),Y    ;get byte
FE2E:91 42     276          STA  (A4L),Y    ;move (copy) it
FE30:20 B4 FC  277          JSR  NXTA4      ;adj source pointer (16 bit) and test for end
FE33:90 F7     278          BCC  MOVE       ;if not end, do again
FE35:60        279          RTS             ;else exit
FE36:          280 ****************************************
FE36:          281 *
FE36:          282 * MEMORY BLOCK COMPARE
FE36:          283 *
FE36:          284 ****************************************
FE36:B1 3C     285 VERIFY   LDA  (A1L),Y    ;get byte from range 1
FE38:D1 42     286          CMP  (A4L),Y    ;compare to matching byte in range 2
FE3A:F0 17     287          BEQ  VE1        ;if match, skip ahead
FE3C:20 92 FD  288          JSR  PRA1       ;else print addr in range 1
FE3F:B1 3C     289          LDA  (A1L),Y    ;retrieve offending byte from range 1
FE41:20 DA FD  290          JSR  PRBYTE     ;output it
FE44:A9 BC     291          LDA  #$BC       ;output <
FE46:20 ED FD  292          JSR  COUT
FE49:A9 BE     293          LDA  #$BE       ;output >
FE4B:20 ED FD  294          JSR  COUT
FE4E:B1 42     295          LDA  (A4L),Y    ;retrieve offending byte from range 2
```

```
FE50:20 DA FD  296          JSR  PRBYTE     ;output it
FE53:20 B4 FC  297 VE1      JSR  NXTA4      ;increment pointer (16 bits) and test for end
FE56:90 DE     298          BCC  VERIFY     ;if not end, do again
FE58:60        299          RTS             ;else exit
FE59:6C F2 03  300 BASCONT  JMP  (SOFTEV)   ;vector for Basic "CONT"
FE5C:4C 00 E0  301 XBASIC   JMP  BASIC      ;vector for Basic with no program
FE5F:00        302          DFB  0          ;filler
FE60:          303 ****************************************
FE60:          304 *
FE60:          305 * MONITOR "LIST"
FE60:          306 *
FE60:          307 ****************************************
FE60:20 75 FE  308 LIST     JSR  A1PC       ;move A1 (16 bits) to PC
FE63:20 D0 F8  309 LI1      JSR  INSTDSP    ;disasm instruction
FE66:20 53 F9  310          JSR  PCADJ      ;adj PC to next instruction
FE69:85 3A     311          STA  PCL
FE6B:84 3B     312          STY  PCH
FE6D:C5 3E     313          CMP  A2L        ;test for list end
FE6F:98        314          TYA
FE70:E5 3F     315          SBC  A2H
FE72:90 EF     316          BCC  LI1        ;if not, do again
FE74:60        317          RTS             ;if so, return
FE75:          318 ****************************************
FE75:          319 *
FE75:          320 * SET PC FROM A1
FE75:          321 *
FE75:          322 ****************************************
FE75:8A        323 A1PC     TXA             ;if X=0
FE76:F0 07     324          BEQ  A1PC2      ;exit
FE78:B5 3C     325 A1PC1    LDA  A1L,X      ;move lo byte
FE7A:95 3A     326          STA  PCL,X
FE7C:CA        327          DEX             ;adjust index
FE7D:10 F9     328          BPL  A1PC1      ;do again for hi byte
FE7F:60        329 A1PC2    RTS             ;then exit
FE80:          330 ****************************************
FE80:          331 *
FE80:          332 * INVERSE
FE80:          333 *
FE80:          334 ****************************************
FE80:A0 7F     335 SETINV   LDY  #$7F       ;true status
FE82:D0 02     336          BNE  SETIFLG    ;(always)
FE84:          337 ****************************************
FE84:          338 *
FE84:          339 * NORMAL
FE84:          340 *
FE84:          341 ****************************************
FE84:A0 FF     342 SETNORM LDY  #$FF       ;false status
FE86:84 32     343 SETIFLG STY  INVFLG     ;set it
FE88:60        344          RTS
FE89:          345 ****************************************
FE89:          346 *
FE89:          347 * SET I/O PORT
FE89:          348 *
FE89:          349 ****************************************
FE89:A9 00     350 SETKBD   LDA  #0         ;default=0 (keyboard)
```

```
FE8B:85 3E     351 INPORT  STA  A2L       ;IN#n entry, save n
FE8D:A2 38     352 INPRT   LDX  #KSWL     ;input vector
FE8F:A0 1B     353         LDY  #$1B      ;keyin lo byte
FE91:D0 08     354         BNE  IOPRT     ;(always)
FE93:A9 00     355 SETVID  LDA  #0        ;default output = 0 (video)
FE95:85 3E     356 OUTPORT STA  A2L       ;PR#n entry, save n
FE97:A2 36     357 OUTPRT  LDX  #CSWL     ;output vector
FE99:A0 F0     358         LDY  #$F0      ;cout1 lo byte
FE9B:A5 3E     359 IOPRT   LDA  A2L       ;retrieve n
FE9D:29 07     360         AND  #7        ;only slots 1..7 are legal
FE9F:F0 06     361         BEQ  IOPRT1    ;slot 0 has no I/O ROM space
FEA1:09 C0     362         ORA  #IOPAGE   ;$C0 to point to appropriate I/O ROM
FEA3:A0 00     363         LDY  #0
FEA5:F0 02     364         BEQ  IOPRT2    ;(always)
FEA7:A9 FD     365 IOPRT1  LDA  #$FD      ;KEYIN/COUT1 hi byte
FEA9:94 00     366 IOPRT2  STY  LOC0,X    ;set vector
FEAB:95 01     367         STA  LOC1,X
FEAD:A5 3E     368         LDA  A2L       ;if slot in [8..15.] then vector =Cs08
FEAF:29 08     369         AND  #8        ;else vector=Cs00
FEB1:15 00     370         ORA  LOC0,X
FEB3:95 00     371         STA  LOC0,X
FEB5:60        372         RTS
FEB6:          373 ****************************************
FEB6:20 75 FE  374 GO      JSR  A1PC      ;set PC from A1
FEB9:20 3F FF  375         JSR  RESTORE   ;restore registers
FEBC:6C 3A 00  376         JMP  (PCL)     ;go where told to
FEBF:          377 ****************************************
FEBF:4C D7 FA  378 REGZ    JMP  REGDSP    ;register display vectors
FEC2:          379 ****************************************
FEC2:20 47 F8  380 SCRN80  JSR  GBASCALC  ;calculate base address
FEC5:4C 2E FF  381         JMP  SCRN802
FEC8:00 00     382         DFB  0,0       ;filler
FECA:          383 ****************************************
FECA:4C F8 03  384 USR     JMP  USRADR    ;user vector
FECD:60        385 WRITE   RTS            ;no tape out
FECE:          386 ****************************************
FECE:          387 *
FECE:          388 * PUT CHARACTER IN SCREEN RAM
FECE:          389 *
FECE:          390 ****************************************
FECE:08        391 STOR80  PHP            ;save status
FECF:A4 24     392         LDY  CH        ;get horiz index
FED1:20 C9 FC  393         JSR  SELBNK    ;select appropriate bank
FED4:4C E1 FE  394         JMP  STRTS     ;go store char
FED7:          395 ****************************************
FED7:          396 *
FED7:          397 * BLINK CURSOR
FED7:          398 *
FED7:          399 ****************************************
FED7:08        400 CURS80  PHP            ;save status
FED8:A4 24     401         LDY  CH        ;get horiz index
FEDA:20 C9 FC  402         JSR  SELBNK    ;select appropriate bank
FEDD:B1 28     403         LDA  (BASL),Y  ;get char
FEDF:49 80     404         EOR  #BIT7     ;invert inverse bit
FEE1:91 28     405 STRTS   STA  (BASL),Y  ;write char,
```

```
FEE3:AC 79 04   406 VIDRTS  LDY  CHY       ;restore Yreg,
FEE6:8D 0C C0   407 VIDPLP  STA  VIDBNK    ;restore memory bank (softswitch)
FEE9:28         408         PLP            ;restore Iflag
FEEA:60         409         RTS            ;exit
FEEB:           410 ****************************************
FEEB:           411 *
FEEB:           412 * CALCULATE VTAB SCREEN ADDRESS
FEEB:           413 *
FEEB:           414 ****************************************
FEEB:20 C6 FD   415 VTAB80  JSR  SW7
FEEE:18         416         CLC
FEEF:65 28      417         ADC  BASL
FEF1:85 28      418         STA  BASL
FEF3:60         419         RTS            ;exit
FEF4:00 00      420         DFB  0,0       ;filler
FEF6:           421 ****************************************
FEF6:           422 *
FEF6:           423 * COMMAND PROCESSOR ENTRY POINT
FEF6:           424 *
FEF6:           425 ****************************************
FEF6:20 01 FE   426 CRMON   JSR  BL1
FEF9:68         427         PLA            ;adjust stac
FEFA:68         428         PLA
FEFB:D0 6C      429         BNE  MONZ      ;enter command processor
FEFD:60         430 READ    RTS            ;no tape input!
FEFE:           431 ****************************************
FEFE:           432 *
FEFE:           433 * SCREEN READ
FEFE:           434 *
FEFE:           435 ****************************************
FEFE:08         436 GET80   PHP             ;save status
FEFF:A4 24      437         LDY  CH        ;get horiz index
FF01:20 C9 FC   438         JSR  SELBNK    ;select appropriate bank
FF04:B1 28      439         LDA  (BASL),Y  ;get char
FF06:4C E3 FE   440         JMP  VIDRTS    ;exit
FF09:           441 ****************************************
FF09:           442 *
FF09:           443 * FAST SCROLL LINE WITHOUT JSR SELBNK
FF09:           444 *
FF09:           445 ****************************************
FF09:08         446 SCR180  PHP             ;save status
FF0A:78         447         SEI             ;disable interrupts (400..BFF is switched!)
FF0B:4A         448         LSR  A         ;LSB to carry
FF0C:A8         449         TAY             ;save A
FF0D:90 0F      450         BCC  EVENCHR   ;first time odd or even?
FF0F:8D 0D C0   451 ODDCHR  STA  VIDBNK+1  ;static RAM on
FF12:B1 28      452         LDA  (BASL),Y  ;copy in static RAM
FF14:91 2A      453         STA  (BAS2L),Y ;up a line
FF16:8D 0C C0   454         STA  VIDBNK    ;static RAM off
FF19:CE 79 04   455         DEC  CHY       ;adjust vert index
FF1C:30 0A      456         BMI  SCRLEX    ;done?
FF1E:B1 28      457 EVENCHR LDA  (BASL),Y  ;copy in dynamic RAM (even chars)
FF20:91 2A      458         STA  (BAS2L),Y ;up a line
FF22:88         459         DEY             ;counter
FF23:CE 79 04   460         DEC  CHY       ;adjust vert index
```

```
FF26:10 E7      461          BPL  ODDCHR     ;more to scroll?
FF28:28         462 SCRLEX   PLP              ;no restore status
FF29:18         463          CLC
FF2A:60         464          RTS              ;and return
FF2B:00 00      465          DFB  0,0         ;filler
FF2D:           466 ****************************************
FF2D:60         467 PRERR    RTS              ;"dummy"
FF2E:           468 ****************************************
FF2E:           469 *
FF2E:           470 * READ SCREEN POINT
FF2E:           471 *
FF2E:           472 ****************************************
FF2E:20 C9 FC   473 SCRN802  JSR  SELBNK      ;set appropriate bank
FF31:B1 26      474          LDA  (GBASL),Y   ;get data
FF33:8D 0C C0   475          STA  VIDBNK      ;restore proper bank
FF36:AC 79 04   476          LDY  CHY         ;retrieve Y index
FF39:60         477          RTS
FF3A:A9 87      478 BELL     LDA  #$87        ;audible signal too
FF3C:4C ED FD   479          JMP  COUT
FF3F:           480 ****************************************
FF3F:           481 *
FF3F:           482 * RESTORE REGISTERS
FF3F:           483 *
FF3F:           484 ****************************************
FF3F:A5 48      485 RESTORE  LDA  STATUS      ;put status
FF41:48         486          PHA              ;on stack
FF42:A5 45      487          LDA  ACC         ;restore A
FF44:A6 46      488 RESTR1   LDX  XREG        ;  "     X
FF46:A4 47      489          LDY  YREG        ;  "     Y
FF48:28         490          PLP              ;  "     status
FF49:60         491          RTS
FF4A:           492 ****************************************
FF4A:           493 *
FF4A:           494 * SAVE REGISTERS
FF4A:           495 *
FF4A:           496 ****************************************
FF4A:85 45      497 SAVE     STA  ACC         ;save A
FF4C:86 46      498 SAV1     STX  XREG        ;  "  X
FF4E:84 47      499          STY  YREG        ;  "  Y
FF50:08         500          PHP              ;  "   status to stack
FF51:68         501          PLA              ;then to
FF52:85 48      502          STA  STATUS      ;memory
FF54:BA         503          TSX
FF55:86 49      504          STX  SPNT        ;save the old stack pointer value!
FF57:D8         505          CLD              ;clear decimal value
FF58:60         506 IORTS    RTS              ;used by slot ROM
FF59:           507 ****************************************
FF59:           508 *
FF59:           509 * RESET MONITOR
FF59:           510 *
FF59:           511 ****************************************
FF59:20 84 FE   512 OLDRST   JSR  SETNORM     ;normal video
FF5C:20 2F FB   513          JSR  INIT        ;set video mode and IFLG
FF5F:20 93 FE   514          JSR  SETVID      ;set default output
FF62:20 89 FE   515          JSR  SETKBD      ;and input
```

```
FF65:D8          516 MON     CLD             ;clear decimal mode
FF66:20 3A FF    517         JSR  BELL       ;tell someone
FF69:            518 ****************************************
FF69:            519 *
FF69:            520 * MONITOR COMMAND PROCESSOR
FF69:            521 *
FF69:            522 ****************************************
FF69:A9 AA       523 MONZ    LDA  #$AA       ;monitor prompt (*)
FF6B:85 33       524         STA  PROMPT
FF6D:20 67 FD    525         JSR  GETLNZ     ;get command input line
FF70:20 C7 FF    526         JSR  ZMODE      ;set mode 0
FF73:20 A7 FF    527 NXTITM  JSR  GETNUM     ;get non-hex chars
FF76:84 34       528         STY  YSAV       ;save place in input line
FF78:A0 11       529         LDY  #$11       ;command table length
FF7A:88          530 CHRSRCH DEY             ;adjust table index
FF7B:30 E8       531         BMI  MON        ;if not found, reenter monitor
FF7D:D9 CC FF    532         CMP  CHRTBL,Y   ;check table entry against command
FF80:D0 F8       533         BNE  CHRSRCH    ;loop until found or end of table
FF82:20 BE FF    534         JSR  TOSUB      ;go to desired routine
FF85:A4 34       535         LDY  YSAV       ;on return, restore Y
FF87:4C 73 FF    536         JMP  NXTITM     ;interrupt next command
FF8A:            537 ****************************************
FF8A:            538 *
FF8A:            539 * PUT HEX IN A2
FF8A:            540 *
FF8A:            541 ****************************************
FF8A:A2 03       542 DIG     LDX  #3
FF8C:0A          543         ASL  A          ;rotate digit up
FF8D:0A          544         ASL  A
FF8E:0A          545         ASL  A
FF8F:0A          546         ASL  A
FF90:0A          547 NXTBIT  ASL  A
FF91:26 3E       548         ROL  A2L        ;corresponding A2 rotation
FF93:26 3F       549         ROL  A2H
FF95:CA          550         DEX
FF96:10 F8       551         BPL  NXTBIT     ;until transferred
FF98:A5 31       552 NXTBAS  LDA  MODE       ;if mode not 0
FF9A:D0 06       553         BNE  NXTBS2     ;skip ahead
FF9C:B5 3F       554         LDA  A2H,X      ;else copy A2H
FF9E:95 3D       555         STA  A1H,X      ;into a1H
FFA0:95 41       556         STA  A3H,X      ;and A3H
FFA2:E8          557 NXTBS2  INX             ;index up
FFA3:F0 F3       558         BEQ  NXTBAS     ;until all copied
FFA5:D0 06       559         BNE  NXTCHR     ;then return to parser
FFA7:            560 ****************************************
FFA7:            561 *
FFA7:            562 * COMMAND LINE PARSER
FFA7:            563 *
FFA7:            564 ****************************************
FFA7:A2 00       565 GETNUM  LDX  #0         ;initialize index
FFA9:86 3E       566         STX  A2L        ;and A2
FFAB:86 3F       567         STX  A2H
FFAD:20 2B FC    568 NXTCHR  JSR  GETUPCS    ;get uppercase char
FFB0:49 B0       569         EOR  #$B0       ;check if hex digit
FFB2:C9 0A       570         CMP  #$A
```

```
FFB4:90 D4      571          BCC  DIG        ;if so, handle it
FFB6:69 88      572          ADC  #$88
FFB8:C9 FA      573          CMP  #$FA
FFBA:B0 CE      574          BCS  DIG        ;and next
FFBC:60         575          RTS             ;return with non-decimal char in A
FFBD:00         576          DFB  0
FFBE:           577 ************************************
FFBE:           578 *
FFBE:           579 * ROUTE TO MONITOR ROUTINES
FFBE:           580 *
FFBE:           581 ************************************
FFBE:A9 FE      582 TOSUB    LDA  #$FE       ;command page for high byte
FFC0:48         583          PHA             ;push on stack
FFC1:B9 DD FF   584          LDA  SUBTBL,Y   ;set lo byte
FFC4:48         585          PHA             ;push it too
FFC5:A5 31      586          LDA  MODE       ;get mode in A
FFC7:A0 00      587 ZMODE    LDY  #0         ;clear mode byte
FFC9:84 31      588          STY  MODE
FFCB:60         589          RTS             ;jump to desired routine (from stack)
FFCC:           590 ************************************
FFCC:           591 *
FFCC:           592 * COMMAND CHARACTERS
FFCC:           593 *
FFCC:           594 ************************************
FFCC:EA         595 CHRTBL   DFB  $EA        ;Q warm start Basic
FFCD:BB         596          DFB  $BB        ;CTRL-B cold start Basic
FFCE:EE         597          DFB  $EE        ;U user vector jump
FFCF:98         598          DFB  $98        ;? display registers
FFD0:EF         599          DFB  $EF        ;V verify memory
FFD1:06         600          DFB  $06        ;M move memory
FFD2:04         601          DFB  $04        ;K input vector
FFD3:E9         602          DFB  $E9        ;P output vector
FFD4:07         603          DFB  $07        ;N normal
FFD5:02         604          DFB  $02        ;I inverse
FFD6:05         605          DFB  $05        ;L list memory (disassm)
FFD7:00         606          DFB  $00        ;G jump to address
FFD8:93         607          DFB  $93        ;: memory store
FFD9:A7         608          DFB  $A7        ;. delimiter
FFDA:95         609          DFB  $95        ;< transfer operator
FFDB:C6         610          DFB  $C6        ;CTRL-M carriage return
FFDC:99         611          DFB  $99        ;BLANK space
FFDD:           612 ************************************
FFDD:           613 *
FFDD:           614 * COMMAND POINTERS
FFDD:           615 *
FFDD:           616 ************************************
FFDD:58         617 SUBTBL   DFB  $58        ;Basic warm JMP $3F2 is moved
FFDE:5B         618          DFB  $5B        ;Basic cold JMP $E000 is moved
FFDF:C9         619          DFB  $C9        ;user JMP $3F8
FFE0:BE         620          DFB  $BE        ;register display
FFE1:35         621          DFB  $35        ;verify
FFE2:2B         622          DFB  $2B        ;move
FFE3:8C         623          DFB  $8C        ;input vector
FFE4:96         624          DFB  $96        ;output vector
FFE5:83         625          DFB  $83        ;normal
```

```
FFE6:7F          626          DFB  $7F        ;inverse
FFE7:5F          627          DFB  $5F        ;list is moved!
FFE8:B5          628          DFB  $B5        ;go
FFE9:18          629          DFB  $18        ;memory store
FFEA:18          630          DFB  $18        ;range separator
FFEB:20          631          DFB  $20        ;transfer, compare
FFEC:F5          632          DFB  $F5        ;<cr>
FFED:04          633          DFB  $04        ;<space>
FFEE:            634 ****************************************
FFEE:            635 *
FFEE:            636 * GET SCREEN WIDTH
FFEE:            637 *
FFEE:            638 ****************************************
FFEE:84 2D       639 CLRSC3   STY  V2         ;save bottom line #
FFF0:A0 4F       640          LDY  #79        ;80 col-1 (default)
FFF2:AD F9 04    641          LDA  SWITCH     ;check switch
FFF5:D0 02       642          BNE  CLR80      ;if not 80 column,
FFF7:A0 27       643          LDY  #39        ;must be 40 col-1
FFF9:60          644 CLR80    RTS
FFFA:FB 03       645          DW   NMI
FFFC:62 FA       646          DW   RESET
FFFE:40 FA       647          DW   IRQ

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

SYMBOL TABLE SORTED BY SYMBOL

3D A1H	3C A1L	FE75 A1PC	FE78 A1PC1
FE7F A1PC2	3F A2H	3E A2L	41 A3H
40 A3L	43 A4H	42 A4L	45 ACC
FD84 ADDINP	FBF4 ADVANCE	?03F5 AMPERV	FBD0 BA1
2B BAS2H	2A BAS2L	FBC1 BASCALC	?FE59 BASCONT
29 BASH	?E003 BASIC2	E000 BASIC	28 BASL
FD71 BCKSPC	FF3A BELL	FBD9 BELL1	80 BIT7
FE01 BL1	?FE05 BLANK	FA4C BREAK	03F0 BRKV
?FC10 BS	FD62 CANCEL	01 CFONT1	00 CFONT2
00 CFONT3	F9B4 CHAR1	F9BA CHAR2	?C006 CHRBAS
C004 CHRGEN1	FF7A CHRSRCH	24 CH	C002 CHRGEN0
C000 CHRINV	FFCC CHRTBL	0479 CHY	F83C CL1
FCDB CLEOL80	FC9E CLEOLZ	FC46 CLEOP1	FFF9 CLR80
FC9C CLREOL	?FC42 CLREOP	CFFF CLRROM	F838 CLRSC2
FFEE CLRSC3	?F832 CLRSCR	F836 CLRTOP	30 COLOR
FDED COUT	?FDF0 COUT1	FC62 CR	?FEF6 CRMON
FD8E CROUT	? 37 CSWH	36 CSWL	FED7 CURS80
25 CV	FDB6 DATAOUT	FF8A DIG	FB01 DISKID
F8A5 ERR	FF1E EVENCHR	FA98 FIXSEV	F962 FMT1
F9A6 FMT2	2E FORMAT	F856 GB1	F847 GBASCALC
27 GBASH	26 GBASL	FEFE GET80	F8A9 GETFMT
?FD6A GETLN	FD67 GETLNZ	FFA7 GETNUM	FC2B GETUPCS
?FEB6 GO	2C H2	F81C HL1	?F819 HLINE
FC58 HOME	F89B IEVEN	?FE8B INPORT	0200 IN
FB2F INIT	?FE8D INPRT	F882 INSDS1	?F88C INSDS2
F8D0 INSTDSP	32 INVFLG	?C000 IOARD	C0 IOPAGE
FE9B IOPRT	FEA7 IOPRT1	FEA9 IOPRT2	?FF58 IORTS
FA40 IRQ	03FE IRQLOC	FBB6 JLOCAL	C008 KBDEXTN
C010 KBDSTRB	C000 KBD	FD21 KE1	FD1B KEYIN
? 39 KSWH	38 KSWL	? 2F LASTIN	2F LENGTH
FC66 LF	FE63 LI1	0400 LINE1	?FE60 LIST
2C LMNEM	FB65 LO1	00 LOC0	01 LOC1
FBA2 LOCA1	FBB2 LOCA2	FBA0 LOCAL	FB09 LOCCHR
FB99 LOCJMP	FD0F LOGO1	FB60 LOGO	C056 LORES
C054 LOWSCR	FE23 LT1	?FE21 LT	2E MASK
C052 MIXCLR	F9C0 MNEML	FA00 MNEMR	F8BE MNNDX1
F8C2 MNNDX2	F8C9 MNNDX3	FDAD MOD8CHK	31 MODE
FF69 MONZ	FF65 MON	FE2C MOVE	07F8 MSLOT
FDFA NEWLN	?FA6F NEWMON	03FB NMI	FBED NOCTRL
FAA0 NOFIX	FD5F NOTCR1	FD3D NOTCR	FCC8 NX1
FCBA NXTA1	FCB4 NXTA4	FF98 NXTBAS	FF90 NXTBIT
FFA2 NXTBS2	FD75 NXTCHAR	FFAD NXTCHR	?F85F NXTCOL
FF73 NXTITM	FF0F ODDCHR	FA59 OLDBRK	?FF59 OLDRST
?FE95 OUTPORT	?FE97 OUTPRT	C064 PADDL0	?F954 PCADJ2
F953 PCADJ	F956 PCADJ3	F95C PCADJ4	3B PCH
3A PCL	FBDF PIP	F80C PL1	F80E PLOT1
FCED PLOT80	F800 PLOT	F8F5 PR1	F8F9 PR2
FD92 PRA1	F910 PRADR1	F914 PRADR2	F926 PRADR3
F92A PRADR4	F930 PRADR5	F94A PRBL2	?F94C PRBL3
F948 PRBLNK	FDDA PRBYTE	FB25 PRE1	?FB1E PREAD
?FF2D PRERR	?FDE3 PRHEX	FDE5 PRHEXZ	?F941 PRNTAX
F8DB PRNTBL	F8D4 PRNTOP	?F944 PRNTX	F940 PRNTYX
33 PROMPT	FD96 PRYX2	C070 PTRIG	FAFD PWRCON
03F4 PWREDUP	FAA3 PWRUP	FBB9 RDCHAR1	FD35 RDCHAR
FD0C RDKEY	FD15 RDKEY2	?FEFD READ	FAD7 REGDSP

SYMBOL TABLE SORTED BY SYMBOL

```
?FEBF REGZ          ?F938 RELADR        FA62 RESET          FF3F RESTORE
?FF44 RESTR1         FAE4 RG1           FADA RGDSP1           2D RMNEM
   4F RNDH             4E RNDL          FB19 RTBL           F831 RTS1
 F961 RTS2           FC2A RTS4          FB2E RTSX           FF4C SAV1
?FF4A SAVE           FC76 SC1           FC95 SC3            FF09 SCR180
 F87F SCR1           FF28 SCRLEX        F879 SCRN2          FF2E SCRN802
?F871 SCRN           FEC2 SCRN80       ?FC70 SCROLL         FCD5 SE1
 FCCC SELBNK2        FCC9 SELBNK       ?F864 SETCOL        ?FB40 SETGR
 FE86 SETIFLG       ?FE80 SETINV        FE89 SETKBD         FE1E SETMDZ
?FE19 SETMODE        FE84 SETNORM      ?FAAA SETPG3         FAAC SETPLP
?FB6F SETPWRC      ?FB39 SETTXT        FE93 SETVID         FB4B SETWND
 FAC8 SL1            FABB SLOOP        03F2 SOFTEV         C030 SPKR
   49 SPNT           FE18 ST1            48 STATUS         FECE STOR80
 FBF0 STORADV       ?FE0C STOR         FBEE STORINV         FEE1 STRTS
 FFDD SUBTBL         FB10 SW1          FB13 SW2            FCFA SW3
 FD03 SW4           ?FC36 SW5          FC37 SW6            FDC6 SW7
 FDCF SW740          04F9 SWITCH      ?FB5B TABV           FB90 TITLE
 FFBE TOSUB         ?C058 TTLOUT0     ?C05A TTLOUT1       ?C05C TTLOUT2
?C05E TTLOUT3        C050 TXTCLR       ?FC2F UPPER        ?FC1A UP
 FC35 UP1           ?FECA USR          03F8 USRADR           2D V2
 FE53 VE1            FE36 VERIFY       FB82 VI1            FB8E VI2
 C00A VID40          C00B VID80        C00C VIDBNK         FBFD VIDOUT
 FEE6 VIDPLP         FEE3 VIDRTS       FB78 VIDWAIT        F826 VLINEZ
 F828 VLINE          FEEB VTAB80       FC24 VTABZ          FC22 VTAB
 FCA9 WA1            FCAA WA2          FCA8 WAIT           FDD7 WDTHOK
   50 WIDTH            23 WNDBTM         20 WNDLFT           22 WNDTOP
   21 WNDWDTH       ?FECD WRITE        FDB3 XAM            FDA3 XAM8
?FE5C XBASIC           46 XREG           47 YREG             35 YSAV1
   34 YSAV           FFC7 ZMODE
```

SYMBOL TABLE SORTED BY ADDRESS

```
   00 CFONT3          00 CFONT2          00 LOC0            01 CFONT1
   01 LOC1            20 WNDLFT          21 WNDWDTH         22 WNDTOP
   23 WNDBTM          24 CH              25 CV              26 GBASL
   27 GBASH           28 BASL            29 BASH            2A BAS2L
   2B BAS2H           2C H2              2C LMNEM           2D V2
   2D RMNEM           2E MASK            2E FORMAT        ? 2F LASTIN
   2F LENGTH          30 COLOR           31 MODE            32 INVFLG
   33 PROMPT          34 YSAV            35 YSAV1           36 CSWL
?  37 CSWH            38 KSWL         ?  39 KSWH            3A PCL
   3B PCH             3C A1L             3D A1H             3E A2L
   3F A2H             40 A3L             41 A3H             42 A4L
   43 A4H             45 ACC             46 XREG            47 YREG
   48 STATUS          49 SPNT            4E RNDL            4F RNDH
   50 WIDTH           80 BIT7            C0 IOPAGE        0200 IN
 03F0 BRKV          03F2 SOFTEV        03F4 PWREDUP      ?03F5 AMPERV
 03F8 USRADR        03FB NMI           03FE IRQLOC        0400 LINE1
 0479 CHY           04F9 SWITCH        07F8 MSLOT        ?C000 IOARD
 C000 KBD           C000 CHRINV        C002 CHRGEN0       C004 CHRGEN1
?C006 CHRBAS        C008 KBDEXTN       C00A VID40         C00B VID80
 C00C VIDBNK        C010 KBDSTRB       C030 SPKR          C050 TXTCLR
 C052 MIXCLR        C054 LOWSCR        C056 LORES        ?C058 TTLOUT0
?C05A TTLOUT1     ?C05C TTLOUT2      ?C05E TTLOUT3       C064 PADDL0
 C070 PTRIG         CFFF CLRROM        E000 BASIC        ?E003 BASIC2
 F800 PLOT          F80C PL1           F80E PLOT1        ?F819 HLINE
 F81C HL1           F826 VLINEZ        F828 VLINE         F831 RTS1
?F832 CLRSCR        F836 CLRTOP        F838 CLRSC2        F83C CL1
 F847 GBASCALC      F856 GB1        ?F85F NXTCOL      ?F864 SETCOL
?F871 SCRN          F879 SCRN2         F87F SCR1          F882 INSDS1
?F88C INSDS2        F89B IEVEN         F8A5 ERR           F8A9 GETFMT
 F8BE MNNDX1        F8C2 MNNDX2        F8C9 MNNDX3        F8D0 INSTDSP
 F8D4 PRNTOP        F8DB PRNTBL        F8F5 PR1           F8F9 PR2
 F910 PRADR1        F914 PRADR2        F926 PRADR3        F92A PRADR4
 F930 PRADR5     ?F938 RELADR          F940 PRNTYX     ?F941 PRNTAX
?F944 PRNTX         F948 PRBLNK        F94A PRBL2      ?F94C PRBL3
 F953 PCADJ      ?F954 PCADJ2          F956 PCADJ3        F95C PCADJ4
 F961 RTS2          F962 FMT1          F9A6 FMT2          F9B4 CHAR1
 F9BA CHAR2         F9C0 MNEML         FA00 MNEMR         FA40 IRQ
 FA4C BREAK         FA59 OLDBRK        FA62 RESET      ?FA6F NEWMON
 FA98 FIXSEV        FAA0 NOFIX         FAA3 PWRUP      ?FAAA SETPG3
 FAAC SETPLP        FABB SLOOP         FAC8 SL1           FAD7 REGDSP
 FADA RGDSP1        FAE4 RG1           FAFD PWRCON        FB01 DISKID
 FB09 LOCCHR        FB10 SW1           FB13 SW2           FB19 RTBL
?FB1E PREAD         FB25 PRE1          FB2E RTSX          FB2F INIT
?FB39 SETTXT     ?FB40 SETGR           FB4B SETWND     ?FB5B TABV
 FB60 LOGO          FB65 LO1        ?FB6F SETPWRC         FB78 VIDWAIT
 FB82 VI1           FB8E VI2           FB90 TITLE         FB99 LOCJMP
 FBA0 LOCAL         FBA2 LOCA1         FBB2 LOCA2         FBB6 JLOCAL
 FBB9 RDCHAR1       FBC1 BASCALC       FBD0 BA1           FBD9 BELL1
 FBDF PIP           FBED NOCTRL        FBEE STORINV       FBF0 STORADV
 FBF4 ADVANCE       FBFD VIDOUT     ?FC10 BS          ?FC1A UP
 FC22 VTAB          FC24 VTABZ         FC2A RTS4          FC2B GETUPCS
?FC2F UPPER         FC35 UP1        ?FC36 SW5             FC37 SW6
?FC42 CLREOP        FC46 CLEOP1        FC58 HOME          FC62 CR
 FC66 LF         ?FC70 SCROLL          FC76 SC1           FC95 SC3
 FC9C CLREOL        FC9E CLEOLZ        FCA8 WAIT          FCA9 WA1
```

SYMBOL TABLE SORTED BY ADDRESS

```
 FCAA WA2          FCB4 NXTA4        FCBA NXTA1        FCC8 NX1
 FCC9 SELBNK       FCCC SELBNK2      FCD5 SE1          FCDB CLEOL80
 FCED PLOT80       FCFA SW3          FD03 SW4          FD0C RDKEY
 FD0F LOGO1        FD15 RDKEY2       FD1B KEYIN        FD21 KE1
 FD35 RDCHAR       FD3D NOTCR        FD5F NOTCR1       FD62 CANCEL
 FD67 GETLNZ      ?FD6A GETLN        FD71 BCKSPC       FD75 NXTCHAR
 FD84 ADDINP       FD8E CROUT        FD92 PRA1         FD96 PRYX2
 FDA3 XAM8         FDAD MOD8CHK      FDB3 XAM          FDB6 DATAOUT
 FDC6 SW7          FDCF SW740        FDD7 WDTHOK       FDDA PRBYTE
?FDE3 PRHEX        FDE5 PRHEXZ       FDED COUT        ?FDF0 COUT1
 FDFA NEWLN        FE01 BL1         ?FE05 BLANK       ?FE0C STOR
 FE18 ST1         ?FE19 SETMODE      FE1E SETMDZ      ?FE21 LT
 FE23 LT1          FE2C MOVE         FE36 VERIFY       FE53 VE1
?FE59 BASCONT     ?FE5C XBASIC      ?FE60 LIST         FE63 LI1
 FE75 A1PC         FE78 A1PC1        FE7F A1PC2       ?FE80 SETINV
 FE84 SETNORM      FE86 SETIFLG      FE89 SETKBD      ?FE8B INPORT
?FE8D INPRT        FE93 SETVID      ?FE95 OUTPORT     ?FE97 OUTPRT
 FE9B IOPRT        FEA7 IOPRT1       FEA9 IOPRT2      ?FEB6 GO
?FEBF REGZ         FEC2 SCRN80      ?FECA USR         ?FECD WRITE
 FECE STOR80       FED7 CURS80       FEE1 STRTS        FEE3 VIDRTS
 FEE6 VIDPLP       FEEB VTAB80      ?FEF6 CRMON       ?FEFD READ
 FEFE GET80        FF09 SCR180       FF0F ODDCHR       FF1E EVENCHR
 FF28 SCRLEX      ?FF2D PRERR        FF2E SCRN802      FF3A BELL
 FF3F RESTORE     ?FF44 RESTR1      ?FF4A SAVE         FF4C SAV1
?FF58 IORTS       ?FF59 OLDRST       FF65 MON          FF69 MONZ
 FF73 NXTITM       FF7A CHRSRCH      FF8A DIG          FF90 NXTBIT
 FF98 NXTBAS       FFA2 NXTBS2       FFA7 GETNUM       FFAD NXTCHR
 FFBE TOSUB        FFC7 ZMODE        FFCC CHRTBL       FFDD SUBTBL
 FFEE CLRSC3       FFF9 CLR80
```

```
0000:            2 ****************************************
0000:            3 *                                      *
0000:            4 *     BASIS 108 MONITOR                *
0000:            5 *                                      *
0000:            6 *     25-46  and  25-86                *
0000:            7 *                                      *
0000:            8 *     (C) COPYRIGHT 1982               *
0000:            9 *     BASIS Incorporated               *
0000:           10 *                                      *
0000:           11 *     ALL RIGHTS RESERVED              *
0000:           12 *                                      *
0000:           13 ****************************************
0000:           14 ;
0001:           15 FORTYCOL EQU 1          ;0 = 80 column display
0000:           16 ;                        1 = 40 column display
0000:           17 CFONT1  EQU  0          ;1 = use this character set
0001:           18 CFONT2  EQU  1          ;0 = don't use this character set
0000:           19 CFONT3  EQU  0          ;only 1 character set can be selected at a time
0000:           20 ;
0000:           21 LOC0    EQU  0          ;used for bootstrap
0001:           22 LOC1    EQU  1
0020:           23 WNDLFT  EQU  $20        ;text window left
0021:           24 WNDWDTH EQU  $21        ;  "    "    width
0022:           25 WNDTOP  EQU  $22        ;  "    "    top
0023:           26 WNDBTM  EQU  $23        ;  "    "    bottom+1
0024:           27 CH      EQU  $24        ;cursor horizontal
0025:           28 CV      EQU  $25        ;cursor verticle
0026:           29 GBASL   EQU  $26        ;left end of LoRes line
0027:           30 GBASH   EQU  $27
0028:           31 BASL    EQU  $28        ;left end of TEXT line
0029:           32 BASH    EQU  $29
002A:           33 BAS2L   EQU  $2A        ;destination line address during scroll
002B:           34 BAS2H   EQU  $2B
002C:           35 H2      EQU  $2C        ;right endpoint for HLIN PLOT
002C:           36 LMNEM   EQU  $2C        ;lo byte table index for dissassm
002D:           37 V2      EQU  $2D
002D:           38 RMNEM   EQU  $2D        ;table index for dissassm
002E:           39 MASK    EQU  $2E        ;for plot
002E:           40 FORMAT  EQU  $2E        ;dism instruction format code
002F:           41 LASTIN  EQU  $2F        ;tape in work area for RDBIT
002F:           42 LENGTH  EQU  $2F        ;dism instruction length
0030:           43 COLOR   EQU  $30        ;LoRes color
0031:           44 MODE    EQU  $31        ;parsing for command processor
0032:           45 INVFLG  EQU  $32        ;FF=norm, 3F=inverse, 7F=flash
0033:           46 PROMPT  EQU  $33        ;prompt char
0034:           47 YSAV    EQU  $34        ;save Y during command for line scanning (input)
0035:           48 YSAV1   EQU  $35        ;save Y during screenout by COUT1
0036:           49 CSWL    EQU  $36        ;output vector
0037:           50 CSWH    EQU  $37
0038:           51 KSWL    EQU  $38        ;input vector
0039:           52 KSWH    EQU  $39
003A:           53 PCL     EQU  $3A        ;go-, list-command, program counter save
003B:           54 PCH     EQU  $3B
003C:           55 A1L     EQU  $3C        ;monitor workspace
003D:           56 A1H     EQU  $3D
```

```
003E:           57 A2L      EQU  $3E
003F:           58 A2H      EQU  $3F
0040:           59 A3L      EQU  $40        ;memory set
0041:           60 A3H      EQU  $41
0042:           61 A4L      EQU  $42
0043:           62 A4H      EQU  $43
0045:           63 ACC      EQU  $45        ;6502 register save area
0046:           64 XREG     EQU  ACC+1      ;  "      "      "    "
0047:           65 YREG     EQU  ACC+2      ;  "      "      "    "
0048:           66 STATUS   EQU  ACC+3      ;  "      "      "    "
0049:           67 SPNT     EQU  ACC+4      ;  "      "      "    "
004E:           68 RNDL     EQU  $4E        ; random number
004F:           69 RNDH     EQU  $4F
0080:           70 BIT7     EQU  $80        ;mask
00C0:           71 IOPAGE   EQU  $C0
0200:           72 IN       EQU  $200       ;keyboard buffer (page)
03F0:           73 BRKV     EQU  $3F0       ;BRK vector (addr)
03F2:           74 SOFTEV   EQU  $3F2       ;soft RESET vector (addr)
03F4:           75 PWREDUP  EQU  $3F4       ;reboot if not (3F3) XOR $A5
03F5:           76 AMPERV   EQU  $3F5       ;Applesoft & vector (instr)
03F8:           77 USRADR   EQU  $3F8       ;U-command Ctrl-Y vector (instr)
03FB:           78 NMI      EQU  $3FB       ;NMI vector (instr)
03FE:           79 IRQLOC   EQU  $3FE       ;IRQ vecto (addr)
0400:           80 LINE1    EQU  $400       ;first screen line, LoRes pg1
07F8:           81 MSLOT    EQU  $7F8       ;active slot ID ($Cn)
C000:           82 IOARD    EQU  $C000
C000:           83 KBD      EQU  $C000      ;ASCII input
C000:           84 CHRINV   EQU  $C000      ;invers/flash switch
C002:           85 CHRGEN0  EQU  $C002      ;char gen A10
C004:           86 CHRGEN1  EQU  $C004      ;char gen A11
C006:           87 CHRBAS   EQU  $C006      ;64+64+128 set (inverse, flash, normal)
C008:           88 KBDEXTN  EQU  $C008      ;functions key input
C00A:           89 VID40    EQU  $C00A      ;40/80 col switch
C00B:           90 VID80    EQU  $C00B
C00C:           91 VIDBNK   EQU  $C00C      ;video RAM switch C00D=off
C010:           92 KBDSTRB  EQU  $C010      ;for alternate 80 columns
C030:           93 SPKR     EQU  $C030
C050:           94 TXTCLR   EQU  $C050      ;video mode swithes
C052:           95 MIXCLR   EQU  $C052      ;"
C054:           96 LOWSCR   EQU  $C054      ;"
C056:           97 LORES    EQU  $C056      ;"
C058:           98 TTLOUT0  EQU  $C058      ;even:off,low <=0.4V
C05A:           99 TTLOUT1  EQU  $C05A      ; odd:on,high >=2.4V
C05C:          100 TTLOUT2  EQU  $C05C
C05E:          101 TTLOUT3  EQU  $C05E
C064:          102 PADDL0   EQU  $C064
C070:          103 PTRIG    EQU  $C070
CFFF:          104 CLRROM   EQU  $CFFF
E000:          105 BASIC    EQU  $E000      ;cold start address for Basic
E003:          106 BASIC2   EQU  $E003      ;warm start address
0000:          107 *
0001:          108          DO   FORTYCOL
0028:          109 WIDTH    EQU  40
002E:          110 CHKSUM   EQU  $2E        ;for tape read
C020:          111 TAPEOUT  EQU  $C020
C060:          112 TAPEIN   EQU  $C060
```

```
0000:           114 *
0000:           115         ELSE
0000:           116 *
0000         S  117 WIDTH   EQU  80
0000         S  118 CHY     EQU  $479      ;80 column video driver
0000         S  119         SBTL "BASIS    108 MONITOR 25-86"
0000:           120         FIN
0000:           121 ****************************************
----- NEXT OBJECT FILE NAME IS MONITOR 25-46
F800:           122         ORG  $F800
F800:           123 ****************************************
F800:           124 *
F800:           125 * PLOT A POINT
F800:           126 *
F800:           127 ****************************************
F800:4A         128 PLOT    LSR  A          ;Y coord LSB to carry
F801:08         129         PHP             ;save
F802:20 47 F8   130         JSR  GBASCALC   ;set addr in GBASL,H
F805:28         131         PLP             ;retrieve LSB
F806:A9 0F      132         LDA  #$F        ;mask=$F if even
F808:90 02      133         BCC  PL1
F80A:69 E0      134         ADC  #$E0       ;OR $F0 if odd
F80C:85 2E      135 PL1     STA  MASK
F80E:           136 *
0001:           137 PLOT1   DO   FORTYCOL
F80E:B1 26      138         LDA  (GBASL),Y ;get data
F810:45 30      139         EOR  COLOR     ;set color
F812:25 2E      140         AND  MASK      ;pick hi or lo nibble
F814:51 26      141         EOR  (GBASL),Y ;restore other nibble
F816:91 26      142         STA  (GBASL),Y ;put it back
F818:60         143         RTS
F819:           144 *
F819:           145         ELSE
F819:           146 *
F819         S  147         PHP             ;save LSB
F819         S  148         JSR  SELBNK     ;set bank
F819         S  149         JMP  PLOT80     ;plot point
F819         S  150         DFB  0,0,0,0    ;fillers
F819:           151         FIN
F819:           152 *
F819:           153 ****************************************
F819:           154 *
F819:           155 * DRAW HORIZONTAL LINE
F819:           156 *
F819:           157 ****************************************
F819:20 00 F8   158 HLINE   JSR  PLOT       ;Basic HLINE,plot a point
F81C:C4 2C      159 HL1     CPY  H2         ;end of line?
F81E:B0 11      160         BCS  RTS1       ;yes return
F820:C8         161         INY             ;no, point to next point in pair
F821:20 0E F8   162         JSR  PLOT1      ;plot it
F824:90 F6      163         BCC  HL1        ;do it again (always)
F826:           164 ****************************************
F826:           165 *
F826:           166 * DRAW VERTICAL LINE
F826:           167 *
```

```
F826:           168 ****************************************
F826:69 01      169 VLINEZ  ADC  #1         ;increment Y coord
F828:48         170 VLINE   PHA             ;Basic VLINE,save it
F829:20 00 F8   171         JSR  PLOT       ;plot point
F82C:68         172         PLA             ;retrieve next
F82D:C5 2D      173         CMP  V2         ;end of line ?
F82F:90 F5      174         BCC  VLINEZ     ;no, do it again
F831:60         175 RTS1    RTS             ;yes,return
F832:           176 ****************************************
F832:           177 *
F832:           178 * CLEAR SCREEN
F832:           179 *
F832:           180 ****************************************
F832:A0 2F      181 CLRSCR  LDY  #$2F       ; Y-max,full screen clear
F834:D0 02      182         BNE  CLRSC2     ;always
F836:A0 27      183 CLRTOP  LDY  #$27       ;Y-max,mixed clear
F838:84 2D      184 CLRSC2  STY  V2         ;save bottom coord
F83A:A0 27      185         LDY  #WIDTH-1   ;X-max
F83C:A9 00      186 CL1     LDA  #0         ;starting point
F83E:85 30      187         STA  COLOR      ;set black
F840:20 28 F8   188         JSR  VLINE      ;draw verticle
F843:88         189         DEY             ;reduce count
F844:10 F6      190         BPL  CL1        ;if not done
F846:60         191         RTS
F847:           192 ****************************************
F847:           193 *
F847:           194 * TRANSLATE LINE # TO BASE ADDRESS
F847:           195 *
F847:           196 ****************************************
F847:48         197 GBASCALC PHA            ;line # bits = 000ABCDE
F848:4A         198         LSR  A
F849:29 03      199         AND  #3
F84B:09 04      200         ORA  #4         ;for LoRes Page 1
F84D:85 27      201         STA  GBASH      ;GBASH=000001CD
F84F:68         202         PLA
F850:29 18      203         AND  #$18
F852:90 02      204         BCC  GB1
F854:09 80      205         ORA  #$80
F856:85 26      206 GB1     STA  GBASL
F858:0A         207         ASL  A
F859:0A         208         ASL  A
F85A:05 26      209         ORA  GBASL
F85C:85 26      210         STA  GBASL      ;GBASL=EABAB000
F85E:60         211         RTS
F85F:           212 ****************************************
F85F:           213 *
F85F:           214 * INCREMENT COLOR
F85F:           215 *
F85F:           216 ****************************************
F85F:A5 30      217 NXTCOL  LDA  COLOR      ;color=color+3
F861:18         218         CLC
F862:69 03      219         ADC  #3
F864:           220 ****************************************
F864:           221 *
F864:           222 * SET COLOR
```

```
F864:           223 *
F864:           224 ****************************************
F864:29 0F      225 SETCOL  AND  #$F        ;COLOR=17*A mod 16
F866:85 30      226         STA  COLOR
F868:0A         227         ASL  A          ;both nibbles = color
F869:0A         228         ASL  A
F86A:0A         229         ASL  A
F86B:0A         230         ASL  A
F86C:05 30      231         ORA  COLOR
F86E:85 30      232         STA  COLOR
F870:60         233         RTS
F871:           234 ****************************************
F871:           235 *
F871:           236 * READ SCREEN POINT
F871:           237 *
F871:           238 ****************************************
F871:4A         239 SCRN    LSR  A          ;Y coord LSB into carry
F872:08         240         PHP             ;save on stack
F873:           241 *
0001:           242         DO   FORTYCOL
F873:20 47 F8   243         JSR  GBASCALC   ;set address
F876:B1 26      244         LDA  (GBASL),Y  ;get data
F878:           245 *
F878:           246         ELSE
F878:           247 *
F878         S  248         JSR  SCRNX      ;read point status
F878         S  249         NOP             ;filler
F878         S  250         NOP
F878:           251         FIN
F878:           252 *
F878:28         253         PLP             ;retrieve LSB
F879:90 04      254 SCRN2   BCC  SCR1       ;use low nibble
F87B:4A         255         LSR  A          ;or move high nibble down
F87C:4A         256         LSR  A
F87D:4A         257         LSR  A
F87E:4A         258         LSR  A
F87F:29 0F      259 SCR1    AND  #$F        ;drop the top
F881:60         260         RTS
F882:           261 ****************************************
F882:           262 *
F882:           263 * DISASSEMBLER DECODER/ERROR CHECK
F882:           264 *
F882:           265 ****************************************
F882:A6 3A      266 INSDS1  LDX  PCL        ;set pc
F884:A4 3B      267         LDY  PCH
F886:20 96 FD   268         JSR  PRYX2      ;display it
F889:20 48 F9   269         JSR  PRBLNK     ;print a blank
F88C:A1 3A      270 INSDS2  LDA  (PCL,X)    ;set opcode
F88E:A8         271         TAY             ;save in Y
F88F:4A         272         LSR  A
F890:90 09      273         BCC  IEVEN      ;if even, go ahead
F892:6A         274         ROR  A
F893:B0 10      275         BCS  ERR        ;all xxxxxx11 opcodes are illegal
F895:C9 A2      276         CMP  #$A2       ;no STA # operation
F897:F0 0C      277         BEQ  ERR
```

```
F899:29 87     278          AND  #$87      ;mask
F89B:4A        279 IEVEN    LSR  A         ;LSB into carry for nybble select
F89C:AA        280          TAX
F89D:BD 62 F9  281          LDA  FMT1,X    ;get format
F8A0:20 79 F8  282          JSR  SCRN2     ;select hi or lo nybble
F8A3:D0 04     283          BNE  GETFMT
F8A5:A0 80     284 ERR      LDY  #$80      ;substitute $80 if opcode invalid
F8A7:A9 00     285          LDA  #0        ;format index=0
F8A9:AA        286 GETFMT   TAX
F8AA:BD A6 F9  287          LDA  FMT2,X    ;get output format
F8AD:85 2E     288          STA  FORMAT
F8AF:29 03     289          AND  #3        ;extract length 1=2 bytes, 2=3 bytes
F8B1:85 2F     290          STA  LENGTH
F8B3:98        291          TYA             ;get opcode back
F8B4:29 8F     292          AND  #$8F      ;mask
F8B6:AA        293          TAX            ;and save
F8B7:98        294          TYA            ;get it  again
F8B8:A0 03     295          LDY  #3        ;calculate index into mnemonic table
F8BA:E0 8A     296          CPX  #$8A      ;"
F8BC:F0 0B     297          BEQ  MNNDX3    ;"
F8BE:4A        298 MNNDX1   LSR  A         ;"
F8BF:90 08     299          BCC  MNNDX3    ;"
F8C1:4A        300          LSR  A         ;"
F8C2:4A        301 MNNDX2   LSR  A         ;"
F8C3:09 20     302          ORA  #$20      ;"
F8C5:88        303          DEY            ;"
F8C6:D0 FA     304          BNE  MNNDX2    ;"
F8C8:C8        305          INY            ;"
F8C9:88        306 MNNDX3   DEY            ;"
F8CA:D0 F2     307          BNE  MNNDX1    ;"
F8CC:60        308          RTS
F8CD:00 00 00  309          DFB  0,0,0
F8D0:          310 ****************************************
F8D0:          311 *
F8D0:          312 * DISASSEMBLER OUTPUT
F8D0:          313 *
F8D0:          314 ****************************************
F8D0:20 82 F8  315 INSTDSP  JSR  INSDS1    ;get FMT, LEN, INDEX, output PC
F8D3:48        316          PHA            ;save index
F8D4:B1 3A     317 PRNTOP   LDA  (PCL),Y   ;output instruction (1-3 byte hex)
F8D6:20 DA FD  318          JSR  PRBYTE
F8D9:A2 01     319          LDX  #1        ;interbyte spacing=2 blanks
F8DB:20 4A F9  320 PRNTBL   JSR  PRBL2
F8DE:C4 2F     321          CPY  LENGTH    ;whole instruction output?
F8E0:C8        322          INY
F8E1:90 F1     323          BCC  PRNTOP    ;no, output next byte
F8E3:A2 03     324          LDX  #3        ;3 char mnemonics
F8E5:C0 04     325          CPY  #4        ;align columns
F8E7:90 F2     326          BCC  PRNTBL
F8E9:68        327          PLA            ;retrieve index
F8EA:A8        328          TAY
F8EB:B9 C0 F9  329          LDA  MNEML,Y   ;retreive mnemonic (3 chars in 2 bytes)
F8EE:85 2C     330          STA  LMNEM
F8F0:B9 00 FA  331          LDA  MNEMR,Y
F8F3:85 2D     332          STA  RMNEM
```

```
F8F5:A9 00     333 PR1     LDA  #0
F8F7:A0 05     334         LDY  #5         ;shift 5 BITs in ACC
F8F9:06 2D     335 PR2     ASL  RMNEM
F8FB:26 2C     336         ROL  LMNEM
F8FD:2A        337         ROL  A
F8FE:88        338         DEY
F8FF:D0 F8     339         BNE  PR2
F901:69 BF     340         ADC  #$BF       ;offset for ?
F903:20 ED FD  341         JSR  COUT       ;output a character
F906:CA        342         DEX             ;count it
F907:D0 EC     343         BNE  PR1        ;do next char if not done
F909:20 48 F9  344         JSR  PRBLNK     ;output 3 blanks when done
F90C:A4 2F     345         LDY  LENGTH
F90E:A2 06     346         LDX  #6         ;6 format bits
F910:E0 03     347 PRADR1  CPX  #3         ;3 byte instructins
F912:F0 1C     348         BEQ  PRADR5     ;contans addresses
F914:06 2E     349 PRADR2  ASL  FORMAT
F916:90 0E     350         BCC  PRADR3
F918:BD B3 F9  351         LDA  CHAR1-1,X
F91B:20 ED FD  352         JSR  COUT
F91E:BD B9 F9  353         LDA  CHAR2-1,X
F921:F0 03     354         BEQ  PRADR3     ;no 2nd char
F923:20 ED FD  355         JSR  COUT
F926:CA        356 PRADR3  DEX
F927:D0 E7     357         BNE  PRADR1     ;next format BIT
F929:60        358         RTS
F92A:88        359 PRADR4  DEY
F92B:30 E7     360         BMI  PRADR2
F92D:20 DA FD  361         JSR  PRBYTE
F930:A5 2E     362 PRADR5  LDA  FORMAT
F932:C9 E8     363         CMP  #$E8       ;relative addressing
F934:B1 3A     364         LDA  (PCL),Y
F936:90 F2     365         BCC  PRADR4
F938:20 56 F9  366 RELADR  JSR  PCADJ3     ;yes, output destination instead of offset
F93B:AA        367         TAX
F93C:E8        368         INX
F93D:D0 01     369         BNE  PRNTYX
F93F:C8        370         INY
F940:98        371 PRNTYX  TYA
F941:20 DA FD  372 PRNTAX  JSR  PRBYTE
F944:8A        373 PRNTX   TXA
F945:4C DA FD  374         JMP  PRBYTE
F948:A2 03     375 PRBLNK  LDX  #3         ;output 3 blanks
F94A:A9 A0     376 PRBL2   LDA  #$A0
F94C:20 ED FD  377 PRBL3   JSR  COUT
F94F:CA        378         DEX
F950:D0 F8     379         BNE  PRBL2
F952:60        380         RTS
F953:38        381 PCADJ   SEC             ;adjust PC to next instruction
F954:A5 2F     382 PCADJ2  LDA  LENGTH      ;length = instr len-1
F956:A4 3B     383 PCADJ3  LDY  PCH
F958:AA        384         TAX
F959:10 01     385         BPL  PCADJ4     ;check for backward branch
F95B:88        386         DEY             ;if so decrement
F95C:65 3A     387 PCADJ4  ADC  PCL
```

```
F95E:90 01     388         BCC  RTS2
F960:C8        389         INY
F961:60        390 RTS2    RTS
F962:          391 ****************************************
F962:          392 *
F962:          393 * FMT1: 128 (DEC) 4-BIT pointer to the FMT2 table for
F962:          394 * xxxx xxx0 opcodes
F962:          395 * 16 (DEC) 4-BIT pointer to the FMT2 table for
F962:          396 * xxxx xx01 opcodes
F962:          397 *
F962:          398 * if Y=0 use high nybble
F962:          399 * if Y=1 use low nybble
F962:          400 * X=index
F962:          401 *
F962:          402 ****************************************
F962:04 20 54  403 FMT1    DFB  $04,$20,$54,$30,$0D,$80,$04,$90
F965:30 0D 80
F968:04 90
F96A:03 22 54  404         DFB  $03,$22,$54,$33,$0D,$80,$04,$90
F96D:33 0D 80
F970:04 90
F972:04 20 54  405         DFB  $04,$20,$54,$33,$0D,$80,$04,$90
F975:33 0D 80
F978:04 90
F97A:04 20 54  406         DFB  $04,$20,$54,$3B,$0D,$80,$04,$90
F97D:3B 0D 80
F980:04 90
F982:00 22 44  407         DFB  $00,$22,$44,$33,$0D,$C8,$44,$00
F985:33 0D C8
F988:44 00
F98A:11 22 44  408         DFB  $11,$22,$44,$33,$0D,$C8,$44,$A9
F98D:33 0D C8
F990:44 A9
F992:01 22 44  409         DFB  $01,$22,$44,$33,$0D,$80,$04,$90
F995:33 0D 80
F998:04 90
F99A:01 22 44  410         DFB  $01,$22,$44,$33,$0D,$80,$04,$90
F99D:33 0D 80
F9A0:04 90
F9A2:          411 * xxxx xx01 class:
F9A2:26 31 87  412         DFB  $26,$31,$87,$9A ; ORA,AND,EOR,ADC,STA,LDA,CMD,SBC
F9A5:9A
F9A6:          413 ****************************************
F9A6:          414 *
F9A6:          415 * FMT2 BIT 0..1 : instruction length-1
F9A6:          416 * FMT2 BIT 7..2 : if BIT[i] then (print chr1[i-2],chr2[i-2])
F9A6:          417 *
F9A6:          418 ****************************************
F9A6:00        419 FMT2    DFB  $00       ;illegal opcode
F9A7:21        420         DFB  $21       ;immediate
F9A8:81        421         DFB  $81       ;zero page
F9A9:82        422         DFB  $82       ;absolute
F9AA:00        423         DFB  $00       ;register
F9AB:00        424         DFB  $00       ;accumulator
F9AC:59        425         DFB  $59       ;(zero page,X)
```

```
F9AD:4D          426          DFB  $4D        ;(zero page),Y
F9AE:91          427          DFB  $91        ;zero page,X
F9AF:92          428          DFB  $92        ;absolute,X
F9B0:86          429          DFB  $86        ;absolute,Y
F9B1:4A          430          DFB  $4A        ;(absolute)
F9B2:85          431          DFB  $85        ;zero page,Y
F9B3:9D          432          DFB  $9D        ;relative
F9B4:            433 ; char1/char2 used by mini assambler
F9B4:AC A9 AC    434 CHAR1    DFB  $AC,$A9,$AC,$A3,$A8,$A4 ; ",),#($"
F9B7:A3 A8 A4
F9BA:D9 00 D8    435 CHAR2    DFB  $D9,$00,$D8,$A4,$A4,$00 ; "Y X$$ "
F9BD:A4 A4 00
F9C0:            436 *
F9C0:            437 * BBBBB000  ;class instructions
F9C0:1C          438 MNEML    DFB  $1C        ; BRK
F9C1:8A          439          DFB  $8A        ; PHP
F9C2:1C          440          DFB  $1C        ; BPL
F9C3:23          441          DFB  $23        ; CLC
F9C4:5D          442          DFB  $5D        ; JSR
F9C5:8B          443          DFB  $8B        ; PLP
F9C6:1B          444          DFB  $1B        ; BMI
F9C7:A1          445          DFB  $A1        ; SEC
F9C8:9D          446          DFB  $9D        ; RTI
F9C9:8A          447          DFB  $8A        ; PHA
F9CA:1D          448          DFB  $1D        ; BVC
F9CB:23          449          DFB  $23        ; CLI
F9CC:9D          450          DFB  $9D        ; RTS
F9CD:8B          451          DFB  $8B        ; PLA
F9CE:1D          452          DFB  $1D        ; BVS
F9CF:A1          453          DFB  $A1        ; SEI
F9D0:00          454          DFB  $00        ; ?
F9D1:29          455          DFB  $29        ; DEY
F9D2:19          456          DFB  $19        ; BCC
F9D3:AE          457          DFB  $AE        ; TYA
F9D4:69          458          DFB  $69        ; LDY
F9D5:A8          459          DFB  $A8        ; TAY
F9D6:19          460          DFB  $19        ; BCS
F9D7:23          461          DFB  $23        ; CLV
F9D8:24          462          DFB  $24        ; CPY
F9D9:53          463          DFB  $53        ; IBY
F9DA:1B          464          DFB  $1B        ; BNE
F9DB:23          465          DFB  $23        ; CLD
F9DC:24          466          DFB  $24        ; CPX
F9DD:53          467          DFB  $53        ; INX
F9DE:19          468          DFB  $19        ; BEQ
F9DF:A1          469          DFB  $A1        ; SED
F9E0:            470 * IBBBxx100:
F9E0:00          471          DFB  $00        ; ?
F9E1:1A          472          DFB  $1A        ; BIT
F9E2:5B          473          DFB  $5B        ; JMP
F9E3:5B          474          DFB  $5B        ; JMP
F9E4:A5          475          DFB  $A5        ; STY
F9E5:69          476          DFB  $69        ; LDY
F9E6:24          477          DFB  $24        ; CPY
F9E7:24          478          DFB  $24        ; CPX
```

```
F9E8:           479 ; 1BBB1010:
F9E8:AE         480         DFB  $AE        ; TXA
F9E9:AE         481         DFB  $AE        ; TXS
F9EA:A8         482         DFB  $A8        ; TAX
F9EB:AD         483         DFB  $AD        ; TSX
F9EC:29         484         DFB  $29        ; DEX
F9ED:00         485         DFB  $00        ; ?
F9EE:7C         486         DFB  $7C        ; NOP
F9EF:00         487         DFB  $00        ; ?
F9F0:           488 * 0IIxxx10:
F9F0:15         489         DFB  $15        ; ASL
F9F1:9C         490         DFB  $9C        ; ROL
F9F2:6D         491         DFB  $6D        ; LSR
F9F3:9C         492         DFB  $9C        ; ROR
F9F4:           493 * 1IIx0010, 1IIx0110, 1IIx1110:
F9F4:A5         494         DFB  $A5        ; STX
F9F5:69         495         DFB  $69        ; LDX
F9F6:29         496         DFB  $29        ; DEC
F9F7:53         497         DFB  $53        ; INC
F9F8:           498 * IIIxxx01:
F9F8:84         499         DFB  $84        ; ORA
F9F9:13         500         DFB  $13        ; AND
F9FA:34         501         DFB  $34        ; EOR
F9FB:11         502         DFB  $11        ; ADC
F9FC:A5         503         DFB  $A5        ; STA
F9FD:69         504         DFB  $69        ; LDA
F9FE:23         505         DFB  $23        ; CMP
F9FF:A0         506         DFB  $A0        ; SBC
FA00:D8 62 5A   507 MNEMR   DFB  $D8,$62,$5A,$48,$26,$62,$94,$88
FA03:48 26 62
FA06:94 88
FA08:54 44 C8   508         DFB  $54,$44,$C8,$54,$68,$44,$E8,$94
FA0B:54 68 44
FA0E:E8 94
FA10:00 B4 08   509         DFB  $00,$B4,$08,$84,$74,$B4,$28,$6E
FA13:84 74 B4
FA16:28 6E
FA18:74 F4 CC   510         DFB  $74,$F4,$CC,$4A,$72,$F2,$A4,$8A
FA1B:4A 72 F2
FA1E:A4 8A
FA20:00 AA A2   511         DFB  $00,$AA,$A2,$A2,$74,$74,$74,$72
FA23:A2 74 74
FA26:74 72
FA28:44 68 B2   512         DFB  $44,$68,$B2,$32,$B2,$00,$22,$00
FA2B:32 B2 00
FA2E:22 00
FA30:1A 1A 26   513         DFB  $1A,$1A,$26,$26
FA33:26
FA34:72 72 88   514         DFB  $72,$72,$88,$C8
FA37:C8
FA38:C4 CA 26   515         DFB  $C4,$CA,$26,$48,$44,$44,$A2,$C8
FA3B:48 44 44
FA3E:A2 C8
FA40:           516 ****************************************
FA40:           517 *
```

```
FA40:           518 * INTERNAL INTERRUPT HANDLER
FA40:           519 *
FA40:           520 ****************************************
FA40:85 45      521 IRQ     STA  ACC        ;save accumulator
FA42:68         522         PLA             ;get top of stack
FA43:48         523         PHA             ;put it back
FA44:29 10      524         AND  #$10       ; test break flag, BIT 4
FA46:D0 04      525         BNE  BREAK      ;yes
FA48:6C FE 03   526         JMP  (IRQLOC)   ;no,goto interrupt handler
FA4B:00         527         DFB  0
FA4C:           528 ****************************************
FA4C:           529 *
FA4C:           530 * INTERNAL BREAK HANDLER
FA4C:           531 *
FA4C:           532 ****************************************
FA4C:28         533 BREAK   PLP             ;pull status
FA4D:20 4C FF   534         JSR  SAV1       ;save registers
FA50:68         535         PLA             ;save PC
FA51:85 3A      536         STA  PCL
FA53:68         537         PLA
FA54:85 3B      538         STA  PCH
FA56:6C F0 03   539         JMP  (BRKV)     ;goto breakhandler
FA59:           540 ****************************************
FA59:           541 *
FA59:           542 * MONITOR BREAK HANDLER
FA59:           543 *
FA59:           544 ****************************************
FA59:20 82 F8   545 OLDBRK  JSR  INSDS1     ;output PC
FA5C:20 DA FA   546         JSR  RGDSP1     ;and registers
FA5F:4C 65 FF   547         JMP  MON        ;enter monitor
FA62:           548 ****************************************
FA62:           549 *
FA62:           550 * RESET HANDLER
FA62:           551 *
FA62:           552 ****************************************
FA62:D8         553 RESET   CLD
FA63:20 84 FE   554         JSR  SETNORM
FA66:20 2F FB   555         JSR  INIT
FA69:20 93 FE   556         JSR  SETVID     ;console output
FA6C:20 89 FE   557         JSR  SETKBD     ;console input
FA6F:D8         558 NEWMON  CLD             ;not decimal mode
FA70:20 3A FF   559         JSR  BELL
FA73:           560 *
0001:           561         DO   FORTYCOL
FA73:8D 0A C0   562         STA  VID40      ;set 40 column display mode
FA76:           563 *
FA76:           564         ELSE
FA76:           565 *
FA76        S   566         STA  VID80      ;set 80 column display mode
FA76:           567         FIN
FA76:           568 *
0000:           569         DO   CFONT1     ;native 46-3
FA76        S   570         STA  CHRGEN0+1 ;turn on char set
FA76        S   571         STA  CHRGEN1
FA76:           572         FIN
```

```
FA76:           573 *
0001:           574         DO   CFONT2     ;ASCII 46-3
FA76:8D 02 C0   575         STA  CHRGEN0    ;turn on char set
FA79:8D 05 C0   576         STA  CHRGEN1+1
FA7C:           577         FIN
FA7C:           578 *
0000:           579         DO   CFONT3     ;APL 46-3
FA7C         S  580         STA  CHRGEN0+1 ;turn on char set
FA7C         S  581         STA  CHRGEN1+1
FA7C:           582         FIN
FA7C:           583 *
FA7C:8D 00 C0   584         STA  CHRINV
FA7F:2C FF CF   585         BIT  CLRROM     ;disable extension ROM
FA82:2C 10 C0   586         BIT  KBDSTRB    ;clear keyboard strobe
FA85:AD F3 03   587         LDA  SOFTEV+1   ;reset high?
FA88:49 A5      588         EOR  #$A5
FA8A:CD F4 03   589         CMP  PWREDUP
FA8D:D0 17      590         BNE  PWRUP      ;do,it's the first time
FA8F:AD F2 03   591         LDA  SOFTEV     ;is it cold start
FA92:D0 0F      592         BNE  NOFIX      ;no,use reset vector
FA94:A9 E0      593         LDA  #$E0       ;is it basic vector
FA96:CD F3 03   594         CMP  SOFTEV+1
FA99:D0 08      595         BNE  NOFIX      ;no,use reset vector
FA9B:A0 03      596 FIXSEV  LDY  #3         ;yes,point to warm start
FA9D:8C F2 03   597         STY  SOFTEV
FAA0:4C 00 E0   598         JMP  BASIC      ;and do cold start
FAA3:6C F2 03   599 NOFIX   JMP  (SOFTEV)   ;goto reset vector
FAA6:           600 ****************************************
FAA6:           601 *
FAA6:           602 * POWER ON START
FAA6:           603 *
FAA6:           604 ****************************************
FAA6:20 60 FB   605 PWRUP   JSR  LOGO       ;display Logo
FAA9:A2 05      606 SETPG3  LDX  #5         ;set vector for Basic
FAAB:BD FC FA   607 SETPLP  LDA  PWRCON-1,X
FAAE:9D EF 03   608         STA  BRKV-1,X
FAB1:CA         609         DEX
FAB2:D0 F7      610         BNE  SETPLP
FAB4:A9 C8      611         LDA  #$C8
FAB6:85 01      612         STA  LOC1       ;set PTR H to highest slot+1
FAB8:86 00      613         STX  LOC0       ;PTRL=0
FABA:A0 07      614 SLOOP   LDY  #7         ;Y is byte offset into the slot ROM
FABC:C6 01      615         DEC  LOC1       ;start scanning the slots
FABE:A5 01      616         LDA  LOC1
FAC0:C9 C1      617         CMP  #$C1       ;slot=1?
FAC2:F0 D7      618         BEQ  FIXSEV     ;yes,there isn't any disk
FAC4:8D F8 07   619         STA  MSLOT
FAC7:B1 00      620 SL1     LDA  (LOC0),Y   ;try to read slot ROM
FAC9:D9 01 FB   621         CMP  DISKID,Y   ;is it a boot device (floppy, harddisk...) ??
FACC:D0 EC      622         BNE  SLOOP      ;no, test next slot
FACE:88         623         DEY
FACF:88         624         DEY             ;yes so check next odd byte
FAD0:10 F5      625         BPL  SL1
FAD2:6C 00 00   626         JMP  (LOC0)     ;it is a disk! jump to ROM
FAD5:00 00      627         DFB  0,0
```

```
FAD7:                 628 ****************************************
FAD7:                 629 *
FAD7:                 630 * DISPLAY REGISTER CONTENTS
FAD7:                 631 *
FAD7:                 632 ****************************************
FAD7:20 8E FD         633 REGDSP  JSR  CROUT      ;new line
FADA:A9 45            634 RGDSP1  LDA  #ACC       ;get pointer to save area
FADC:85 40            635         STA  A3L
FADE:A9 00            636         LDA  #0
FAE0:85 41            637         STA  A3H
FAE2:A2 FB            638         LDX  #$FB       ;register count
FAE4:A9 A0            639 RG1     LDA  #$A0       ;output a blank
FAE6:20 ED FD         640         JSR  COUT
FAE9:BD 1E FA         641         LDA  RTBL-251,X;get register label
FAEC:20 ED FD         642         JSR  COUT
FAEF:A9 BD            643         LDA  #$BD       ;output=sign
FAF1:20 ED FD         644         JSR  COUT
FAF4:B5 4A            645         LDA  ACC+5,X    ;get register contents>
FAF6:20 DA FD         646         JSR  PRBYTE     ;print it
FAF9:E8               647         INX             ;index to next
FAFA:30 E8            648         BMI  RG1        ;repeat until done
FAFC:60               649         RTS             ;then return
FAFD:                 650 ****************************************
FAFD:59 FA            651 PWRCON  DW   OLDBRK
FAFF:00 E0            652         DW   BASIC
FB01:45 20            653 DISKID  EOR  $20        ;opcode (OE0^0A5=45) used for mask!!
FB03:A0 00            654         LDY  #0         ;code never executed,
FB05:A2 03            655         LDX  #3         ;only for disk ID
FB07:86 3C            656         STX  $3C
FB09:08 15 0A         657 LOCCHR  DFB  $08,$15,$0A,$0B,$40,$0E,$0F ; special characters
FB0C:0B 40 0E
FB0F:0F
FB10:                 658 *
0001:                 659         DO   FORTYCOL
FB10:8D 0A C0         660         STA  VID40      ;video switch
FB13:                 661 *
FB13:                 662         ELSE
FB13:                 663 *
FB13        S         664         STA  VID80      ;video switch
FB13:                 665         FIN
FB13:                 666 *
FB13:60               667         RTS
FB14:00 00 00         668         DFB  0,0,0,0,0 ;fillers
FB17:00 00
FB19:C1 D8 D9         669 RTBL    DFB  $C1,$D8,$D9,$D0,$D3 ;register labels "AXYPS"
FB1C:D0 D3
FB1E:                 670 ****************************************
FB1E:                 671 *
FB1E:                 672 * BASIC PDL(n) FUNCTION
FB1E:                 673 *
FB1E:                 674 ****************************************
FB1E:AD 70 C0         675 PREAD   LDA  PTRIG      ;Trigger read
FB21:A0 00            676         LDY  #0         ;initiate count
FB23:EA               677         NOP             ;timing compensation
FB24:EA               678         NOP
```

```
FB25:BD 64 C0  679 PRE1    LDA  PADDL0,X  ;test every 12 usec
FB28:10 04     680         BPL  RTSX      ;exit if read complete
FB2A:C8        681         INY            ;or count loop in Y
FB2B:D0 F8     682         BNE  PRE1      ;and do it again (max 255)
FB2D:88        683         DEY            ;back up to 255 if full count
FB2E:60        684 RTSX    RTS
FB2F:          685 ****************************************
FB2F:          686 *
FB2F:          687 * VIDEO INITIALIZATION
FB2F:          688 *
FB2F:          689 ****************************************
FB2F:A9 04     690 INIT    LDA  #4        ;set I flag!
FB31:85 48     691         STA  STATUS
FB33:2C 56 C0  692         BIT  LORES     ;set screen switches
FB36:2C 54 C0  693         BIT  LOWSCR
FB39:          694 ****************************************
FB39:          695 *
FB39:          696 * SET TEXT MODE, BASIC TEXT
FB39:          697 *
FB39:          698 ****************************************
FB39:2C 51 C0  699 SETTXT  BIT  TXTCLR+1  ;text switch
FB3C:A9 00     700         LDA  #0        ;always skip
FB3E:F0 0B     701         BEQ  SETWND    ;to SETWND
FB40:          702 ****************************************
FB40:          703 *
FB40:          704 * SET GRAPHICS, BASIC GR
FB40:          705 *
FB40:          706 ****************************************
FB40:2C 50 C0  707 SETGR   BIT  TXTCLR
FB43:2C 53 C0  708         BIT  MIXCLR+1  ;set mixed mode
FB46:20 36 F8  709         JSR  CLRTOP    ;clear graphics area
FB49:A9 14     710         LDA  #20       ;top of text window
FB4B:          711 ****************************************
FB4B:          712 *
FB4B:          713 * ESTABLISH WINDOW
FB4B:          714 *
FB4B:          715 ****************************************
FB4B:85 22     716 SETWND  STA  WNDTOP
FB4D:A9 00     717         LDA  #0        ;full width
FB4F:85 20     718         STA  WNDLFT
FB51:A9 28     719         LDA  #WIDTH
FB53:85 21     720         STA  WNDWDTH
FB55:A9 18     721         LDA  #24       ;to last line
FB57:85 23     722         STA  WNDBTM
FB59:A9 17     723         LDA  #23       ;set vertical tab to 23
FB5B:85 25     724 TABV    STA  CV
FB5D:4C 22 FC  725         JMP  VTAB
FB60:          726 ****************************************
FB60:          727 *
FB60:          728 * PUT "Basis 108" ON TOP OF SCREEN
FB60:          729 *
FB60:          730 ****************************************
FB60:20 58 FC  731 LOGO    JSR  HOME      ;clear the screen
FB63:A0 08     732         LDY  #8        ;title length
FB65:B9 90 FB  733 L01     LDA  TITLE,Y   ;get a char
```

```
FB68:99 00 04  734          STA  LINE1,Y   ;put on the screen
FB6B:88        735          DEY            ;count it
FB6C:10 F7     736          BPL  LO1       ;and continue till done
FB6E:60        737          RTS
FB6F:          738 ****************************************
FB6F:          739 *
FB6F:          740 * SET PAGE 3 POWER UP BYTE
FB6F:          741 *
FB6F:          742 ****************************************
FB6F:AD F3 03  743 SETPWRC LDA  SOFTEV+1
FB72:49 A5     744          EOR  #$A5
FB74:8D F4 03  745          STA  PWREDUP
FB77:60        746          RTS
FB78:          747 ****************************************
FB78:          748 *
FB78:          749 * CHECK KEYBOARD BEFORE VIDEO OUTPUT
FB78:          750 *
FB78:          751 ****************************************
FB78:AC 00 C0  752 VIDWAIT LDY  KBD
FB7B:C0 93     753          CPY  #$93      ;ctrl-S pressed?
FB7D:D0 0F     754          BNE  VI2       ;no, continue
FB7F:2C 10 C0  755          BIT  KBDSTRB   ;clear keyboard strobe
FB82:AC 00 C0  756 VI1      LDY  KBD       ;wait for next keypress
FB85:10 FB     757          BPL  VI1
FB87:C0 83     758          CPY  #$83      ;ctrl-C?
FB89:F0 72     759          BEQ  VIDOUT    ;yes, it is for Basic
FB8B:8D 10 C0  760          STA  KBDSTRB   ;no, clear strobe
FB8E:D0 6D     761 VI2      BNE  VIDOUT    ;display char in accu
FB90:          762 ****************************************
FB90:C2 E1 F3  763 TITLE    DFB  $C2,$E1,$F3,$E9,$F3,$A0,$B1,$B0,$B8 ;Basis 108
FB93:E9 F3 A0
FB96:B1 B0 B8
FB99:0F 3E 65  764 LOCJMP   DFB  $F,$3E,$65,$19,$57,$9B,$41
FB9C:19 57 9B
FB9F:41
FBA0:          765 ****************************************
FBA0:          766 *
FBA0:          767 * AUDIBLE KEYPRESS FEEDBACK TESTS
FBA0:          768 *
FBA0:          769 ****************************************
FBA0:A0 07     770 LOCAL    LDY  #7        ;# of codes in list
FBA2:D9 09 FB  771 LOCA1    CMP  LOCCHR,Y  ;test against each
FBA5:D0 0B     772          BNE  LOCA2     ;loop if not it
FBA7:A9 FC     773          LDA  #$FC      ;if it is, put -4 on stack
FBA9:48        774          PHA
FBAA:B9 99 FB  775          LDA  LOCJMP,Y  ;pickup matching item from LOCJMP table
FBAD:48        776          PHA            ;stack it
FBAE:A0 18     777          LDY  #$18
FBB0:D0 2D     778          BNE  PIP       ;audible feedback for legel keys
FBB2:88        779 LOCA2    DEY
FBB3:10 ED     780          BPL  LOCA1     ;until last one checked
FBB5:60        781          RTS            ;exit if not on list
FBB6:20 A0 FB  782 JLOCAL   JSR  LOCAL     ;vector
FBB9:20 0C FD  783 RDCHAR1  JSR  RDKEY     ;get keyboard input
FBBC:29 FF     784          AND  #$FF      ;test BIT 7
```

```
FBBE:10 F6      785         BPL  JLOCAL    ;if special key
FBC0:60         786         RTS            ;else return
FBC1:           787 ****************************************
FBC1:           788 *
FBC1:           789 * CALCULATE BASE ADDRESS FOR LINE ON SCREEN
FBC1:           790 *
FBC1:           791 ****************************************
FBC1:48         792 BASCALC PHA            ;save line # in form 000ABCDE
FBC2:4A         793         LSR  A
FBC3:29 03      794         AND  #3
FBC5:09 04      795         ORA  #4        ;for text page 1
FBC7:85 29      796         STA  BASH      ;BASH=000001CD
FBC9:68         797         PLA
FBCA:29 18      798         AND  #$18
FBCC:90 02      799         BCC  BA1
FBCE:09 80      800         ORA  #$80
FBD0:85 28      801 BA1     STA  BASL
FBD2:0A         802         ASL  A
FBD3:0A         803         ASL  A
FBD4:05 28      804         ORA  BASL
FBD6:85 28      805         STA  BASL      ;BASL=EABAB000
FBD8:60         806         RTS
FBD9:           807 ****************************************
FBD9:           808 *
FBD9:           809 * 2 TONE BELL
FBD9:           810 *
FBD9:           811 ****************************************
FBD9:C9 87      812 BELL1   CMP  #$87      ;Ctrl-G?
FBDB:D0 10      813         BNE  NOCTRL    ;no, exit
FBDD:A0 70      814         LDY  #$70      ;new sound
FBDF:98         815 PIP     TYA            ;another sound
FBE0:4A         816         LSR  A
FBE1:4A         817         LSR  A
FBE2:09 07      818         ORA  #7        ;set minimum time
FBE4:20 A8 FC   819         JSR  WAIT      ;wait a little while
FBE7:2C 30 C0   820         BIT  SPKR      ;tick speaker
FBEA:88         821         DEY
FBEB:D0 F2      822         BNE  PIP       ;loop until sound over
FBED:60         823 NOCTRL  RTS
FBEE:           824 ****************************************
FBEE:           825 *
FBEE:           826 * PUT CHARACTER IN SCREEN RAM
FBEE:           827 *
FBEE:           828 ****************************************
FBEE:25 32      829 STORINV AND  INVFLG    ;set inverse bit
FBF0:           830 *
0001:           831 STORADV DO   FORTYCOL
FBF0:A4 24      832         LDY  CH        ;get Horiz index
FBF2:91 28      833         STA  (BASL),Y  ;store char in line
FBF4:           834 *
FBF4:           835         ELSE
FBF4:           836 *
FBF4        S   837         JSR  STOR80    ;do 80 column output
FBF4        S   838         NOP
FBF4:           839         FIN
```

```
FBF4:           840 *
FBF4:           841 ****************************************
FBF4:           842 *
FBF4:           843 * MOVE TO NEXT SCREEN POSITION
FBF4:           844 *
FBF4:           845 ****************************************
FBF4:E6 24      846 ADVANCE INC  CH         ;inc horiz index
FBF6:A4 24      847         LDY  CH
FBF8:C4 21      848         CPY  WNDWDTH    ;wrap at window end
FBFA:B0 66      849         BCS  CR         ;to next line
FBFC:60         850         RTS
FBFD:           851 ****************************************
FBFD:           852 *
FBFD:           853 * TEXT OUTPUT TO VIDEO
FBFD:           854 *
FBFD:           855 ****************************************
FBFD:C9 A0      856 VIDOUT  CMP  #$A0       ;ctrl?
FBFF:B0 ED      857         BCS  STORINV    ;no, diPLAy it normal
FC01:A8         858         TAY             ;or
FC02:10 EC      859         BPL  STORADV    ;normal
FC04:C9 8D      860         CMP  #$8D       ;Return?
FC06:F0 5A      861         BEQ  CR         ;yes, do it
FC08:C9 8A      862         CMP  #$8A       ;linefeed?
FC0A:F0 5A      863         BEQ  LF         ;yes,do it
FC0C:C9 88      864         CMP  #$88       ;Ctrl-H?
FC0E:D0 C9      865         BNE  BELL1      ;no, check for Ctrl-G
FC10:           866 ****************************************
FC10:           867 *
FC10:           868 * MOVE TO PREVIOUS SCREEN POSITION
FC10:           869 *
FC10:           870 ****************************************
FC10:C6 24      871 BS      DEC  CH         ;yes, decrement Horiz index
FC12:10 17      872         BPL  RTS4       ;if in window,leave
FC14:A5 21      873         LDA  WNDWDTH    ;else go to window right edge
FC16:85 24      874         STA  CH
FC18:C6 24      875         DEC  CH         ;minus one
FC1A:A5 22      876 UP      LDA  WNDTOP     ;if at top line
FC1C:C5 25      877         CMP  CV
FC1E:B0 0B      878         BCS  RTS4       ;exit
FC20:C6 25      879         DEC  CV         ;else move up a line
FC22:           880 ****************************************
FC22:           881 *
FC22:           882 * CALCULATE VTAB SCREEN ADDRESS
FC22:           883 *
FC22:           884 ****************************************
FC22:A5 25      885 VTAB    LDA  CV
FC24:20 C1 FB   886 VTABZ   JSR  BASCALC    ;calculate new base address
FC27:           887 *
0001:           888         DO   FORTYCOL
FC27:65 20      889         ADC  WNDLFT     ;offset for window
FC29:85 28      890         STA  BASL
FC2B:           891 *
FC2B:           892         ELSE
FC2B:           893 *
FC2B        S   894         JMP  VTAB80     ;set vert pos for 80 column
```

```
FC2B:           895         FIN
FC2B:           896 *
FC2B:60         897 RTS4    RTS
FC2C:           898 ****************************************
FC2C:           899 *
FC2C:           900 * GET A CHARACTER IN UPPER CASE
FC2C:           901 *
FC2C:           902 ****************************************
FC2C:B9 00 02   903 GETUPCS LDA  IN,Y      ;read char from input buffer
FC2F:C8         904         INY            ;point to next char
FC30:C9 E0      905 UPPER   CMP  #$E0      ;max uppercase ASCII
FC32:90 02      906         BCC  UP1       ;exit if already uppercase
FC34:29 DF      907         AND  #$DF      ;else shift to uppercase
FC36:60         908 UP1     RTS
FC37:00 00 00   909         DFB  0,0,0,0,0,0; fillers
FC3A:00 00 00
FC3D:           910 *
0001:           911         DO   FORTYCOL
FC3D:00 00      912         DFB  0,0       ;fillers
FC3F:           913 *
FC3F:           914         ELSE
FC3F:           915 *
FC3F         S  916         DFB  0,0,0     ;fillers
FC3F:           917         FIN
FC3F:           918 *
FC3F:4C F4 FB   919         JMP  ADVANCE   ;cursor advance vector
FC42:           920 ****************************************
FC42:           921 *
FC42:           922 * CLEAR TO END OF PAGE
FC42:           923 *
FC42:           924 ****************************************
FC42:A4 24      925 CLREOP  LDY  CH        ;get horiz
FC44:A5 25      926         LDA  CV        ;and vert indices
FC46:48         927 CLEOP1  PHA            ;save vert
FC47:20 24 FC   928         JSR  VTABZ     ;get base addr for current line
FC4A:20 9E FC   929         JSR  CLEOLZ    ;clear to end of line
FC4D:A0 00      930         LDY  #0        ;start at left end of subsequent lines
FC4F:68         931         PLA            ;retrieve vert index
FC50:69 00      932         ADC  #0        ;and inc, carry=1 from cleolz
FC52:C5 23      933         CMP  WNDBTM    ;off windowbottom?
FC54:90 F0      934         BCC  CLEOP1    ;if not,do nextline
FC56:B0 CA      935         BCS  VTAB      ;if so set vert index to it
FC58:           936 ****************************************
FC58:           937 *
FC58:           938 * BASIC "HOME"
FC58:           939 *
FC58:           940 ****************************************
FC58:A5 22      941 HOME    LDA  WNDTOP    ;set vert index to windowtop
FC5A:85 25      942         STA  CV
FC5C:A0 00      943         LDY  #0        ;set horiz index to 0
FC5E:84 24      944         STY  CH
FC60:F0 E4      945         BEQ  CLEOP1    ;clear to end of screen (always)
FC62:           946 ****************************************
FC62:           947 *
FC62:           948 * VIDEO CARRIAGE RETURN
```

```
FC62:                 949 *
FC62:                 950 ****************************************
FC62:A9 00            951 CR      LDA  #0         ;set horiz index to 0
FC64:85 24            952         STA  CH
FC66:E6 25            953 LF      INC  CV         ;increment vert index
FC68:A5 25            954         LDA  CV
FC6A:C5 23            955         CMP  WNDBTM     ;off bottom of window?
FC6C:90 B6            956         BCC  VTABZ      ;no, calc new base addr
FC6E:C6 25            957         DEC  CV         ;yes, back up one
FC70:                 958 ****************************************
FC70:                 959 *
FC70:                 960 * SCROLL TEXT SCREEN
FC70:                 961 *
FC70:                 962 ****************************************
FC70:A5 22            963 SCROLL  LDA  WNDTOP     ;get top linenumber
FC72:48               964         PHA             ;save it
FC73:20 24 FC         965         JSR  VTABZ      ;get base addr
FC76:A5 28            966 SC1     LDA  BASL       ;copy it
FC78:85 2A            967         STA  BAS2L
FC7A:A5 29            968         LDA  BASH
FC7C:85 2B            969         STA  BAS2H
FC7E:A4 21            970         LDY  WNDWDTH    ;point to last char in line
FC80:88               971         DEY
FC81:68               972         PLA             ;increment linenumber
FC82:69 01            973         ADC  #1         ;carry=0 from line feed
FC84:C5 23            974         CMP  WNDBTM     ;off bottom?
FC86:B0 0D            975         BCS  SC3        ;yes,finnish
FC88:48               976         PHA             ;no,save it linenumber for next time
FC89:20 24 FC         977         JSR  VTABZ      ;calc base addr
FC8C:                 978 *
0001:                 979         DO   FORTYCOL
FC8C:B1 28            980 SC2     LDA  (BASL),Y   ;get a character
FC8E:91 2A            981         STA  (BAS2L),Y  ;move it up a line
FC90:88               982         DEY             ;index to previous char on line
FC91:10 F9            983         BPL  SC2        ;if line not done do next char
FC93:30 E1            984         BMI  SC1        ;if line done do next line
FC95:                 985 *
FC95:                 986         ELSE
FC95:                 987 *
FC95          S       988         STY  CHY        ;char counter
FC95          S       989         TYA             ;for 80 col scroll
FC95          S       990         JSR  SCRL80     ;move an 80 column line up
FC95          S       991         BCC  SC1        ;do next line (always)
FC95:                 992         FIN
FC95:                 993 *
FC95:A0 00            994 SC3     LDY  #0         ;point to start of line
FC97:20 9E FC         995         JSR  CLEOLZ     ;clear it
FC9A:B0 86            996         BCS  VTAB       ;set (always)
FC9C:                 997 ****************************************
FC9C:                 998 *
FC9C:                 999 * CLEAR TO END OF LINE
FC9C:                1000 *
FC9C:                1001 ****************************************
FC9C:A4 24           1002 CLREOL  LDY  CH
FC9E:                1003 *
```

```
0001:          1004          DO   FORTYCOL
FC9E:A9 A0     1005 CLEOLZ   LDA  #$A0       ;load ablank
FCA0:91 28     1006 CLE1     STA  (BASL),Y   ;store inchar location
FCA2:C8        1007          INY             ;point to next location
FCA3:C4 21     1008          CPY  WNDWDTH    ;last char done?
FCA5:90 F9     1009          BCC  CLE1       ;no clear another
FCA7:60        1010          RTS             ;yes,exit
FCA8:          1011 *
FCA8:          1012          ELSE
FCA8:          1013 *
FCA8         S 1014 CLEOLZ   SEC             ;carry=1 after PLP
FCA8         S 1015          PHP
FCA8         S 1016          JMP  CLEOL80
FCA8         S 1017          DFB  0,0,0,0,0 ;fillers
FCA8:          1018          FIN
FCA8:          1019 *
FCA8:          1020          CHN  MONITOR2
FCA8:             1 ****************************************
FCA8:             2 *
FCA8:             3 * TIME DELAY
FCA8:             4 *
FCA8:             5 ****************************************
FCA8:38           6 WAIT     SEC              ;wait for ord(Accu^2) time
FCA9:48           7 WA1      PHA
FCAA:E9 01        8 WA2      SBC  #1
FCAC:D0 FC        9          BNE  WA2
FCAE:68          10          PLA
FCAF:E9 01       11          SBC  #1
FCB1:D0 F6       12          BNE  WA1
FCB3:60          13          RTS
FCB4:            14 ****************************************
FCB4:            15 *
FCB4:            16 * 16 BIT INCREMENT OF A4
FCB4:            17 *
FCB4:            18 ****************************************
FCB4:E6 42       19 NXTA4    INC  A4L        ;increment low byte
FCB6:D0 02       20          BNE  NXTA1      ;if no overflow, skip ahead
FCB8:E6 43       21          INC  A4H        ;if overflow, increment hi byte
FCBA:A5 3C       22 NXTA1    LDA  A1L        ;compare A1 to A2 (16 bits)
FCBC:C5 3E       23          CMP  A2L
FCBE:A5 3D       24          LDA  A1H
FCC0:E5 3F       25          SBC  A2H        ;carry set if A1 equals or exceeds A2
FCC2:E6 3C       26          INC  A1L        ;increment A1 (16 bits)
FCC4:D0 02       27          BNE  NX1
FCC6:E6 3D       28          INC  A1H
FCC8:60          29 NX1      RTS
FCC9:            30 *
0001:            31          DO   FORTYCOL
FCC9:            32 ****************************************
FCC9:            33 *
FCC9:            34 * WRITE TAPE HEADER
FCC9:            35 *
FCC9:            36 ****************************************
FCC9:A0 4B       37 HEADR    LDY  #$4B       ;write .166*A seconds of header tone
FCCB:20 DB FC    38          JSR  ZERDLY     ;650 usec half cycles
```

```
FCCE:D0 F9      39          BNE  HEADR
FCD0:69 FE      40          ADC  #$FE
FCD2:B0 F5      41          BCS  HEADR      ;until A+C <2
FCD4:A0 21      42          LDY  #$21       ;400 usec break
FCD6:           43 ****************************************
FCD6:           44 *
FCD6:           45 * TAPE DRIVER
FCD6:           46 *
FCD6:           47 ****************************************
FCD6:20 DB FC   48 WRBIT    JSR  ZERDLY     ;2 half cycles of 250 or 500 usec
FCD9:C8         49          INY
FCDA:C8         50          INY
FCDB:88         51 ZERDLY   DEY             ;delay loop
FCDC:D0 FD      52          BNE  ZERDLY     ;until Y=0
FCDE:90 05      53          BCC  WRTAPE     ;0/1 select
FCE0:A0 32      54          LDY  #$32
FCE2:88         55 ONEDLY   DEY
FCE3:D0 FD      56          BNE  ONEDLY
FCE5:AC 20 C0   57 WRTAPE   LDY  TAPEOUT    ;tickle tape out
FCE8:A0 2C      58          LDY  #$2C
FCEA:CA         59          DEX
FCEB:60         60          RTS
FCEC:           61 ****************************************
FCEC:           62 *
FCEC:           63 * READ A BYTE FROM TAPE
FCEC:           64 *
FCEC:           65 ****************************************
FCEC:A2 08      66 RDBYTE   LDX  #8         ;number of bits
FCEE:48         67 RDBYT2   PHA             ;save A
FCEF:20 FA FC   68          JSR  RD2BIT     ;find edge
FCF2:68         69          PLA
FCF3:2A         70          ROL  A
FCF4:A0 3A      71          LDY  #$3A
FCF6:CA         72          DEX             ;count the bit
FCF7:D0 F5      73          BNE  RDBYT2     ;loop if not done
FCF9:60         74          RTS
FCFA:           75 ****************************************
FCFA:           76 *
FCFA:           77 * FIND TAPE TRANSITION
FCFA:           78 *
FCFA:           79 ****************************************
FCFA:20 FD FC   80 RD2BIT   JSR  RDBIT      ;read 1st bit
FCFD:88         81 RDBIT    DEY             ;count loops in Y
FCFE:AD 60 C0   82          LDA  TAPEIN     ;find transition
FD01:45 2F      83          EOR  LASTIN
FD03:10 F8      84          BPL  RDBIT      ;if not, loop
FD05:45 2F      85          EOR  LASTIN     ;if so,
FD07:85 2F      86          STA  LASTIN     ;save new status
FD09:C0 80      87          CPY  #$80       ;set carry
FD0B:60         88          RTS
FD0C:           89 *
FD0C:           90          ELSE
FD0C:           91 *
FD0C:           92 ****************************************
FD0C:           93 *
```

```
FDUC:            94 * 80 COLUMN SCREEN DRIVER
FDOC:            95 *
FDOC:            96 * BANKSWITCH ON ODD/EVEN CHARACTER POSITION
FDOC:            97 *
FDOC:            98 ****************************************
FDOC         S   99 SELBNK  PHA             ;save A
FDOC         S  100         TYA             ;move overall Horiz index to A
FDOC         S  101         LSR  A          ;LSB to carry
FDOC         S  102         STA  VIDBNK     ;400..BFF: dynamic RAM (default)
FDOC         S  103         BCC  SE1        ;skip if Y even
FDOC         S  104         SEI             ;if Y odd, enable interrupts
FDOC         S  105         STA  VIDBNK+1   ;400..BFF: static RAM
FDOC         S  106 SE1     STY  CHY        ;save Yreg in active bank!
FDOC         S  107         TAY             ;for horiz index in active bank
FDOC         S  108         PLA             ;retrieve A
FDOC         S  109         RTS
FDOC         S  110 ****************************************
FDOC         S  111 *
FDOC         S  112 * CLEAR TO END OF LINE
FDOC         S  113 *
FDOC         S  114 ****************************************
FDOC         S  115 CLEOL80 JSR  SELBNK     ;select bank for this char
FDOC         S  116         LDA  #$A0       ;a blank
FDOC         S  117         STA  (BASL),Y   ;clear this char
FDOC         S  118         LDY  CHY        ;last char done?
FDOC         S  119         INY
FDOC         S  120         CPY  WNDWDTH
FDOC         S  121         BCC  CLEOL80    ;no, do another
FDOC         S  122         JMP  VIDPLP     ;yes,restore status
FDOC         S  123 ****************************************
FDOC         S  124 *
FDOC         S  125 * PLOT A POINT
FDOC         S  126 *
FDOC         S  127 ****************************************
FDOC         S  128 PLOT80  LDA  (GBASL),Y ;get current point value
FDOC         S  129         EOR  COLOR      ;set color
FDOC         S  130         AND  MASK       ;mask off
FDOC         S  131         EOR  (GBASL),Y ;put it back
FDOC         S  132         STA  (GBASL),Y
FDOC         S  133         JMP  VIDRTS     ;restore status
FDOC         S  134 ****************************************
FDOC         S  135 *
FDOC         S  136 * READ SCREEN
FDOC         S  137 *
FDOC         S  138 ****************************************
FDOC         S  139 SCRNX   JSR  GBASCALC  ;compute  addr
FDOC         S  140         JSR  SELBNK     ;set proper bank
FDOC         S  141         LDA  (GBASL),Y ;get value
FDOC         S  142         STA  VIDBNK     ;set dynamic bank
FDOC         S  143         LDY  CHY        ;get horiz position
FDOC         S  144         RTS
FDOC         S  145         DFB  0,0,0      ;fillers
FDOC:           146         FIN
FDOC:           147 *
FDOC:           148 ****************************************
```

```
FD0C:           149 *
FD0C:           150 * FLASH CURSOR
FD0C:           151 *
FD0C:           152 ****************************************
0001:           153 RDKEY   DO   FORTYCOL
FD0C:A4 24      154         LDY  CH         ;get horiz index
FD0E:B1 28      155         LDA  (BASL),Y   ;get character from screen RAM
FD10:49 80      156         EOR  #BIT7      ;invert it 7
FD12:91 28      157         STA  (BASL),Y   ;store inverse character on screen
FD14:49 80      158         EOR  #BIT7      ;restore BIT 7 in A
FD16:           159 *
FD16:           160         ELSE
FD16:           161 *
FD16         S  162         NOP             ;fillers
FD16         S  163         NOP
FD16         S  164         NOP
FD16         S  165         NOP
FD16         S  166         NOP
FD16         S  167         NOP
FD16         S  168         NOP
FD16         S  169         JSR  CURS80     ;display cursor
FD16:           170         FIN
FD16:           171 *
FD16:EA         172         NOP             ;fillers
FD17:EA         173         NOP
FD18:           174 *
FD18:6C 38 00   175         JMP  (KSWL)     ;go to User routine
FD1B:           176 ****************************************
FD1B:           177 *
FD1B:           178 * READ KEYBOARD
FD1B:           179 *
FD1B:           180 ****************************************
FD1B:E6 4E      181 KEYIN   INC  RNDL
FD1D:D0 02      182         BNE  KE1        ;cycle random number
FD1F:E6 4F      183         INC  RNDH
FD21:2C 00 C0   184 KE1     BIT  KBD        ;key pressed?
FD24:10 F5      185         BPL  KEYIN      ;no, wait for one.
FD26:           186 *
0001:           187         DO   FORTYCOL
FD26:91 28      188         STA  (BASL),Y   ;put at current screen position
FD28:           189 *
FD28:           190         ELSE
FD28:           191 *
FD28         S  192         JSR  CURS80     ;remove cursor
FD28:           193         FIN
FD28:           194 *
FD28:AD 08 C0   195         LDA  KBDEXTN    ;read function key BIT
FD2B:29 80      196         AND  #BIT7
FD2D:4D 00 C0   197         EOR  KBD        ;merge with ASCII code
FD30:8D 10 C0   198         STA  KBDSTRB    ;save it
FD33:60         199         RTS
FD34:           200 *
0001:           201         DO   FORTYCOL
FD34:00         202         DFB  0          ;filler
FD35:           203         FIN
```

```
FD35:                 204 *
FD35:                 205 ****************************************
FD35:                 206 *
FD35:                 207 * READ INPUT LINE
FD35:                 208 *
FD35:                 209 ****************************************
FD35:4C B9 FB         210 RDCHAR   JMP  RDCHAR1   ;vector
FD38:00 00 00         211          DFB  0,0,0,0,0 ;fillers
FD3B:00 00
FD3D:A5 32            212 NOTCR    LDA  INVFLG    ;get current INVFLG status
FD3F:48               213          PHA            ;save it
FD40:A9 FF            214          LDA  #$FF      ;set INVFLG off
FD42:85 32            215          STA  INVFLG
FD44:BD 00 02         216          LDA  IN,X      ;get char from input buffer
FD47:20 ED FD         217          JSR  COUT      ;output in normal
FD4A:68               218          PLA            ;restore INVFLG
FD4B:85 32            219          STA  INVFLG
FD4D:BD 00 02         220          LDA  IN,X      ;get char again
FD50:C9 88            221          CMP  #$88      ;ctrl-H (backspace)?
FD52:F0 1D            222          BEQ  BCKSPC    ;yes, back up index
FD54:C9 98            223          CMP  #$98      ;ctrl-X?
FD56:F0 0A            224          BEQ  CANCEL    ;yes, cancel input line
FD58:E0 F8            225          CPX  #$F8      ;end of buffer?
FD5A:90 03            226          BCC  NOTCR1    ;no, skip ahead
FD5C:20 3A FF         227          JSR  BELL      ;yes, ring bell
FD5F:E8               228 NOTCR1   INX            ;point to next char
FD60:D0 13            229          BNE  NXTCHAR   ;get it if not too many
FD62:A9 A3            230 CANCEL   LDA  #$A3      ;or flag cancelled input on screen
FD64:20 ED FD         231          JSR  COUT
FD67:20 8E FD         232 GETLNZ   JSR  CROUT     ;output Return
FD6A:A5 33            233 GETLN    LDA  PROMPT    ;output prompt
FD6C:20 ED FD         234          JSR  COUT
FD6F:A2 01            235          LDX  #1        ;point to start of buffer
FD71:8A               236 BCKSPC   TXA            ;if backspaced to 0,
FD72:F0 F3            237          BEQ  GETLNZ    ;start new input line
FD74:CA               238          DEX            ;else back up one char
FD75:20 35 FD         239 NXTCHAR  JSR  RDCHAR    ;get a char from keyboard
FD78:C9 95            240          CMP  #$95      ;ctrl-U (right arrow)?
FD7A:D0 08            241          BNE  ADDINP    ;no, skip ahead
FD7C:                 242 *
0001:                 243          DO   FORTYCOL
FD7C:B1 28            244          LDA  (BASL),Y  ;get input char from screen memory
FD7E:EA               245          NOP            ;filler
FD7F:                 246 *
FD7F:                 247          ELSE
FD7F:                 248 *
FD7F          S       249          JSR  GET80     ;get input char from screen
FD7F:                 250          FIN
FD7F:                 251 *
FD7F:EA               252          NOP            ;filler
FD80:EA               253          NOP
FD81:EA               254          NOP
FD82:EA               255          NOP
FD83:EA               256          NOP
FD84:9D 00 02         257 ADDINP   STA  IN,X      ;put in input buffer
```

```
FD87:C9 8D      258          CMP  #$8D       ;Return? (end of input)
FD89:D0 B2      259          BNE  NOTCR      ;no, get aother
FD8B:20 9C FC   260          JSR  CLREOL     ;clear rest of line
FD8E:A9 8D      261 CROUT    LDA  #$8D       ;echo Return
FD90:D0 5B      262          BNE  COUT       ;and exit via COUT
FD92:           263 ****************************************
FD92:           264 *
FD92:           265 * OUTPUT (A1)
FD92:           266 *
FD92:           267 ****************************************
FD92:A4 3D      268 PRA1     LDY  A1H        ;get A1 contents (16 bit)
FD94:A6 3C      269          LDX  A1L
FD96:20 FA FD   270 PRYX2    JSR  NEWLN      ;start new output line
FD99:20 40 F9   271          JSR  PRNTYX     ;output contents
FD9C:A0 00      272          LDY  #0         ;index to linestart
FD9E:A9 BA      273          LDA  #$BA       ;output a ":"
FDA0:4C ED FD   274          JMP  COUT       ;exit via COUT
FDA3:           275 ****************************************
FDA3:           276 *
FDA3:           277 * HEX MEMORY DUMP
FDA3:           278 *
FDA3:           279 ****************************************
FDA3:A5 3C      280 XAM8     LDA  A1L        ;set starting addr lo byte
FDA5:           281 *
0001:           282          DO   FORTYCOL
FDA5:09 07      283          ORA  #7         ;end at mod7 addr
FDA7:           284 *
FDA7:           285          ELSE
FDA7:           286 *
FDA7         S  287          ORA  #$F        ;end at mod8 addr
FDA7:           288          FIN
FDA7:           289 *
FDA7:85 3E      290          STA  A2L        ;set ending addr in A2
FDA9:A5 3D      291          LDA  A1H
FDAB:85 3F      292          STA  A2H
FDAD:A5 3C      293 MOD8CHK  LDA  A1L        ;retrieve starting addr (lo byte)
FDAF:           294 *
0001:           295          DO   FORTYCOL
FDAF:29 07      296          AND  #7         ;strip hi bit
FDB1:           297 *
FDB1:           298          ELSE
FDB1:           299 *
FDB1         S  300          AND  #$F        ;strip hi bit
FDB1:           301          FIN
FDB1:           302 *
FDB1:D0 03      303          BNE  DATAOUT    ;if lo=0 (each 8 or 16 bytes)
FDB3:20 92 FD   304 XAM      JSR  PRA1       ;output addr 0
FDB6:A9 A0      305 DATAOUT  LDA  #$A0       ;output a blank
FDB8:20 ED FD   306          JSR  COUT
FDBB:B1 3C      307          LDA  (A1L),Y    ;get  byte
FDBD:20 DA FD   308          JSR  PRBYTE     ;and output it
FDC0:20 BA FC   309          JSR  NXTA1      ;increment pointer and test for end
FDC3:90 E8      310          BCC  MOD8CHK    ;loop if not end
FDC5:60         311          RTS             ;exit if end
FDC6:           312 ****************************************
```

```
FDC6:           313 *
FDC6:           314 * MODE CHECK
FDC6:           315 *
FDC6:           316 ****************************************
FDC6:4A         317 XAMPM   LSR  A          ;if LSB A=0
FDC7:90 EA      318         BCC  XAM        ;do memory dump
FDC9:4A         319         LSR  A          ;put original bit 2
FDCA:4A         320         LSR  A          ;into carry
FDCB:A5 3E      321         LDA  A2L        ;get byte
FDCD:90 02      322         BCC  ADD        ;add if original bit 2 was 0
FDCF:49 FF      323         EOR  #$FF       ;else make two's complement
FDD1:65 3C      324 ADD     ADC  A1L        ;add to other byte
FDD3:48         325         PHA             ;save result
FDD4:A9 BD      326         LDA  #$BD       ;output =
FDD6:20 ED FD   327         JSR  COUT
FDD9:68         328         PLA             ;retrieve value
FDDA:48         329 PRBYTE  PHA             ;keeping it on stack
FDDB:4A         330         LSR  A          ;move high nibble down
FDDC:4A         331         LSR  A
FDDD:4A         332         LSR  A
FDDE:4A         333         LSR  A
FDDF:20 E5 FD   334         JSR  PRHEXZ     ;output char
FDE2:68         335         PLA             ;retrieve value again
FDE3:29 0F      336 PRHEX   AND  #$F        ;strip off hi bit
FDE5:09 B0      337 PRHEXZ  ORA  #$B0       ;convert to ASCII
FDE7:C9 BA      338         CMP  #$BA       ;>9?
FDE9:90 02      339         BCC  COUT       ;no, output it
FDEB:69 06      340         ADC  #6         ;"A".."F"
FDED:           341 ****************************************
FDED:           342 *
FDED:           343 * ASCII OUTPUT
FDED:           344 *
FDED:           345 ****************************************
FDED:6C 36 00   346 COUT    JMP  (CSWL)     ;convert to user output routine
FDF0:48         347 COUT1   PHA             ;save A
FDF1:84 35      348         STY  YSAV1      ;and Y
FDF3:20 78 FB   349         JSR  VIDWAIT    ;output it
FDF6:A4 35      350         LDY  YSAV1      ;restore Y
FDF8:68         351         PLA             ;and A
FDF9:60         352         RTS             ;and return
FDFA:           353 ****************************************
FDFA:           354 *
FDFA:           355 * START NEW OUTPUT LINE
FDFA:           356 *
FDFA:           357 ****************************************
FDFA:20 8E FD   358 NEWLN   JSR  CROUT      ;output Return
FDFD:A9 A0      359         LDA  #$A0       ;and blank
FDFF:D0 EC      360         BNE  COUT       ;exit through COUT (always)
FE01:           361 ****************************************
FE01:           362 *
FE01:           363 * MONITOR COMMAND PAGE
FE01:           364 *
FE01:           365 ****************************************
FE01:C6 34      366 BL1     DEC  YSAV       ;if YSAVE=1
FE03:F0 9E      367         BEQ  XAM8       ;do memory dump
```

```
FE05:CA        368 BLANK   DEX             ;if X doesn't = 1
FE06:D0 16     369         BNE  SETMDZ     ;set mode
FE08:C9 BA     370         CMP  #$BA       ;if not store mode,
FE0A:D0 BA     371         BNE  XAMPM      ;do dump, add, or subtract
FE0C:          372 ****************************************
FE0C:          373 *
FE0C:          374 * MONITOR STORE
FE0C:          375 *
FE0C:          376 ****************************************
FE0C:85 31     377 STOR    STA  MODE       ;set mode
FE0E:A5 3E     378         LDA  A2L        ;get byte
FE10:91 40     379         STA  (A3L),Y    ;store it
FE12:E6 40     380         INC  A3L        ;adjust pointer (16 bits)
FE14:D0 02     381         BNE  ST1
FE16:E6 41     382         INC  A3H
FE18:60        383 ST1     RTS             ;and exit
FE19:A4 34     384 SETMODE LDY  YSAV       ;retrieve pointer
FE1B:B9 FF 01  385         LDA  IN-1,Y     ;get command (:, +, -, or .)
FE1E:85 31     386 SETMDZ  STA  MODE       ;set mode
FE20:60        387         RTS             ;and exit
FE21:          388 ****************************************
FE21:          389 *
FE21:          390 * COPY A2 TO A4, A5 (16 BITS)
FE21:          391 *
FE21:          392 ****************************************
FE21:A2 01     393 LT      LDX  #1         ;index
FE23:B5 3E     394 LT1     LDA  A2L,X      ;move lo byte
FE25:95 42     395         STA  A4L,X
FE27:CA        396         DEX             ;adjust index
FE28:10 F9     397         BPL  LT1        ;move hi byte
FE2A:60        398         RTS             ;then exit
FE2B:00        399         DFB  0          ;filler
FE2C:          400 ****************************************
FE2C:          401 *
FE2C:          402 * MEMORY BLOCK MOVE
FE2C:          403 *
FE2C:          404 ****************************************
FE2C:B1 3C     405 MOVE    LDA  (A1L),Y    ;get byte
FE2E:91 42     406         STA  (A4L),Y    ;move (copy) it
FE30:20 B4 FC  407         JSR  NXTA4      ;adj source pointer (16 bit) and test for end
FE33:90 F7     408         BCC  MOVE       ;if not end, do again
FE35:60        409         RTS             ;else exit
FE36:          410 ****************************************
FE36:          411 *
FE36:          412 * MEMORY BLOCK COMPARE
FE36:          413 *
FE36:          414 ****************************************
FE36:B1 3C     415 VERIFY  LDA  (A1L),Y    ;get byte from range 1
FE38:D1 42     416         CMP  (A4L),Y    ;compare to matching byte in range 2
FE3A:F0 17     417         BEQ  VE1        ;if match, skip ahead
FE3C:20 92 FD  418         JSR  PRA1       ;else print addr in range 1
FE3F:B1 3C     419         LDA  (A1L),Y    ;retrieve offending byte from range 1
FE41:20 DA FD  420         JSR  PRBYTE     ;output it
FE44:A9 BC     421         LDA  #$BC       ;output <
FE46:20 ED FD  422         JSR  COUT
```

```
FE49:A9 BE     423          LDA  #$BE       ;output >
FE4B:20 ED FD  424          JSR  COUT
FE4E:B1 42     425          LDA  (A4L),Y    ;retrieve offending byte from range 2
FE50:20 DA FD  426          JSR  PRBYTE     ;output it
FE53:20 B4 FC  427 VE1      JSR  NXTA4      ;increment pointer (16 bits) and test for end
FE56:90 DE     428          BCC  VERIFY     ;if not end, do again
FE58:60        429          RTS             ;else exit
FE59:6C F2 03  430 BASCONT  JMP  (SOFTEV)   ;vector for Basic "CONT"
FE5C:4C 00 E0  431 XBASIC   JMP  BASIC      ;vector for Basic with no program
FE5F:00        432          DFB  0          ;filler
FE60:          433 ****************************************
FE60:          434 *
FE60:          435 * MONITOR "LIST"
FE60:          436 *
FE60:          437 ****************************************
FE60:20 75 FE  438 LIST     JSR  A1PC       ;move A1 (16 bits) to PC
FE63:20 D0 F8  439 LI1      JSR  INSTDSP    ;disasm instruction
FE66:20 53 F9  440          JSR  PCADJ      ;adj PC to next instruction
FE69:85 3A     441          STA  PCL
FE6B:84 3B     442          STY  PCH
FE6D:C5 3E     443          CMP  A2L        ;test for list end
FE6F:98        444          TYA
FE70:E5 3F     445          SBC  A2H
FE72:90 EF     446          BCC  LI1        ;if not, do again
FE74:60        447          RTS             ;if so, return
FE75:          448 ****************************************
FE75:          449 *
FE75:          450 * SET PC FROM A1
FE75:          451 *
FE75:          452 ****************************************
FE75:8A        453 A1PC     TXA             ;if X=0
FE76:F0 07     454          BEQ  A1PC2      ;exit
FE78:B5 3C     455 A1PC1    LDA  A1L,X      ;move lo byte
FE7A:95 3A     456          STA  PCL,X
FE7C:CA        457          DEX             ;adjust index
FE7D:10 F9     458          BPL  A1PC1      ;do again for hi byte
FE7F:60        459 A1PC2    RTS             ;then exit
FE80:          460 ****************************************
FE80:          461 *
FE80:          462 * INVERSE
FE80:          463 *
FE80:          464 ****************************************
FE80:A0 7F     465 SETINV   LDY  #$7F       ;true status
FE82:D0 02     466          BNE  SETIFLG    ;(always)
FE84:          467 ****************************************
FE84:          468 *
FE84:          469 * NORMAL
FE84:          470 *
FE84:          471 ****************************************
FE84:A0 FF     472 SETNORM  LDY  #$FF       ;false status
FE86:84 32     473 SETIFLG  STY  INVFLG     ;set it
FE88:60        474          RTS
FE89:          475 ****************************************
FE89:          476 *
FE89:          477 * SET I/O PORT
```

```
FE89:           478 *
FE89:           479 ****************************************
FE89:A9 00      480 SETKBD  LDA  #0         ;default=0 (keyboard)
FE8B:85 3E      481 INPORT  STA  A2L        ;IN#n entry, save n
FE8D:A2 38      482 INPRT   LDX  #KSWL      ;input vector
FE8F:A0 1B      483         LDY  #$1B       ;keyin lo byte
FE91:D0 08      484         BNE  IOPRT      ;(always)
FE93:A9 00      485 SETVID  LDA  #0         ;default output = 0 (video)
FE95:85 3E      486 OUTPORT STA  A2L        ;PR#n entry, save n
FE97:A2 36      487 OUTPRT  LDX  #CSWL      ;output vector
FE99:A0 F0      488         LDY  #$F0       ;cout1 lo byte
FE9B:A5 3E      489 IOPRT   LDA  A2L        ;retrieve n
FE9D:29 07      490         AND  #7         ;only slots 1..7 are legal
FE9F:F0 06      491         BEQ  IOPRT1     ;slot 0 has no I/O ROM space
FEA1:09 C0      492         ORA  #IOPAGE    ;$C0 to point to appropriate I/O ROM
FEA3:A0 00      493         LDY  #0
FEA5:F0 02      494         BEQ  IOPRT2     ;(always)
FEA7:A9 FD      495 IOPRT1  LDA  #$FD       ;KEYIN/COUT1 hi byte
FEA9:94 00      496 IOPRT2  STY  LOC0,X     ;set vector
FEAB:95 01      497         STA  LOC1,X
FEAD:A5 3E      498         LDA  A2L        ;if slot in [8..15.] then vector =Cs08
FEAF:29 08      499         AND  #8         ;else vector=Cs00
FEB1:15 00      500         ORA  LOC0,X
FEB3:95 00      501         STA  LOC0,X
FEB5:60         502         RTS
FEB6:           503 ****************************************
FEB6:20 75 FE   504 GO      JSR  A1PC       ;set PC from A1
FEB9:20 3F FF   505         JSR  RESTORE    ;restore registers
FEBC:6C 3A 00   506         JMP  (PCL)      ;go where told to
FEBF:           507 ****************************************
FEBF:4C D7 FA   508 REGZ    JMP  REGDSP     ;register display vectors
FEC2:           509 ****************************************
FEC2:60         510 TRACE   RTS             ;trace is no more
FEC3:EA         511         NOP             ;fillers
FEC4:           512 ****************************************
FEC4:60         513 STEPZ   RTS             ;STEP is gone too
FEC5:EA         514         NOP
FEC6:EA         515         NOP
FEC7:EA         516         NOP
FEC8:EA         517         NOP
FEC9:EA         518         NOP
FECA:           519 ****************************************
FECA:4C F8 03   520 USR     JMP  USRADR     ;user vector
FECD:           521 *
0001:           522         DO   FORTYCOL
FECD:           523 ****************************************
FECD:           524 *
FECD:           525 * TAPE WRITE
FECD:           526 *
FECD:           527 ****************************************
FECD:A9 40      528 WRITE   LDA  #$40       ;write 10 sec header
FECF:20 C9 FC   529         JSR  HEADR
FED2:A0 27      530         LDY  #$27
FED4:A2 00      531 WR1     LDX  #0
FED6:41 3C      532         EOR  (A1L,X)    ;for Checksum
```

```
FED8:48          533          PHA             ;save
FED9:A1 3C       534          LDA  (A1L,X)    ;get a byte
FEDB:20 ED FE    535          JSR  WRBYTE     ;send to tape
FEDE:20 BA FC    536          JSR  NXTA1      ;update pointer
FEE1:A0 1D       537          LDY  #$1D
FEE3:68          538          PLA
FEE4:90 EE       539          BCC  WR1
FEE6:A0 22       540          LDY  #$22
FEE8:20 ED FE    541          JSR  WRBYTE     ;put byte on tape
FEEB:F0 4D       542          BEQ  BELL       ;make noise
FEED:A2 10       543 WRBYTE   LDX  #$10
FEEF:0A          544 WRBYT2   ASL  A          ;hi bit to carry
FEF0:20 D6 FC    545          JSR  WRBIT      ;write a bit
FEF3:D0 FA       546          BNE  WRBYT2     ;repeat 'til byte done
FEF5:60          547          RTS             ;then exit
FEF6             548 *
FEF6             549          ELSE
FEF6             550 *
FEF6         S   551 WRITE    RTS             ;no tape out
FEF6         S   552 ****************************************
FEF6         S   553 *
FEF6         S   554 * PUT CHARACTER IN SCREEN RAM
FEF6         S   555 *
FEF6         S   556 ****************************************
FEF6         S   557 STOR80   PHP             ;save status
FEF6         S   558          LDY  CH         ;get horiz index
FEF6         S   559          JSR  SELBNK     ;select appropriate bank
FEF6         S   560          JMP  STRTS      ;go store char
FEF6         S   561 ****************************************
FEF6         S   562 *
FEF6         S   563 * BLINK CURSOR
FEF6         S   564 *
FEF6         S   565 ****************************************
FEF6         S   566 CURS80   PHP             ;save status
FEF6         S   567          LDY  CH         ;get horiz index
FEF6         S   568          JSR  SELBNK     ;select appropriate bank
FEF6         S   569          LDA  (BASL),Y   ;get char
FEF6         S   570          EOR  #BIT7      ;invert inverse bit
FEF6         S   571 STRTS    STA  (BASL),Y   ;write char,
FEF6         S   572 VIDRTS   LDY  CHY        ;restore Yreg,
FEF6         S   573 VIDPLP   STA  VIDBNK     ;restore memory bank (softswitch)
FEF6         S   574          PLP             ;restore Iflag
FEF6         S   575          RTS             ;exit
FEF6         S   576 ****************************************
FEF6         S   577 *
FEF6         S   578 * CALCULATE VTAB SCREEN ADDRESS
FEF6         S   579 *
FEF6         S   580 ****************************************
FEF6         S   581 VTAB80   LDA  WNDLFT     ;get window offset
FEF6         S   582          LSR  A          ; /2 for split bank
FEF6         S   583          CLC
FEF6         S   584          ADC  BASL
FEF6         S   585          STA  BASL
FEF6         S   586          RTS             ;exit
FEF6         S   587          DFB  0,0        ;filler
```

```
FEF6:                 588          FIN
FEF6:                 589 *
FEF6:                 590 ****************************************
FEF6:                 591 *
FEF6:                 592 * COMMAND PROCESSOR ENTRY POINT
FEF6:                 593 *
FEF6:                 594 ****************************************
FEF6:20 01 FE         595 CRMON    JSR  BL1
FEF9:68               596          PLA             ;adjust stac
FEFA:68               597          PLA
FEFB:D0 6C            598          BNE  MONZ       ;enter command processor
FEFD:                 599 *
FEFD:                 600 ****************************************
FEFD:                 601 *
FEFD:                 602 * TAPE READ
FEFD:                 603 *
FEFD:                 604 ****************************************
0001:                 605          DO   FORTYCOL
FEFD:20 FA FC         606 READ     JSR  RD2BIT     ;find edge
FF00:A9 16            607          LDA  #$16       ;3.5 sec delay
FF02:20 C9 FC         608          JSR  HEADR
FF05:85 2E            609          STA  CHKSUM     ;initialize to FF
FF07:20 FA FC         610          JSR  RD2BIT     ;find edge
FF0A:A0 24            611 RD2      LDY  #$24       ;look for sync
FF0C:20 FD FC         612          JSR  RDBIT      ;skip second sync
FF0F:B0 F9            613          BCS  RD2        ;loop until found
FF11:20 FD FC         614          JSR  RDBIT      ;skip second sync
FF14:A0 3B            615          LDY  #$3B       ;index for 0/1 detect
FF16:20 EC FC         616 RD3      JSR  RDBYTE     ;read a byte
FF19:81 3C            617          STA  (A1L,X)    ;save it
FF1B:45 2E            618          EOR  CHKSUM     ;maintain checksum
FF1D:85 2E            619          STA  CHKSUM
FF1F:20 BA FC         620          JSR  NXTA1      ;update memory pointer,check for end
FF22:A0 35            621          LDY  #$35       ;for 0/1 detect
FF24:90 F0            622          BCC  RD3        ;loop until all bytes read
FF26:20 EC FC         623          JSR  RDBYTE     ;read checksum from tape
FF29:C5 2E            624          CMP  CHKSUM     ;same as read checksum?
FF2B:F0 0D            625          BEQ  BELL       ;yes, ring bell, (exit there)
FF2D:                 626 *
FF2D:                 627          ELSE
FF2D:                 628 *
FF2D        S         629 READ     RTS  ;no        tape input!
FF2D        S         630 ****************************************
FF2D        S         631 *
FF2D        S         632 * SCREEN READ
FF2D        S         633 *
FF2D        S         634 ****************************************
FF2D        S         635 GET80    PHP             ;save status
FF2D        S         636          LDY  CH         ;get horiz index
FF2D        S         637          JSR  SELBNK     ;select appropriate bank
FF2D        S         638          LDA  (BASL),Y   ;get char
FF2D        S         639          JMP  VIDRTS     ;exit
FF2D        S         640 ****************************************
FF2D        S         641 *
FF2D        S         642 * FAST SCROLL LINE WITHOUT JSR SELBNK
```

```
FF2D        S  643 *
FF2D        S  644 ***************************************
FF2D        S  645 SCRL80  PHP            ;save status
FF2D        S  646         SEI            ;disable interrupts (400..BFF is switched!)
FF2D        S  647         LSR  A         ;LSB to carry
FF2D        S  648         TAY            ;save A
FF2D        S  649         BCC  EVENCHR   ;first time odd or even?
FF2D        S  650 ODDCHR  STA  VIDBNK+1  ;static RAM on
FF2D        S  651         LDA  (BASL),Y  ;copy in static RAM
FF2D        S  652         STA  (BAS2L),Y ;up a line
FF2D        S  653         STA  VIDBNK    ;static RAM off
FF2D        S  654         DEC  CHY       ;adjust vert index
FF2D        S  655         BMI  SCRLEX    ;done?
FF2D        S  656 EVENCHR LDA  (BASL),Y  ;copy in dynamic RAM (even chars)
FF2D        S  657         STA  (BAS2L),Y ;up a line
FF2D        S  658         DEY            ;counter
FF2D        S  659         DEC  CHY       ;adjust vert index
FF2D        S  660         BPL  ODDCHR    ;more to scroll?
FF2D        S  661 SCRLEX  PLP            ;no restore status
FF2D        S  662         CLC
FF2D        S  663         RTS            ;and return
FF2D        S  664         DFB  0,0       ;filler
FF2D:          665         FIN
FF2D:          666 *
FF2D:          667 ***************************************
FF2D:          668 *
FF2D:          669 * OUTPUT ERROR MESSAGE
FF2D:          670 *
FF2D:          671 ***************************************
FF2D:A9 C5     672 PRERR   LDA  #$C5      ;"E"
FF2F:20 ED FD  673         JSR  COUT
FF32:A9 D2     674         LDA  #$D2      ;"R" (twice)
FF34:20 ED FD  675         JSR  COUT
FF37:20 ED FD  676         JSR  COUT
FF3A:A9 87     677 BELL    LDA  #$87      ;audible signal too
FF3C:4C ED FD  678         JMP  COUT
FF3F:          679 ***************************************
FF3F:          680 *
FF3F:          681 * RESTORE REGISTERS
FF3F:          682 *
FF3F:          683 ***************************************
FF3F:A5 48     684 RESTORE LDA  STATUS    ;put status
FF41:48        685         PHA            ;on stack
FF42:A5 45     686         LDA  ACC       ;restore A
FF44:A6 46     687 RESTR1  LDX  XREG      ;  "     X
FF46:A4 47     688         LDY  YREG      ;  "     Y
FF48:28        689         PLP            ;  "     status
FF49:60        690         RTS
FF4A:          691 ***************************************
FF4A:          692 *
FF4A:          693 * SAVE REGISTERS
FF4A:          694 *
FF4A:          695 ***************************************
FF4A:85 45     696 SAVE    STA  ACC       ;save A
FF4C:86 46     697 SAV1    STX  XREG      ;  "  X
```

```
FF4E:84 47      698          STY  YREG       ;  "  Y
FF50:08         699          PHP             ;  "   status to stack
FF51:68         700          PLA             ;then to
FF52:85 48      701          STA  STATUS     ;memory
FF54:BA         702          TSX
FF55:86 49      703          STX  SPNT       ;save the old stack pointer value!
FF57:D8         704          CLD             ;clear decimal value
FF58:60         705 IORTS    RTS             ;used by slot ROM
FF59:           706 ****************************************
FF59:           707 *
FF59:           708 * RESET MONITOR
FF59:           709 *
FF59:           710 ****************************************
FF59:20 84 FE   711 OLDRST   JSR  SETNORM    ;normal video
FF5C:20 2F FB   712          JSR  INIT       ;set video mode and IFLG
FF5F:20 93 FE   713          JSR  SETVID     ;set default output
FF62:20 89 FE   714          JSR  SETKBD     ;and input
FF65:D8         715 MON      CLD             ;clear decimal mode
FF66:20 3A FF   716          JSR  BELL       ;tell someone
FF69:           717 ****************************************
FF69:           718 *
FF69:           719 * MONITOR COMMAND PROCESSOR
FF69:           720 *
FF69:           721 ****************************************
FF69:A9 AA      722 MONZ     LDA  #$AA       ;monitor prompt (*)
FF6B:85 33      723          STA  PROMPT
FF6D:20 67 FD   724          JSR  GETLNZ     ;get command input line
FF70:20 C7 FF   725          JSR  ZMODE      ;set mode 0
FF73:20 A7 FF   726 NXTITM   JSR  GETNUM     ;get non-hex chars
FF76:84 34      727          STY  YSAV       ;save place in input line
FF78:A0 17      728          LDY  #$17       ;command table length
FF7A:88         729 CHRSRCH  DEY             ;adjust table index
FF7B:30 E8      730          BMI  MON        ;if not found, reenter monitor
FF7D:D9 CC FF   731          CMP  CHRTBL,Y   ;check table entry against command
FF80:D0 F8      732          BNE  CHRSRCH    ;loop until found or end of table
FF82:20 BE FF   733          JSR  TOSUB      ;go to desired routine
FF85:A4 34      734          LDY  YSAV       ;on return, restore Y
FF87:4C 73 FF   735          JMP  NXTITM     ;interrupt next command
FF8A:           736 ****************************************
FF8A:           737 *
FF8A:           738 * PUT HEX IN A2
FF8A:           739 *
FF8A:           740 ****************************************
FF8A:A2 03      741 DIG      LDX  #3
FF8C:0A         742          ASL  A          ;rotate digit up
FF8D:0A         743          ASL  A
FF8E:0A         744          ASL  A
FF8F:0A         745          ASL  A
FF90:0A         746 NXTBIT   ASL  A
FF91:26 3E      747          ROL  A2L        ;corresponding A2 rotation
FF93:26 3F      748          ROL  A2H
FF95:CA         749          DEX
FF96:10 F8      750          BPL  NXTBIT     ;until transferred
FF98:A5 31      751 NXTBAS   LDA  MODE       ;if mode not 0
FF9A:D0 06      752          BNE  NXTBS2     ;skip ahead
```

```
FF9C:B5 3F      753          LDA  A2H,X      ;else copy A2H
FF9E:95 3D      754          STA  A1H,X      ;into a1H
FFA0:95 41      755          STA  A3H,X      ;and A3H
FFA2:E8         756 NXTBS2   INX             ;index up
FFA3:F0 F3      757          BEQ  NXTBAS     ;until all copied
FFA5:D0 06      758          BNE  NXTCHR     ;then return to parser
FFA7:           759 ****************************************
FFA7:           760 *
FFA7:           761 * COMMAND LINE PARSER
FFA7:           762 *
FFA7:           763 ****************************************
FFA7:A2 00      764 GETNUM   LDX  #0         ;initialize index
FFA9:86 3E      765          STX  A2L        ;and A2
FFAB:86 3F      766          STX  A2H
FFAD:20 2C FC   767 NXTCHR   JSR  GETUPCS    ;get uppercase char
FFB0:49 B0      768          EOR  #$B0       ;check if hex digit
FFB2:C9 0A      769          CMP  #$A
FFB4:90 D4      770          BCC  DIG        ;if so, handle it
FFB6:69 88      771          ADC  #$88
FFB8:C9 FA      772          CMP  #$FA
FFBA:B0 CE      773          BCS  DIG        ;and next
FFBC:60         774          RTS             ;return with non-decimal char in A
FFBD:00         775          DFB  0
FFBE:           776 ****************************************
FFBE:           777 *
FFBE:           778 * ROUTE TO MONITOR ROUTINES
FFBE:           779 *
FFBE:           780 ****************************************
FFBE:A9 FE      781 TOSUB    LDA  #$FE       ;command page for high byte
FFC0:48         782          PHA             ;push on stack
FFC1:B9 E3 FF   783          LDA  SUBTBL,Y   ;set lo byte
FFC4:48         784          PHA             ;push it too
FFC5:A5 31      785          LDA  MODE       ;get mode in A
FFC7:A0 00      786 ZMODE    LDY  #0         ;clear mode byte
FFC9:84 31      787          STY  MODE
FFCB:60         788          RTS             ;jump to desired routine (from stack)
FFCC:           789 ****************************************
FFCC:           790 *
FFCC:           791 * COMMAND CHARACTERS
FFCC:           792 *
FFCC:           793 ****************************************
FFCC:EA         794 CHRTBL   DFB  $EA        ;Q warm start Basic
FFCD:BB         795          DFB  $BB        ;CTRL-B cold start Basic
FFCE:EE         796          DFB  $EE        ;U user vector jump
FFCF:98         797          DFB  $98        ;? display registers
FFD0:EF         798          DFB  $EF        ;V verify memory
FFD1:06         799          DFB  $06        ;M move memory
FFD2:04         800          DFB  $04        ;K input vector
FFD3:E9         801          DFB  $E9        ;P output vector
FFD4:EC         802          DFB  $EC        ;(S) step inactive
FFD5:ED         803          DFB  $ED        ;(T) trace inactive
FFD6:A6         804          DFB  $A6        ;- subtract
FFD7:A4         805          DFB  $A4        ;+ add
FFD8:07         806          DFB  $07        ;N normal
FFD9:02         807          DFB  $02        ;I inverse
```

```
FFDA:05         808          DFB  $05        ;L list memory (disassm)
FFDB:EB         809          DFB  $EB        ;R read from tape
FFDC:F0         810          DFB  $F0        ;W write to tape
FFDD:00         811          DFB  $00        ;G jump to address
FFDE:93         812          DFB  $93        ;: memory store
FFDF:A7         813          DFB  $A7        ;. delimiter
FFE0:95         814          DFB  $95        ;< transfer operator
FFE1:C6         815          DFB  $C6        ;CTRL-M carriage return
FFE2:99         816          DFB  $99        ;BLANK space
FFE3:           817 ****************************************
FFE3:           818 *
FFE3:           819 * COMMAND POINTERS
FFE3:           820 *
FFE3:           821 ****************************************
FFE3:58         822 SUBTBL   DFB  $58        ;Basic warm JMP $3F2 is moved
FFE4:5B         823          DFB  $5B        ;Basic cold JMP $E000 is moved
FFE5:C9         824          DFB  $C9        ;user JMP $3F8
FFE6:BE         825          DFB  $BE        ;register display
FFE7:35         826          DFB  $35        ;verify
FFE8:2B         827          DFB  $2B        ;move
FFE9:8C         828          DFB  $8C        ;input vector
FFEA:96         829          DFB  $96        ;output vector
FFEB:C3         830          DFB  $C3        ;(step)
FFEC:C1         831          DFB  $C1        ;(trace)
FFED:18         832          DFB  $18        ;-
FFEE:18         833          DFB  $18        ;+
FFEF:83         834          DFB  $83        ;normal
FFF0:7F         835          DFB  $7F        ;inverse
FFF1:5F         836          DFB  $5F        ;list is moved!
FFF2:FC         837          DFB  $FC        ;read
FFF3:CC         838          DFB  $CC        ;write
FFF4:B5         839          DFB  $B5        ;go
FFF5:18         840          DFB  $18        ;:
FFF6:18         841          DFB  $18        ;.
FFF7:20         842          DFB  $20        ;<
FFF8:F5         843          DFB  $F5        ;<cr>
FFF9:04         844          DFB  $04        ;<space>
FFFA:FB 03      845          DW   NMI
FFFC:62 FA      846          DW   RESET
FFFE:40 FA      847          DW   IRQ

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

SYMBOL TABLE SORTED BY SYMBOL

3D A1H	3C A1L	FE78 A1PC1	FE7F A1PC2
FE75 A1PC	3F A2H	3E A2L	41 A3H
40 A3L	43 A4H	42 A4L	45 ACC
FDD1 ADD	FD84 ADDINP	FBF4 ADVANCE	?03F5 AMPERV
FBD0 BA1	2B BAS2H	2A BAS2L	FBC1 BASCALC
?FE59 BASCONT	29 BASH	?E003 BASIC2	E000 BASIC
28 BASL	FD71 BCKSPC	FF3A BELL	FBD9 BELL1
80 BIT7	FE01 BL1	?FE05 BLANK	FA4C BREAK
03F0 BRKV	?FC10 BS	FD62 CANCEL	00 CFONT1
01 CFONT2	00 CFONT3	24 CH	F9B4 CHAR1
F9BA CHAR2	2E CHKSUM	?C006 CHRBAS	C002 CHRGEN0
C004 CHRGEN1	C000 CHRINV	FF7A CHRSRCH	FFCC CHRTBL
F83C CL1	FCA0 CLE1	FC9E CLEOLZ	FC46 CLEOP1
FC9C CLREOL	?FC42 CLREOP	CFFF CLRROM	F838 CLRSC2
?F832 CLRSCR	F836 CLRTOP	30 COLOR	FDED COUT
?FDF0 COUT1	?FEF6 CRMON	FD8E CROUT	FC62 CR
? 37 CSWH	36 CSWL	25 CV	FDB6 DATAOUT
FF8A DIG	FB01 DISKID	F8A5 ERR	FA9B FIXSEV
F962 FMT1	F9A6 FMT2	2E FORMAT	01 FORTYCOL
F856 GB1	F847 GBASCALC	27 GBASH	26 GBASL
F8A9 GETFMT	?FD6A GETLN	FD67 GETLNZ	FFA7 GETNUM
FC2C GETUPCS	?FEB6 GO	2C H2	FCC9 HEADR
F81C HL1	?F819 HLINE	FC58 HOME	F89B IEVEN
?FE8B INPORT	0200 IN	FB2F INIT	?FE8D INPRT
F882 INSDS1	?F88C INSDS2	F8D0 INSTDSP	32 INVFLG
?C000 IOARD	C0 IOPAGE	FE9B IOPRT	FEA7 IOPRT1
FEA9 IOPRT2	?FF58 IORTS	03FE IRQLOC	FA40 IRQ
FBB6 JLOCAL	C008 KBDEXTN	C010 KBDSTRB	C000 KBD
FD21 KE1	FD1B KEYIN	? 39 KSWH	38 KSWL
2F LASTIN	2F LENGTH	FC66 LF	FE63 LI1
0400 LINE1	?FE60 LIST	2C LMNEM	FB65 LO1
00 LOC0	01 LOC1	FBA2 LOCA1	FBB2 LOCA2
FBA0 LOCAL	FB09 LOCCHR	FB99 LOCJMP	FB60 LOGO
C056 LORES	C054 LOWSCR	FE23 LT1	?FE21 LT
2E MASK	C052 MIXCLR	F9C0 MNEML	FA00 MNEMR
F88E MNNDX1	F8C2 MNNDX2	F8C9 MNNDX3	FDAD MOD8CHK
31 MODE	FF65 MON	FF69 MONZ	FE2C MOVE
07F8 MSLOT	FDFA NEWLN	?FA6F NEWMON	03FB NMI
FBED NOCTRL	FAA3 NOFIX	FD5F NOTCR1	FD3D NOTCR
FCC8 NX1	FCBA NXTA1	FCB4 NXTA4	FF98 NXTBAS
FF90 NXTBIT	FFA2 NXTBS2	FD75 NXTCHAR	FFAD NXTCHR
?F85F NXTCOL	FF73 NXTITM	FA59 OLDBRK	?FF59 OLDRST
FCE2 ONEDLY	?FE95 OUTPORT	?FE97 OUTPRT	C064 PADDL0
?F954 PCADJ2	F956 PCADJ3	F953 PCADJ	F95C PCADJ4
3B PCH	3A PCL	FBDF PIP	F80C PL1
F80E PLOT1	F800 PLOT	F8F5 PR1	F8F9 PR2
FD92 PRA1	F910 PRADR1	F914 PRADR2	F926 PRADR3
F92A PRADR4	F930 PRADR5	F94A PRBL2	?F94C PRBL3
F948 PRBLNK	FDDA PRBYTE	FB25 PRE1	?FB1E PREAD
?FF2D PRERR	?FDE3 PRHEX	FDE5 PRHEXZ	?F941 PRNTAX
F8DB PRNTBL	F8D4 PRNTOP	?F944 PRNTX	F940 PRNTYX
33 PROMPT	FD96 PRYX2	C070 PTRIG	FAFD PWRCON
03F4 PWREDUP	FAA6 PWRUP	FCFA RD2BIT	FF0A RD2
FF16 RD3	FCFD RDBIT	FCEE RDBYT2	FCEC RDBYTE
FD35 RDCHAR	FBB9 RDCHAR1	FD0C RDKEY	?FEFD READ

SYMBOL TABLE SORTED BY SYMBOL

```
 FAD7 REGDSP        ?FEBF REGZ         ?F938 RELADR        FA62 RESET
 FF3F RESTORE       ?FF44 RESTR1        FAE4 RG1           FADA RGDSP1
   2D RMNEM           4F RNDH             4E RNDL          FB19 RTBL
 F831 RTS1          F961 RTS2           FC2B RTS4          FB2E RTSX
 FF4C SAV1          ?FF4A SAVE          FC76 SC1           FC8C SC2
 FC95 SC3           F87F SCR1          ?F871 SCRN          F879 SCRN2
?FC70 SCROLL        ?F864 SETCOL       ?FB40 SETGR         FE86 SETIFLG
?FE80 SETINV         FE89 SETKBD        FE1E SETMDZ       ?FE19 SETMODE
 FE84 SETNORM       ?FAA9 SETPG3        FAAB SETPLP       ?FB6F SETPWRC
?FB39 SETTXT         FE93 SETVID        FB4B SETWND         FAC7 SL1
 FABA SLOOP         03F2 SOFTEV         C030 SPKR            49 SPNT
 FE18 ST1             48 STATUS        ?FEC4 STEPZ        ?FE0C STOR
 FBF0 STORADV       FBEE STORINV        FFE3 SUBTBL       ?FB5B TABV
 C060 TAPEIN        C020 TAPEOUT        FB90 TITLE         FFBE TOSUB
?FEC2 TRACE         ?C058 TTLOUT0      ?C05A TTLOUT1       ?C05C TTLOUT2
?C05E TTLOUT3        C050 TXTCLR        FC36 UP1          ?FC1A UP
?FC30 UPPER         03F8 USRADR        ?FECA USR            2D V2
 FE53 VE1           FE36 VERIFY         FB82 VI1           FB8E VI2
 C00A VID40         ?C00B VID80        ?C00C VIDBNK        FBFD VIDOUT
 FB78 VIDWAIT        F826 VLINEZ        F828 VLINE         FC22 VTAB
 FC24 VTABZ          FCA9 WA1           FCAA WA2           FCA8 WAIT
   28 WIDTH            23 WNDBTM          20 WNDLFT          22 WNDTOP
   21 WNDWDTH        FED4 WR1           FCD6 WRBIT         FEEF WRBYT2
 FEED WRBYTE        ?FECD WRITE         FCE5 WRTAPE        FDA3 XAM8
 FDC6 XAMPM          FDB3 XAM          ?FE5C XBASIC          46 XREG
   47 YREG             34 YSAV            35 YSAV1         FCDB ZERDLY
 FFC7 ZMODE
```

SYMBOL TABLE SORTED BY ADDRESS

```
   00 LOCO          00 CFONT1        00 CFONT3        01 FORTYCOL
   01 CFONT2        01 LOC1          20 WNDLFT        21 WNDWDTH
   22 WNDTOP        23 WNDBTM        24 CH            25 CV
   26 GBASL         27 GBASH         28 BASL          28 WIDTH
   29 BASH          2A BAS2L         2B BAS2H         2C LMNEM
   2C H2            2D RMNEM         2D V2            2E MASK
   2E CHKSUM        2E FORMAT        2F LASTIN        2F LENGTH
   30 COLOR         31 MODE          32 INVFLG        33 PROMPT
   34 YSAV          35 YSAV1         36 CSWL        ? 37 CSWH
   38 KSWL        ? 39 KSWH          3A PCL           3B PCH
   3C A1L           3D A1H           3E A2L           3F A2H
   40 A3L           41 A3H           42 A4L           43 A4H
   45 ACC           46 XREG          47 YREG          48 STATUS
   49 SPNT          4E RNDL          4F RNDH          80 BIT7
   C0 IOPAGE      0200 IN          03F0 BRKV        03F2 SOFTEV
 03F4 PWREDUP   ?03F5 AMPERV      03F8 USRADR      03FB NMI
 03FE IRQLOC      0400 LINE1       07F8 MSLOT       C000 CHRINV
?C000 IOARD       C000 KBD         C002 CHRGEN0     C004 CHRGEN1
?C006 CHRBAS      C008 KBDEXTN     C00A VID40     ?C00B VID80
?C00C VIDBNK      C010 KBDSTRB     C020 TAPEOUT     C030 SPKR
 C050 TXTCLR      C052 MIXCLR      C054 LOWSCR      C056 LORES
?C058 TTLOUT0   ?C05A TTLOUT1    ?C05C TTLOUT2    ?C05E TTLOUT3
 C060 TAPEIN      C064 PADDL0      C070 PTRIG       CFFF CLRROM
 E000 BASIC     ?E003 BASIC2       F800 PLOT        F80C PL1
 F80E PLOT1     ?F819 HLINE        F81C HL1         F826 VLINEZ
 F828 VLINE       F831 RTS1      ?F832 CLRSCR       F836 CLRTOP
 F838 CLRSC2      F83C CL1         F847 GBASCALC    F856 GB1
?F85F NXTCOL    ?F864 SETCOL     ?F871 SCRN         F879 SCRN2
 F87F SCR1        F882 INSDS1    ?F88C INSDS2       F89B IEVEN
 F8A5 ERR         F8A9 GETFMT      F8BE MNNDX1      F8C2 MNNDX2
 F8C9 MNNDX3      F8D0 INSTDSP     F8D4 PRNTOP      F8DB PRNTBL
 F8F5 PR1         F8F9 PR2         F910 PRADR1      F914 PRADR2
 F926 PRADR3      F92A PRADR4      F930 PRADR5    ?F938 RELADR
 F940 PRNTYX    ?F941 PRNTAX     ?F944 PRNTX        F948 PRBLNK
 F94A PRBL2     ?F94C PRBL3        F953 PCADJ     ?F954 PCADJ2
 F956 PCADJ3      F95C PCADJ4      F961 RTS2        F962 FMT1
 F9A6 FMT2        F9B4 CHAR1       F9BA CHAR2       F9C0 MNEML
 FA00 MNEMR       FA40 IRQ         FA4C BREAK       FA59 OLDBRK
 FA62 RESET     ?FA6F NEWMON       FA9B FIXSEV      FAA3 NOFIX
 FAA6 PWRUP     ?FAA9 SETPG3       FAAB SETPLP      FABA SLOOP
 FAC7 SL1         FAD7 REGDSP      FADA RGDSP1      FAE4 RG1
 FAFD PWRCON      FB01 DISKID      FB09 LOCCHR      FB19 RTBL
?FB1E PREAD       FB25 PRE1        FB2E RTSX        FB2F INIT
?FB39 SETTXT    ?FB40 SETGR        FB4B SETWND    ?FB5B TABV
 FB60 LOGO        FB65 LO1       ?FB6F SETPWRC      FB78 VIDWAIT
 FB82 VI1         FB8E VI2         FB90 TITLE       FB99 LOCJMP
 FBA0 LOCAL       FBA2 LOCA1       FBB2 LOCA2       FBB6 JLOCAL
 FBB9 RDCHAR1     FBC1 BASCALC     FBD0 BA1         FBD9 BELL1
 FBDF PIP         FBED NOCTRL      FBEE STORINV     FBF0 STORADV
 FBF4 ADVANCE     FBFD VIDOUT    ?FC10 BS         ?FC1A UP
 FC22 VTAB        FC24 VTABZ       FC2B RTS4        FC2C GETUPCS
?FC30 UPPER       FC36 UP1       ?FC42 CLREOP       FC46 CLEOP1
 FC58 HOME        FC62 CR          FC66 LF        ?FC70 SCROLL
 FC76 SC1         FC8C SC2         FC95 SC3         FC9C CLREOL
 FC9E CLEOLZ      FCA0 CLE1        FCA8 WAIT        FCA9 WA1
```

SYMBOL TABLE SORTED BY ADDRESS

FCAA WA2	FCB4 NXTA4	FCBA NXTA1	FCC8 NX1
FCC9 HEADR	FCD6 WRBIT	FCDB ZERDLY	FCE2 ONEDLY
FCE5 WRTAPE	FCEC RDBYTE	FCEE RDBYT2	FCFA RD2BIT
FCFD RDBIT	FD0C RDKEY	FD1B KEYIN	FD21 KE1
FD35 RDCHAR	FD3D NOTCR	FD5F NOTCR1	FD62 CANCEL
FD67 GETLNZ	?FD6A GETLN	FD71 BCKSPC	FD75 NXTCHAR
FD84 ADDINP	FD8E CROUT	FD92 PRA1	FD96 PRYX2
FDA3 XAM8	FDAD MOD8CHK	FDB3 XAM	FDB6 DATAOUT
FDC6 XAMPM	FDD1 ADD	FDDA PRBYTE	?FDE3 PRHEX
FDE5 PRHEXZ	FDED COUT	?FDF0 COUT1	FDFA NEWLN
FE01 BL1	?FE05 BLANK	?FE0C STOR	FE18 ST1
?FE19 SETMODE	FE1E SETMDZ	?FE21 LT	FE23 LT1
FE2C MOVE	FE36 VERIFY	FE53 VE1	?FE59 BASCONT
?FE5C XBASIC	?FE60 LIST	FE63 LI1	FE75 A1PC
FE78 A1PC1	FE7F A1PC2	?FE80 SETINV	FE84 SETNORM
FE86 SETIFLG	FE89 SETKBD	?FE8B INPORT	?FE8D INPRT
FE93 SETVID	?FE95 OUTPORT	?FE97 OUTPRT	FE9B IOPRT
FEA7 IOPRT1	FEA9 IOPRT2	?FEB6 GO	?FEBF REGZ
?FEC2 TRACE	?FEC4 STEPZ	?FECA USR	?FECD WRITE
FED4 WR1	FEED WRBYTE	FEEF WRBYT2	?FEF6 CRMON
?FEFD READ	FF0A RD2	FF16 RD3	?FF2D PRERR
FF3A BELL	FF3F RESTORE	?FF44 RESTR1	?FF4A SAVE
FF4C SAV1	?FF58 IORTS	?FF59 OLDRST	FF65 MON
FF69 MONZ	FF73 NXTITM	FF7A CHRSRCH	FF8A DIG
FF90 NXTBIT	FF98 NXTBAS	FFA2 NXTBS2	FFA7 GETNUM
FFAD NXTCHR	FFBE TOSUB	FFC7 ZMODE	FFCC CHRTBL
FFE3 SUBTBL			

```
0000:            2 ****************************************
0000:            3 *                                      *
0000:            4 *     BASIS 108 MONITOR                *
0000:            5 *                                      *
0000:            6 *     25-46  and  25-86                *
0000:            7 *                                      *
0000:            8 *     (C) COPYRIGHT 1982               *
0000:            9 *     BASIS Incorporated               *
0000:           10 *                                      *
0000:           11 *     ALL RIGHTS RESERVED              *
0000:           12 *                                      *
0000:           13 ****************************************
0000:           14 ;
0000:           15 FORTYCOL EQU 0          ;0 = 80 column display
0000:           16 ;                        1 = 40 column display
0000:           17 CFONT1  EQU  0          ;1 = use this character set
0001:           18 CFONT2  EQU  1          ;0 = don't use this character set
0000:           19 CFONT3  EQU  0          ;only 1 character set can be selected at a time
0000:           20 ;
0000:           21 LOC0    EQU  0          ;used for bootstrap
0001:           22 LOC1    EQU  1

0000:          107 *
0000:          108         DO   FORTYCOL
0000       S   109 WIDTH   EQU  40
0000       S   110 CHKSUM  EQU  $2E        ;for tape read
0000       S   111 TAPEOUT EQU  $C020
0000       S   112 TAPEIN  EQU  $C060
0000       S   113         SBTL "BASIS    108 MONITOR 25-46"
0000       S   114 *
0000:          115         ELSE
0000:          116 *
0050:          117 WIDTH   EQU  80
0479:          118 CHY     EQU  $479       ;80 column video driver
0000:          120         FIN


F80E:          136 *
0000:          137 PLOT1   DO   FORTYCOL
F80E       S   138         LDA  (GBASL),Y ;get data
F80E       S   139         EOR  COLOR     ;set color
F80E       S   140         AND  MASK      ;pick hi or lo nibble
F80E       S   141         EOR  (GBASL),Y ;restore other nibble
F80E       S   142         STA  (GBASL),Y ;put it back
F80E       S   143         RTS
F80E       S   144 *
F80E:          145         ELSE
F80E:          146 *
F80E:08        147         PHP             ;save LSB
F80F:20 C9 FC  148         JSR  SELBNK     ;set bank
F812:4C ED FC  149         JMP  PLOT80     ;plot point
F815:00 00 00  150         DFB  0,0,0,0    ;fillers
F818:00
F819:          151         FIN
F819:          152 *
```

```
F873:           241 *
0000:           242         DO   FORTYCOL
F873         S  243         JSR  GBASCALC  ;set address
F873         S  244         LDA  (GBASL),Y ;get data
F873         S  245 *
F873:           246         ELSE
F873:           247 *
F873:20 FA FC   248         JSR  SCRNX     ;read point status
F876:EA         249         NOP             ;filler
F877:EA         250         NOP
F878:           251         FIN
F878:           252 *

FA73:           560 *
0000:           561         DO   FORTYCOL
FA73         S  562         STA  VID40     ;set 40 column display mode
FA73         S  563 *
FA73:           564         ELSE
FA73:           565 *
FA73:8D 0B C0   566         STA  VID80     ;set 80 column display mode
FA76:           567         FIN
FA76:           568 *

FB10:           658 *
0000:           659         DO   FORTYCOL
FB10         S  660         STA  VID40     ;video switch
FB10         S  661 *
FB10:           662         ELSE
FB10:           663 *
FB10:8D 0B C0   664         STA  VID80     ;video switch
FB13:           665         FIN
FB13:           666 *

FBF0:           830 *
0000:           831 STORADV DO   FORTYCOL
FBF0         S  832         LDY  CH        ;get Horiz index
FBF0         S  833         STA  (BASL),Y  ;store char in line
FBF0         S  834 *
FBF0:           835         ELSE
FBF0:           836 *
FBF0:20 CE FE   837         JSR  STOR80    ;do 80 column output
FBF3:EA         838         NOP
FBF4:           839         FIN
FBF4:           840 *

FC27:           887 *
0000:           888         DO   FORTYCOL
FC27         S  889         ADC  WNDLFT    ;offset for window
FC27         S  890         STA  BASL
FC27         S  891 *
FC27:           892         ELSE
FC27:           893 *
FC27:4C EB FE   894         JMP  VTAB80    ;set vert pos for 80 column
FC2A:           895         FIN
FC2A:           896 *
```

```
FC3C:           910 *
0000:           911         DO    FORTYCOL
FC3C         S  912         DFB   0,0         ;fillers
FC3C         S  913 *
FC3C:           914         ELSE
FC3C:           915 *
FC3C:00 00 00   916         DFB   0,0,0       ;fillers
FC3F:           917         FIN
FC3F:           918 *


FC8C:           978 *
0000:           979         DO    FORTYCOL
FC8C         S  980 SC2     LDA   (BASL),Y    ;get a character
FC8C         S  981         STA   (BAS2L),Y   ;move it up a line
FC8C         S  982         DEY   ;index      to previous char on line
FC8C         S  983         BPL   SC2         ;if line not done do next char
FC8C         S  984         BMI   SC1         ;if line done do next line
FC8C         S  985 *
FC8C:           986         ELSE
FC8C:           987 *
FC8C:8C 79 04   988         STY   CHY         ;char counter
FC8F:98         989         TYA               ;for 80 col scroll
FC90:20 09 FF   990         JSR   SCRL80      ;move an 80 column line up
FC93:90 E1      991         BCC   SC1         ;do next line (always)
FC95:           992         FIN
FC95:           993 *


FC9E:          1003 *
0000:          1004         DO    FORTYCOL
FC9E         S 1005 CLEOLZ  LDA   #$A0        ;load ablank
FC9E         S 1006 CLE1    STA   (BASL),Y    ;store inchar location
FC9E         S 1007         INY   ;point      to next location
FC9E         S 1008         CPY   WNDWDTH     ;last char done?
FC9E         S 1009         BCC   CLE1        ;no clear another
FC9E         S 1010         RTS   ;yes,exit
FC9E         S 1011 *
FC9E:          1012         ELSE
FC9E:          1013 *
FC9E:38        1014 CLEOLZ  SEC               ;carry=1 after PLP
FC9F:08        1015         PHP
FCA0:4C DB FC  1016         JMP   CLEOL80
FCA3:00 00 00  1017         DFB   0,0,0,0,0   ;fillers
FCA6:00 00
FCA8:          1018         FIN
FCA8:          1019 *
```

```
FCC9:           30 *
0000:           31          DO   FORTYCOL
FCC9:           32 ****************************************
FCC9:           33 *
FCC9:           34 * WRITE TAPE HEADER
FCC9:           35 *
FCC9:           36 ****************************************
FCC9        S   37 HEADR   LDY  #$4B       ;write .166*A seconds of header tone
FCC9        S   38         JSR  ZERDLY     ;650 usec half cycles
FCC9        S   39         BNE  HEADR
FCC9        S   40         ADC  #$FE
FCC9        S   41         BCS  HEADR      ;until A+C <2
FCC9        S   42         LDY  #$21       ;400 usec break
FCC9        S   43 ****************************************
FCC9        S   44 *
FCC9        S   45 * TAPE DRIVER
FCC9        S   46 *
FCC9        S   47 ****************************************
FCC9        S   48 WRBIT   JSR  ZERDLY     ;2 half cycles of 250 or 500 usec
FCC9        S   49         INY
FCC9        S   50         INY
FCC9        S   51 ZERDLY  DEY  ;delay     loop
FCC9        S   52         BNE  ZERDLY     ;until Y=0
FCC9        S   53         BCC  WRTAPE     ;0/1 select
FCC9        S   54         LDY  #$32
FCC9        S   55 ONEDLY  DEY
FCC9        S   56         BNE  ONEDLY
FCC9        S   57 WRTAPE  LDY  TAPEOUT    ;tickle tape out
FCC9        S   58         LDY  #$2C
FCC9        S   59         DEX
FCC9        S   60         RTS
FCC9        S   61 ****************************************
FCC9        S   62 *
FCC9        S   63 * READ A BYTE FROM TAPE
FCC9        S   64 *
FCC9        S   65 ****************************************
FCC9        S   66 RDBYTE  LDX  #8         ;number of bits
FCC9        S   67 RDBYT2  PHA  ;save      A
FCC9        S   68         JSR  RD2BIT     ;find edge
FCC9        S   69         PLA
FCC9        S   70         ROL  A
FCC9        S   71         LDY  #$3A
FCC9        S   72         DEX  ;count     the bit
FCC9        S   73         BNE  RDBYT2     ;loop if not done
FCC9        S   74         RTS
FCC9        S   75 ****************************************
FCC9        S   76 *
FCC9        S   77 * FIND TAPE TRANSITION
FCC9        S   78 *
FCC9        S   79 ****************************************
FCC9        S   80 RD2BIT  JSR  RDBIT      ;read 1st bit
FCC9        S   81 RDBIT   DEY  ;count     loops in Y
FCC9        S   82         LDA  TAPEIN     ;find transition
FCC9        S   83         EOR  LASTIN
```

```
FCC9        S   84          BPL  RDBIT     ;if not, loop
FCC9        S   85          EOR  LASTIN    ;if so,
FCC9        S   86          STA  LASTIN    ;save new status
FCC9        S   87          CPY  #$80      ;set carry
FCC9        S   88          RTS
FCC9        S   89 *
FCC9:           90          ELSE
FCC9:           91 *
FCC9:           92 ****************************************
FCC9:           93 *
FCC9:           94 * 80 COLUMN SCREEN DRIVER
FCC9:           95 *
FCC9:           96 * BANKSWITCH ON ODD/EVEN CHARACTER POSITION
FCC9:           97 *
FCC9:           98 ****************************************
FCC9:48         99 SELBNK  PHA              ;save A
FCCA:98        100         TYA              ;move overall Horiz index to A
FCCB:4A        101         LSR  A           ;LSB to carry
FCCC:8D 0C C0  102         STA  VIDBNK      ;400..BFF: dynamic RAM (default)
FCCF:90 04     103         BCC  SE1         ;skip if Y even
FCD1:78        104         SEI              ;if Y odd, enable interrupts
FCD2:8D 0D C0  105         STA  VIDBNK+1    ;400..BFF: static RAM
FCD5:8C 79 04  106 SE1     STY  CHY         ;save Yreg in active bank!
FCD8:A8        107         TAY              ;for horiz index in active bank
FCD9:68        108         PLA              ;retrieve A
FCDA:60        109         RTS
FCDB:          110 ****************************************
FCDB:          111 *
FCDB:          112 * CLEAR TO END OF LINE
FCDB:          113 *
FCDB:          114 ****************************************
FCDB:20 C9 FC  115 CLEOL80 JSR  SELBNK      ;select bank for this char
FCDE:A9 A0     116         LDA  #$A0        ;a blank
FCE0:91 28     117         STA  (BASL),Y    ;clear this char
FCE2:AC 79 04  118         LDY  CHY         ;last char done?
FCE5:C8        119         INY
FCE6:C4 21     120         CPY  WNDWDTH
FCE8:90 F1     121         BCC  CLEOL80     ;no, do another
FCEA:4C E6 FE  122         JMP  VIDPLP      ;yes,restore status
FCED:          123 ****************************************
FCED:          124 *
FCED:          125 * PLOT A POINT
FCED:          126 *
FCED:          127 ****************************************
FCED:B1 26     128 PLOT80  LDA  (GBASL),Y ;get current point value
FCEF:45 30     129         EOR  COLOR     ;set color
FCF1:25 2E     130         AND  MASK      ;mask off
FCF3:51 26     131         EOR  (GBASL),Y ;put it back
FCF5:91 26     132         STA  (GBASL),Y
FCF7:4C E3 FE  133         JMP  VIDRTS    ;restore status
FCFA:          134 ****************************************
FCFA:          135 *
FCFA:          136 * READ SCREEN
FCFA:          137 *
FCFA:          138 ****************************************
```

```
FCFA:20 47 F8  139 SCRNX   JSR  GBASCALC  ;compute  addr
FCFD:20 C9 FC  140         JSR  SELBNK    ;set proper bank
FD00:B1 26     141         LDA  (GBASL),Y ;get value
FD02:8D 0C C0  142         STA  VIDBNK    ;set dynamic bank
FD05:AC 79 04  143         LDY  CHY       ;get horiz position
FD08:60        144         RTS
FD09:00 00 00  145         DFB  0,0,0     ;fillers
FD0C:          146         FIN
FD0C:          147 *


FD0C:          148 ****************************************
FD0C:          149 *
FD0C:          150 * FLASH CURSOR
FD0C:          151 *
FD0C:          152 ****************************************
0000:          153 RDKEY   DO   FORTYCOL
FD0C        S  154         LDY  CH        ;get horiz index
FD0C        S  155         LDA  (BASL),Y  ;get character from screen RAM
FD0C        S  156         EOR  #BIT7     ;invert it 7
FD0C        S  157         STA  (BASL),Y  ;store inverse character on screen
FD0C        S  158         EOR  #BIT7     ;restore BIT 7 in A
FD0C        S  159 *
FD0C:          160         ELSE
FD0C:          161 *
FD0C:EA        162         NOP             ;fillers
FD0D:EA        163         NOP
FD0E:EA        164         NOP
FD0F:EA        165         NOP
FD10:EA        166         NOP
FD11:EA        167         NOP
FD12:EA        168         NOP
FD13:20 D7 FE  169         JSR  CURS80    ;display cursor
FD16:          170         FIN
FD16:          171 *


FD26:          186 *
0000:          187         DO   FORTYCOL
FD26        S  188         STA  (BASL),Y  ;put at current screen position
FD26        S  189 *
FD26:          190         ELSE
FD26:          191 *
FD26:20 D7 FE  192         JSR  CURS80    ;remove cursor
FD29:          193         FIN
FD29:          194 *


FD35:          200 *
0000:          201         DO   FORTYCOL
FD35        S  202         DFB  0         ;filler
FD35:          203         FIN
FD35:          204 *
```

```
FD7C:           242 *
0000:           243         DO   FORTYCOL
FD7C        S   244         LDA  (BASL),Y  ;get input char from screen memory
FD7C        S   245         NOP  ;filler
FD7C        S   246 *
FD7C:           247         ELSE
FD7C:           248 *
FD7C:20 FE FE   249         JSR  GET80     ;get input char from screen
FD7F:           250         FIN
FD7F:           251 *

FDA5:           281 *
0000:           282         DO   FORTYCOL
FDA5        S   283         ORA  #7        ;end at mod7 addr
FDA5        S   284 *
FDA5:           285         ELSE
FDA5:           286 *
FDA5:09 0F      287         ORA  #$F       ;end at mod8 addr
FDA7:           288         FIN
FDA7:           289 *

FDAF:           294 *
0000:           295         DO   FORTYCOL
FDAF        S   296         AND  #7        ;strip hi bit
FDAF        S   297 *
FDAF:           298         ELSE
FDAF:           299 *
FDAF:29 0F      300         AND  #$F       ;strip hi bit
FDB1:           301         FIN
FDB1:           302 *


FECD:           521 *
0000:           522         DO   FORTYCOL
FECD:           523 ****************************************
FECD:           524 *
FECD:           525 * TAPE WRITE
FECD:           526 *
FECD:           527 ****************************************
FECD        S   528 WRITE   LDA  #$40      ;write 10 sec header
FECD        S   529         JSR  HEADR
FECD        S   530         LDY  #$27
FECD        S   531 WR1     LDX  #0
FECD        S   532         EOR  (A1L,X)   ;for Checksum
FECD        S   533         PHA  ;save
FECD        S   534         LDA  (A1L,X)   ;get a byte
FECD        S   535         JSR  WRBYTE    ;send to tape
FECD        S   536         JSR  NXTA1     ;update pointer
```

```
FECD        S  537          LDY  #$1D
FECD        S  538          PLA
FECD        S  539          BCC  WR1
FECD        S  540          LDY  #$22
FECD        S  541          JSR  WRBYTE    ;put byte on tape
FECD        S  542          BEQ  BELL      ;make noise
FECD        S  543 WRBYTE   LDX  #$10
FECD        S  544 WRBYT2   ASL  A         ;hi bit to carry
FECD        S  545          JSR  WRBIT     ;write a bit
FECD        S  546          BNE  WRBYT2    ;repeat 'til byte done
FECD        S  547          RTS  ;then     exit
FECD        S  548 *
FECD:          549          ELSE
FECD:          550 *
FECD:60        551 WRITE    RTS            ;no tape out
FECE:          552 ****************************************
FECE:          553 *
FECE:          554 * PUT CHARACTER IN SCREEN RAM
FECE:          555 *
FECE:          556 ****************************************
FECE:08        557 STOR80   PHP            ;save status
FECF:A4 24     558          LDY  CH        ;get horiz index
FED1:20 C9 FC  559          JSR  SELBNK    ;select appropriate bank
FED4:4C E1 FE  560          JMP  STRTS     ;go store char
FED7:          561 ****************************************
FED7:          562 *
FED7:          563 * BLINK CURSOR
FED7:          564 *
FED7:          565 ****************************************
FED7:08        566 CURS80   PHP            ;save status
FED8:A4 24     567          LDY  CH        ;get horiz index
FEDA:20 C9 FC  568          JSR  SELBNK    ;select appropriate bank
FEDD:B1 28     569          LDA  (BASL),Y  ;get char
FEDF:49 80     570          EOR  #BIT7     ;invert inverse bit
FEE1:91 28     571 STRTS    STA  (BASL),Y  ;write char,
FEE3:AC 79 04  572 VIDRTS   LDY  CHY       ;restore Yreg,
FEE6:8D 0C C0  573 VIDPLP   STA  VIDBNK    ;restore memory bank (softswitch)
FEE9:28        574          PLP            ;restore Iflag
FEEA:60        575          RTS            ;exit
FEEB:          576 ****************************************
FEEB:          577 *
FEEB:          578 * CALCULATE VTAB SCREEN ADDRESS
FEEB:          579 *
FEEB:          580 ****************************************
FEEB:A5 20     581 VTAB80   LDA  WNDLFT    ;get window offset
FEED:4A        582          LSR  A         ; /2 for split bank
FEEE:18        583          CLC
FEEF:65 28     584          ADC  BASL
FEF1:85 28     585          STA  BASL
FEF3:60        586          RTS            ;exit
FEF4:00 00     587          DFB  0,0       ;filler
FEF6:          588          FIN
FEF6:          589 *
FEF6:          590 ****************************************
```

```
FEFD:           599 *
FEFD:           600 ****************************************
FEFD:           601 *
FEFD:           602 * TAPE READ
FEFD:           603 *
FEFD:           604 ****************************************
0000:           605          DO   FORTYCOL
FEFD         S  606 READ     JSR  RD2BIT     ;find edge
FEFD         S  607          LDA  #$16       ;3.5 sec delay
FEFD         S  608          JSR  HEADR
FEFD         S  609          STA  CHKSUM     ;initialize to FF
FEFD         S  610          JSR  RD2BIT     ;find edge
FEFD         S  611 RD2      LDY  #$24       ;look for sync
FEFD         S  612          JSR  RDBIT      ;skip second sync
FEFD         S  613          BCS  RD2        ;loop until found
FEFD         S  614          JSR  RDBIT      ;skip second sync
FEFD         S  615          LDY  #$3B       ;index for 0/1 detect
FEFD         S  616 RD3      JSR  RDBYTE     ;read a byte
FEFD         S  617          STA  (A1L,X)    ;save it
FEFD         S  618          EOR  CHKSUM     ;maintain checksum
FEFD         S  619          STA  CHKSUM
FEFD         S  620          JSR  NXTA1      ;update memory pointer,check for end
FEFD         S  621          LDY  #$35       ;for 0/1 detect
FEFD         S  622          BCC  RD3        ;loop until all bytes read
FEFD         S  623          JSR  RDBYTE     ;read checksum from tape
FEFD         S  624          CMP  CHKSUM     ;same as read checksum?
FEFD         S  625          BEQ  BELL       ;yes, ring bell, (exit there)
FEFD         S  626 *
FEFD:           627          ELSE
FEFD:           628 *
FEFD:60         629 READ     RTS             ;no tape input!
FEFE:           630 ****************************************
FEFE:           631 *
FEFE:           632 * SCREEN READ
FEFE:           633 *
FEFE:           634 ****************************************
FEFE:08         635 GET80    PHP              ;save status
FEFF:A4 24      636          LDY  CH         ;get horiz index
FF01:20 C9 FC   637          JSR  SELBNK     ;select appropriate bank
FF04:B1 28      638          LDA  (BASL),Y   ;get char
FF06:4C E3 FE   639          JMP  VIDRTS     ;exit
FF09:           640 ****************************************
FF09:           641 *
FF09:           642 * FAST SCROLL LINE WITHOUT JSR SELBNK
FF09:           643 *
FF09:           644 ****************************************
FF09:08         645 SCRL80   PHP              ;save status
FF0A:78         646          SEI              ;disable interrupts (400..BFF is switched!)
```

```
FF0B:4A        647          LSR  A          ;LSB to carry
FF0C:A8        648          TAY              ;save A
FF0D:90 0F     649          BCC  EVENCHR     ;first time odd or even?
FF0F:8D 0D C0  650 ODDCHR   STA  VIDBNK+1   ;static RAM on
FF12:B1 28     651          LDA  (BASL),Y    ;copy in static RAM
FF14:91 2A     652          STA  (BAS2L),Y ;up a line
FF16:8D 0C C0  653          STA  VIDBNK      ;static RAM off
FF19:CE 79 04  654          DEC  CHY         ;adjust vert index
FF1C:30 0A     655          BMI  SCRLEX      ;done?
FF1E:B1 28     656 EVENCHR  LDA  (BASL),Y    ;copy in dynamic RAM (even chars)
FF20:91 2A     657          STA  (BAS2L),Y ;up a line
FF22:88        658          DEY              ;counter
FF23:CE 79 04  659          DEC  CHY         ;adjust vert index
FF26:10 E7     660          BPL  ODDCHR      ;more to scroll?
FF28:28        661 SCRLEX   PLP              ;no restore status
FF29:18        662          CLC
FF2A:60        663          RTS              ;and return
FF2B:00 00     664          DFB  0,0         ;filler
FF2D:          665          FIN
FF2D:          666 *
```

SYMBOL TABLE SORTED BY SYMBOL

3D A1H	3C A1L	FE78 A1PC1	FE7F A1PC2
FE75 A1PC	3F A2H	3E A2L	41 A3H
40 A3L	43 A4H	42 A4L	45 ACC
FD84 ADDINP	FDD1 ADD	FBF4 ADVANCE	?03F5 AMPERV
FBDO BA1	2B BAS2H	2A BAS2L	FBC1 BASCALC
?FE59 BASCONT	29 BASH	?E003 BASIC2	E000 BASIC
28 BASL	FD71 BCKSPC	FF3A BELL	FBD9 BELL1
80 BIT7	FE01 BL1	?FE05 BLANK	FA4C BREAK
03F0 BRKV	?FC10 BS	FD62 CANCEL	00 CFONT1
01 CFONT2	00 CFONT3	F9B4 CHAR1	?C006 CHRBAS
C002 CHRGEN0	C004 CHRGEN1	C000 CHRINV	FF7A CHRSRCH
24 CH	F9BA CHAR2	FFCC CHRTBL	0479 CHY
F83C CL1	FCDB CLEOL80	FC9E CLEOLZ	FC46 CLEOP1
FC9C CLREOL	?FC42 CLREOP	CFFF CLRROM	F838 CLRSC2
?F832 CLRSCR	F836 CLRTOP	30 COLOR	FDED COUT
?FDF0 COUT1	?FEF6 CRMON	FD8E CROUT	FC62 CR
? 37 CSWH	36 CSWL	FED7 CURS80	25 CV
FDB6 DATAOUT	FF8A DIG	FB01 DISKID	F8A5 ERR
FF1E EVENCHR	FA9B FIXSEV	F962 FMT1	F9A6 FMT2
2E FORMAT	00 FORTYCOL	F856 GB1	F847 GBASCALC
27 GBASH	26 GBASL	FEFE GET80	F8A9 GETFMT
?FD6A GETLN	FD67 GETLNZ	FFA7 GETNUM	FC2B GETUPCS
?FEB6 GO	2C H2	F81C HL1	?F819 HLINE
FC58 HOME	F89B IEVEN	FB2F INIT	?FE8B INPORT
?FE8D INPRT	?F88C INSDS2	0200 IN	F882 INSDS1
F8D0 INSTDSP	32 INVFLG	?C000 IOARD	C0 IOPAGE
FE9B IOPRT	FEA7 IOPRT1	FEA9 IOPRT2	?FF58 IORTS
FA40 IRQ	03FE IRQLOC	FBB6 JLOCAL	C008 KBDEXTN
C000 KBD	C010 KBDSTRB	FD21 KE1	FD1B KEYIN
? 39 KSWH	38 KSWL	? 2F LASTIN	2F LENGTH
FC66 LF	FE63 LI1	0400 LINE1	?FE60 LIST
2C LMNEM	FB65 LO1	00 LOC0	01 LOC1
FBA2 LOCA1	FBB2 LOCA2	FBA0 LOCAL	FB09 LOCCHR
FB99 LOCJMP	FB60 LOGO	C056 LORES	C054 LOWSCR
?FE21 LT	FE23 LT1	2E MASK	C052 MIXCLR
F9C0 MNEML	FA00 MNEMR	F8BE MNNDX1	F8C2 MNNDX2
F8C9 MNNDX3	FDAD MOD8CHK	31 MODE	FF65 MON
FF69 MONZ	FE2C MOVE	07F8 MSLOT	FDFA NEWLN
?FA6F NEWMON	03FB NMI	FBED NOCTRL	FAA3 NOFIX
FD5F NOTCR1	FD3D NOTCR	FCC8 NX1	FCBA NXTA1
FCB4 NXTA4	FF98 NXTBAS	FF90 NXTBIT	FFA2 NXTBS2
FD75 NXTCHAR	FFAD NXTCHR	?F85F NXTCOL	FF73 NXTITM
FF0F ODDCHR	FA59 OLDBRK	?FF59 OLDRST	?FE95 OUTPORT
?FE97 OUTPRT	C064 PADDL0	?F954 PCADJ2	F953 PCADJ
F956 PCADJ3	F95C PCADJ4	3B PCH	3A PCL
FBDF PIP	F80C PL1	F800 PLOT	F80E PLOT1
FCED PLOT80	F8F5 PR1	F8F9 PR2	FD92 PRA1
F910 PRADR1	F914 PRADR2	F926 PRADR3	F92A PRADR4
F930 PRADR5	F94A PRBL2	?F94C PRBL3	F948 PRBLNK
FDDA PRBYTE	FB25 PRE1	?FB1E PREAD	?FF2D PRERR
?FDE3 PRHEX	FDE5 PRHEXZ	?F941 PRNTAX	F8DB PRNTBL
F8D4 PRNTOP	?F944 PRNTX	F940 PRNTYX	33 PROMPT
FD96 PRYX2	C070 PTRIG	FAFD PWRCON	03F4 PWREDUP
FAA6 PWRUP	FD35 RDCHAR	FBB9 RDCHAR1	FD0C RDKEY
?FEFD READ	FAD7 REGDSP	?FEBF REGZ	?F938 RELADR

SYMBOL TABLE SORTED BY SYMBOL

FA62 RESET	FF3F RESTORE	?FF44 RESTR1	FAE4 RG1
FADA RGDSP1	2D RMNEM	4F RNDH	4E RNDL
FB19 RTBL	F831 RTS1	F961 RTS2	FC2A RTS4
FB2E RTSX	FF4C SAV1	?FF4A SAVE	FC76 SC1
FC95 SC3	F87F SCR1	FF09 SCRL80	FF28 SCRLEX
F879 SCRN2	FCFA SCRNX	?F871 SCRN	?FC70 SCROLL
FCD5 SE1	FCC9 SELBNK	?F864 SETCOL	?FB40 SETGR
FE86 SETIFLG	?FE80 SETINV	FE89 SETKBD	FE1E SETMDZ
?FE19 SETMODE	FE84 SETNORM	?FAA9 SETPG3	FAAB SETPLP
?FB6F SETPWRC	?FB39 SETTXT	FE93 SETVID	FB4B SETWND
FAC7 SL1	FABA SLOOP	03F2 SOFTEV	C030 SPKR
49 SPNT	FE18 ST1	48 STATUS	?FEC4 STEPZ
FECE STOR80	FBF0 STORADV	FBEE STORINV	?FE0C STOR
FEE1 STRTS	FFE3 SUBTBL	?FB5B TABV	FB90 TITLE
FFBE TOSUB	?FEC2 TRACE	?C058 TTLOUT0	?C05A TTLOUT1
?C05C TTLOUT2	?C05E TTLOUT3	C050 TXTCLR	FC35 UP1
?FC2F UPPER	?FC1A UP	03F8 USRADR	?FECA USR
2D V2	FE53 VE1	FE36 VERIFY	FB82 VI1
FB8E VI2	?C00A VID40	C00B VID80	C00C VIDBNK
FBFD VIDOUT	FEE6 VIDPLP	FEE3 VIDRTS	FB78 VIDWAIT
F828 VLINE	F826 VLINEZ	FC22 VTAB	FEEB VTAB80
FC24 VTABZ	FCA9 WA1	FCAA WA2	FCA8 WAIT
50 WIDTH	23 WNDBTM	20 WNDLFT	22 WNDTOP
21 WNDWDTH	?FECD WRITE	FDA3 XAM8	FDB3 XAM
FDC6 XAMPM	?FE5C XBASIC	46 XREG	47 YREG
34 YSAV	35 YSAV1	FFC7 ZMODE	

SYMBOL TABLE SORTED BY ADDRESS

```
   00 FORTYCOL        00 CFONT1          00 LOC0            00 CFONT3
   01 CFONT2          01 LOC1            20 WNDLFT          21 WNDWDTH
   22 WNDTOP          23 WNDBTM          24 CH              25 CV
   26 GBASL           27 GBASH           28 BASL            29 BASH
   2A BAS2L           2B BAS2H           2C H2              2C LMNEM
   2D V2              2D RMNEM           2E MASK            2E FORMAT
?  2F LASTIN          2F LENGTH          30 COLOR           31 MODE
   32 INVFLG          33 PROMPT          34 YSAV            35 YSAV1
   36 CSWL         ?  37 CSWH            38 KSWL         ?  39 KSWH
   3A PCL             3B PCH             3C A1L             3D A1H
   3E A2L             3F A2H             40 A3L             41 A3H
   42 A4L             43 A4H             45 ACC             46 XREG
   47 YREG            48 STATUS          49 SPNT            4E RNDL
   4F RNDH            50 WIDTH           80 BIT7            C0 IOPAGE
 0200 IN           03F0 BRKV          03F2 SOFTEV        03F4 PWREDUP
?03F5 AMPERV        03F8 USRADR        03FB NMI           03FE IRQLOC
 0400 LINE1         0479 CHY           07F8 MSLOT         C000 CHRINV
?C000 IOARD         C000 KBD           C002 CHRGEN0       C004 CHRGEN1
?C006 CHRBAS        C008 KBDEXTN       ?C00A VID40        C00B VID80
 C00C VIDBNK        C010 KBDSTRB       C030 SPKR          C050 TXTCLR
 C052 MIXCLR        C054 LOWSCR        C056 LORES         ?C058 TTLOUT0
?C05A TTLOUT1       ?C05C TTLOUT2      ?C05E TTLOUT3       C064 PADDL0
 C070 PTRIG         CFFF CLRROM        E000 BASIC         ?E003 BASIC2
 F800 PLOT          F80C PL1           F80E PLOT1         ?F819 HLINE
 F81C HL1           F826 VLINEZ        F828 VLINE          F831 RTS1
?F832 CLRSCR        F836 CLRTOP        F838 CLRSC2         F83C CL1
 F847 GBASCALC      F856 GB1           ?F85F NXTCOL       ?F864 SETCOL
?F871 SCRN          F879 SCRN2         F87F SCR1           F882 INSDS1
?F88C INSDS2        F89B IEVEN         F8A5 ERR            F8A9 GETFMT
 F8BE MNNDX1        F8C2 MNNDX2        F8C9 MNNDX3         F8D0 INSTDSP
 F8D4 PRNTOP        F8DB PRNTBL        F8F5 PR1            F8F9 PR2
 F910 PRADR1        F914 PRADR2        F926 PRADR3         F92A PRADR4
 F930 PRADR5       ?F938 RELADR        F940 PRNTYX        ?F941 PRNTAX
?F944 PRNTX         F948 PRBLNK        F94A PRBL2         ?F94C PRBL3
 F953 PCADJ        ?F954 PCADJ2        F956 PCADJ3         F95C PCADJ4
 F961 RTS2          F962 FMT1          F9A6 FMT2           F9B4 CHAR1
 F9BA CHAR2         F9C0 MNEML         FA00 MNEMR          FA40 IRQ
 FA4C BREAK         FA59 OLDBRK        FA62 RESET         ?FA6F NEWMON
 FA9B FIXSEV        FAA3 NOFIX         FAA6 PWRUP         ?FAA9 SETPG3
 FAAB SETPLP        FABA SLOOP         FAC7 SL1            FAD7 REGDSP
 FADA RGDSP1        FAE4 RG1           FAFD PWRCON         FB01 DISKID
 FB09 LOCCHR        FB19 RTBL         ?FB1E PREAD          FB25 PRE1
 FB2E RTSX          FB2F INIT         ?FB39 SETTXT        ?FB40 SETGR
 FB4B SETWND       ?FB5B TABV          FB60 LOGO           FB65 LO1
?FB6F SETPWRC       FB78 VIDWAIT       FB82 VI1            FB8E VI2
 FB90 TITLE         FB99 LOCJMP        FBA0 LOCAL          FBA2 LOCA1
 FBB2 LOCA2         FBB6 JLOCAL        FBB9 RDCHAR1        FBC1 BASCALC
 FBD0 BA1           FBD9 BELL1         FBDF PIP            FBED NOCTRL
 FBEE STORINV       FBF0 STORADV       FBF4 ADVANCE        FBFD VIDOUT
?FC10 BS           ?FC1A UP            FC22 VTAB          FC24 VTABZ
 FC2A RTS4          FC2B GETUPCS      ?FC2F UPPER          FC35 UP1
?FC42 CLREOP        FC46 CLEOP1        FC58 HOME           FC62 CR
 FC66 LF           ?FC70 SCROLL        FC76 SC1            FC95 SC3
 FC9C CLREOL        FC9E CLEOLZ        FCA8 WAIT           FCA9 WA1
 FCAA WA2           FCB4 NXTA4         FCBA NXTA1          FCC8 NX1
```

SYMBOL TABLE SORTED BY ADDRESS

```
 FCC9 SELBNK        FCD5 SE1          FCDB CLEOL80      FCED PLOT80
 FCFA SCRNX         FD0C RDKEY        FD1B KEYIN        FD21 KE1
 FD35 RDCHAR        FD3D NOTCR        FD5F NOTCR1       FD62 CANCEL
 FD67 GETLNZ       ?FD6A GETLN        FD71 BCKSPC       FD75 NXTCHAR
 FD84 ADDINP        FD8E CROUT        FD92 PRA1         FD96 PRYX2
 FDA3 XAM8          FDAD MOD8CHK      FDB3 XAM          FDB6 DATAOUT
 FDC6 XAMPM         FDD1 ADD          FDDA PRBYTE      ?FDE3 PRHEX
 FDE5 PRHEXZ        FDED COUT        ?FDF0 COUT1        FDFA NEWLN
 FE01 BL1          ?FE05 BLANK       ?FE0C STOR         FE18 ST1
?FE19 SETMODE       FE1E SETMDZ      ?FE21 LT           FE23 LT1
 FE2C MOVE          FE36 VERIFY       FE53 VE1         ?FE59 BASCONT
?FE5C XBASIC       ?FE60 LIST         FE63 LI1          FE75 A1PC
 FE78 A1PC1         FE7F A1PC2       ?FE80 SETINV       FE84 SETNORM
 FE86 SETIFLG       FE89 SETKBD      ?FE8B INPORT      ?FE8D INPRT
 FE93 SETVID       ?FE95 OUTPORT     ?FE97 OUTPRT       FE9B IOPRT
 FEA7 IOPRT1        FEA9 IOPRT2      ?FEB6 GO          ?FEBF REGZ
?FEC2 TRACE        ?FEC4 STEPZ       ?FECA USR         ?FECD WRITE
 FECE STOR80        FED7 CURS80       FEE1 STRTS        FEE3 VIDRTS
 FEE6 VIDPLP        FEEB VTAB80      ?FEF6 CRMON       ?FEFD READ
 FEFE GET80         FF09 SCRL80       FF0F ODDCHR       FF1E EVENCHR
 FF28 SCRLEX       ?FF2D PRERR        FF3A BELL         FF3F RESTORE
?FF44 RESTR1       ?FF4A SAVE         FF4C SAV1        ?FF58 IORTS
?FF59 OLDRST        FF65 MON          FF69 MONZ         FF73 NXTITM
 FF7A CHRSRCH       FF8A DIG          FF90 NXTBIT       FF98 NXTBAS
 FFA2 NXTBS2        FFA7 GETNUM       FFAD NXTCHR       FFBE TOSUB
 FFC7 ZMODE         FFCC CHRTBL       FFE3 SUBTBL
```

APPENDIX D

PRINTER ROM LISTING

This Appendix contains an assembly listing of the version 2.1 ROM used for parallel and serial communications control. The ROM is located in the peripheral I/O address space for Slot 1 and begins at $C100.

```
PAGE -   1  PRINTER   FILE:


0000|                           .proc   printer
Current memory available:    8644
0000|
0000| 0021                    version .equ    21      ; version 2.1
0000|
0000| C100                    rom     .equ    0C100
0000| 00C1                    rompage .equ    0C1
0000|                           .org    rom
C100|
C100| C090                    devsel  .equ    0C090
C100| C1C1                    pready  .equ    0C1C1
C100|
C100| C090                    preg    .equ    devsel
C100| C098                    acia    .equ    devsel+8
C100|
C100| C098                    inreg   .equ    acia+0  ;   7    6    5    4    3    2    1    0
C100| C098                    outreg  .equ    acia+0  ;
C100|
C100| C099                    stsreg  .equ    acia+1  ; IRQ   DSR   DCD   tran  rec   ovr-  frm-  par-
C100|                                           ; occur inact inact empty full    e r r o r
C100|
C100| C09A                    cmdreg  .equ    acia+2  ;    parity   par   rec   transmit-   rec-  DTR
C100|                                           ;  mode-ctrl  enabl echo  IRQ,RTS,brk IRQ   activ
C100|
C100| C09B                    ctrlreg .equ    acia+3  ; 2 stop word-      clock  b a u d    r a t e
C100|                                           ; bits       length intrn
C100|
C100| 0478                    Accu    .equ    478     ; save char
C100| 04F8                    chanel  .equ    4F8     ; par/ser out switch : if chanel<80 then par else ser
C100|
C100| 0479                    vid0    .equ    479     ; used in the 80-col screen driver
C100| 04F9                    vid1    .equ    4F9     ; reserved
C100| 0579                    vid2    .equ    579     ; reserved
C100| 05F9                    modechk .equ    5F9     ; warmstart byte
C100| 0679                    mode    .equ    679     ; CR->CR/LF    video echo
C100|                                           ; par    ser   par  ser
C100| 06F9                    ctrl    .equ    6F9     ; value for ACIA ctrl-reg
C100| 0779                    cmd     .equ    779     ; value for ACIA cmd-reg
C100|
C100| 07F9                    hCount  .equ    7F9
C100|
C100| 0024                    ch      .equ    24
C100| 0036                    csw     .equ    36
C100| 0038                    ksw     .equ    38
C100| FDF0                    cout1   .equ    0FDF0
C100|
C100| 002C                    bit_a   .equ    2C
C100| 20 ****                         jsr     init
C103| 90**                            bcc     pwrite2
C105|
C105|                           .org    rom+5
C105| 48                      byte5   pha             ; tested by Pascal
C106| 21                              .byte   version
C107| 48                      byte7   pha             ; tested by Pascal
C108|
```

```
C108|                           .org    rom+8
C108| 48                  v24   pha             ; first entry for IN#9 or PR#9
C109| A5 39                      lda     ksw+1
C10B| C9 C1                      cmp     #rompage
C10D| D0**                      bne     swrite  ; no
C10F| 68                        pla
C110| A9 14              sread   lda     #14     ; yes, first entry
C112| 85 38                      sta     ksw     ; zap entry to sread2
C114|
C114|                           .org    rom+14
C114| 20 ****            sread2  jsr     init
C117| A9 08                      lda     #8
C119| 2C 99C0            $0      bit     stsreg
C11C| F0FB                      beq     $0
C11E| AD 98C0                   lda     inreg
C121| 49 80                      eor     #80
C123| 60                        rts
C124|
C124|
C10D* 15
C124| A9 29              swrite  lda     #29     ; first PR#9 entry
C126| 85 36                      sta     csw     ; zap entry vector
C128| 68                        pla
C129|
C129|                           .org    rom+29
C129| 20 ****            swrite2 jsr     init    ; setup the 6551
C12C| 38                        sec
C12D|
C103* 28
C12D| 6E F804            pwrite2 ror     chanel
C130|                     output
C130| EE F907            $1      inc     hCount
C133| A5 24                      lda     ch
C135| CD F907                    cmp     hCount
C138| 90**                       bcc     notab
C13A| A9 A0                      lda     #0A0
C13C| 20 ****                    jsr     out1
C13F| 4C 30C1                    jmp     $1
C138* 08
C142| 20 ****            notab   jsr     out
C145| C9 0D                      cmp     #0D
C147| D0**                       bne     nocr
C149| 20 ****                    jsr     cCount
C14C| 2C 7906                    bit     mode
C14F| 10**                       bpl     nocr
C151| A9 8A                      lda     #8A
C153| 20 ****                    jsr     out1
C14F* 05
C147* 0D
C156| 2C 7906            nocr    bit     mode
C159| AD 7804                    lda     accu
C15C| 50**                       bvc     ret
C15E| 4C F0FD                    jmp     cout1
C161|
C161| 00                        brk
```

```
C162|
C12A* 62C1
C115* 62C1
C101* 62C1
C162| 8D 7804              init    sta     Accu    ; low(addr)=Fx
C165| AD F905                      lda     modechk
C168| 49 A5                        eor     #0A5    ; printer/v24 warmstart?
C16A| CD 7906                      cmp     mode
C16D| F0**                        beq     warm    ; yes
C16F|
C16F| A9 9E                        lda     #9E     ; no, set default values: 8 data+2 stop bits,
C171| 8D F906                      sta     ctrl    ;                          9600 baud
C174|
C174| A9 0B                        lda     #0B     ; no parity, DTR=low,   RTS=low
C176| 8D 7907                      sta     cmd
C179|
C179| A9 C0                        lda     #0C0     ; mode bit 7:   CR->CR/LF translation on
C17B| 8D 7906                      sta     mode     ;      bit 6:   output echo to video
C17E|
C17E| 49 A5                        eor     #0A5
C180| 8D F905                      sta     modechk  ; set warmstart flag
C183|
C14A* 83C1
C183| A9 00              cCount   lda     #0
C185| 8D F907                      sta     hCount  ; init Tabulator count
C188|
C16D* 19
C188| AD 7907            warm     lda     cmd
C18B| CD 9AC0                      cmp     cmdreg   ; is the 6551 cmd  register ok ?
C18E| F0**                        beq     $1
C190| 8D 9AC0                      sta     cmdreg   ; no
C18E* 03
C193| AD F906            $1       lda     ctrl
C196| CD 9BC0                      cmp     ctrlreg  ; is the 6551 ctrl register ok ?
C199| F0**                        beq     $2
C19B| 8D 9BC0                      sta     ctrlreg  ; no
C199* 03
C19E| 18                $2       clc
C15C* 41
C19F| 60                ret      rts
C1A0|
C143* A0C1
C1A0| AD 7804            out      lda     Accu
C154* A3C1
C13D* A3C1
C1A3| 49 80              out1     eor     #80
C1A5| 2C F804                     bit     chanel
C1A8| 10**                       bpl     pout
C1AA|
C1AA| 48                sout     pha             ; save char
C1AB| A9 10                       lda     #10
C1AD| 2C 99C0            $0       bit     stsreg  ; ready for next char ?
C1B0| F0FB                       beq     $0      ; no, wait
C1B2| 68                        pla             ; yes
C1B3| 8D 98C0                     sta     outreg  ; send it
```

```
PAGE -   4  PRINTER   FILE:


C1B6| 60                              rts
C1B7|
C1A8* 0D
C1B7| 2C C1C1                 pout    bit     pready
C1BA| 30FB                            bmi     pout
C1BC| 8D 90C0                         sta     devsel
C1BF| 60                              rts
C1C0|
C1C0|                                 .org    rom+0C0
C1C0|                                 .end
```

```
PAGE -   5  PRINTER   FILE:  SYMBOLTABLE DUMP


AB - Absolute     LB - Label     UD - Undefined     MC - Macro
RF - Ref          DF - Def       PR - Proc          FC - Func
PB - Public       PV - Private   CS - Consts

ACCU     AB 0478|  ACIA     AB C098|  BITA     AB 002C|  BYTE5    LB C105|  BYTE7    LB C107|
CHANEL   AB 04F8|  CMD      AB 0779|  CMDREG   AB C09A|  COUT1    AB FDF0|  CSW      AB 0036|
DEVSEL   AB C090|  HCOUNT   AB 07F9|  INIT     LB C162|  INREG    AB C098|  KSW      AB 0038|
NOCR     LB C156|  NOTAB    LB C142|  OUT      LB C1A0|  OUT1     LB C1A3|  OUTPUT   LB C130|
PREADY   AB C1C1|  PREG     AB C090|  PRINTER  PR ----|  PWRITE2  LB C12D|  RET      LB C19F|
SOUT     LB C1AA|  SREAD    LB C110|  SREAD2   LB C114|  STSREG   AB C099|  SWRITE   LB C124|
VERSION  AB 0021|  VID0     AB 0479|  VID1     AB 04F9|  VID2     AB 0579|  WARM     LB C188|

CCOUNT   LB C183|  CH       AB 0024
CTRL     AB 06F9|  CTRLREG  AB C09B
MODE     AB 0679|  MODECHK  AB 05F9
OUTREG   AB C098|  POUT     LB C1B7
ROM      AB C100|  ROMPAGE  AB 00C1
SWRITE2  LB C129|  V24      LB C108
```

APPENDIX E

6502 MICROPROCESSOR

This Appendix contains information about the 6502 microprocessor's architecture, operation and instruction set.

Thank you to Synertek for permission to reprint their material.

8-Bit Microprocessor Family

SYNERTEK

SY6500

MICROPROCESSOR PRODUCTS

- Single 5 V ±5% power supply
- N channel, silicon gate, depletion load technology
- Eight bit parallel processing
- 56 Instructions
- Decimal and binary arithmetic
- Thirteen addressing modes
- True indexing capability
- Programmable stack pointer
- Variable length stack
- Interrupt capability
- Non-maskable interrupt
- Use with any type or speed memory
- Bi-directional Data Bus
- Instruction decoding and control
- Addressable memory range of up to 65 K bytes
- "Ready" input
- Direct memory access capability
- Bus compatible with MC6800
- Choice of external or on-board clocks
- 1 MHz, 2 MHz, 3 MHz and 4 MHz operation
- On-chip clock options
  - External single clock input
  - Crystal time base input
- 40 and 28 pin package versions
- Pipeline architecture

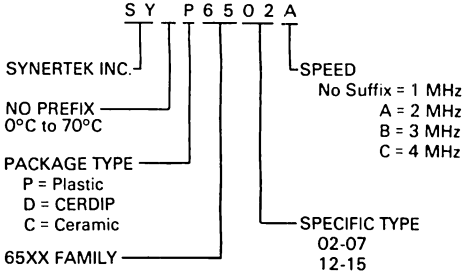
The SY6500 Series Microprocessors represent the first totally software compatible microprocessor family. This family of products includes a range of software compatible microprocessors which provide a selection of addressable memory range, interrupt input options and on-chip clock oscillators and drivers. All of the microprocessors in the SY6500 family are software compatible within the group and are bus compatible with the MC6800 product offering.

The family includes six microprocessors with on-board clock oscillators and drivers and four microprocessors driven by external clocks. The on-chip clock versions are aimed at high performance, low cost applications where single phase inputs or crystals provide the time base. The external clock versions are geared for the multi-processor system applications where maximum timing control is mandatory. All versions of the microprocessors are available in 1 MHz, 2 MHz, 3 MHz and 4 MHz maximum operating frequencies.

MEMBERS OF THE FAMILY

PART NUMBERS	CLOCKS	PINS	$\overline{IRQ}$	$\overline{NMI}$	RYD	ADDRESSING
SY6502	On-Chip	40	√	√	√	64 K
SY6503	"	28	√	√		4 K
SY6504	"	28	√			8 K
SY6505	"	28	√		√	4 K
SY6506	"	28	√			4 K
SY6507	"	28			√	8 K
SY6512	External	40	√	√	√	64 K
SY6513	"	28	√	√		4 K
SY6514	"	28	√			8 K
SY6515	"	28	√		√	4 K

ORDERING INFORMATION

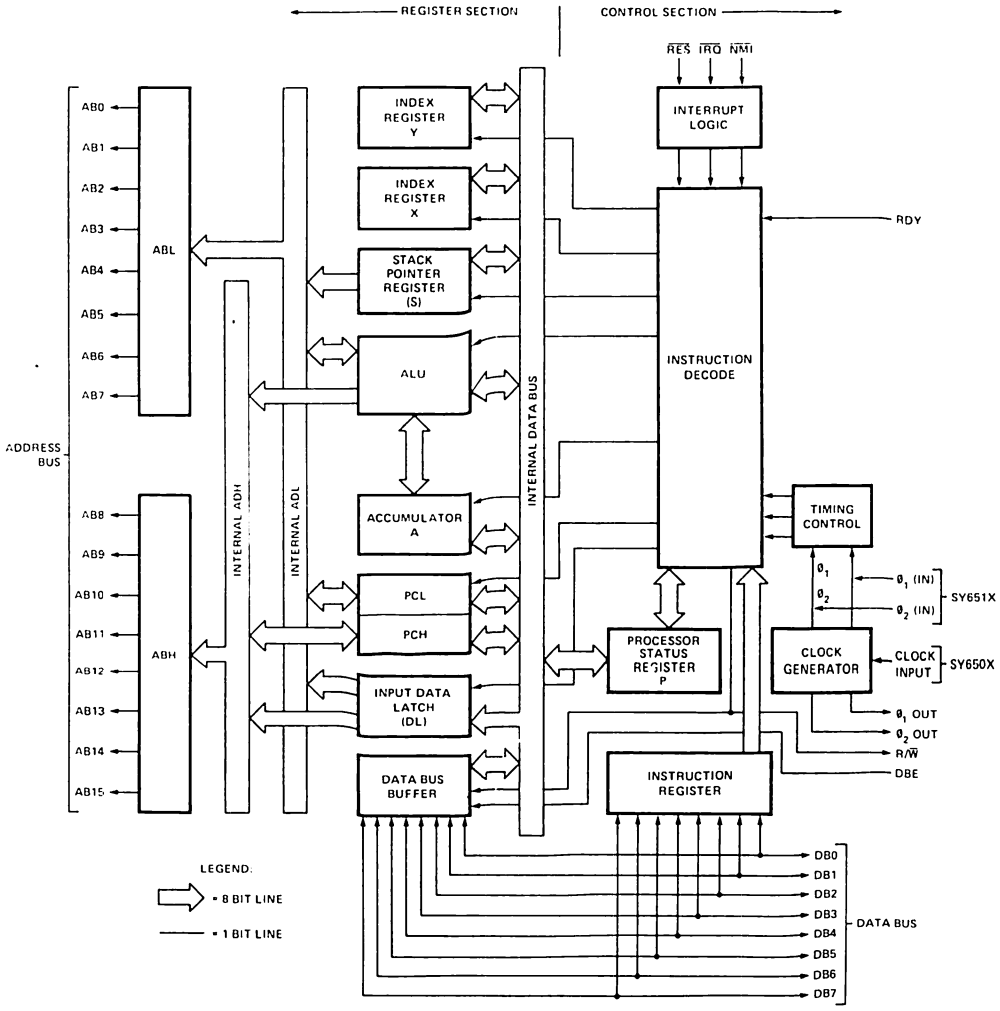


Only 6502 and 6512 are available in 3 and 4 MHz

COMMENTS ON THE DATA SHEET

The data sheet is constructed to review first the basic "Common Characteristics" – those features which are common to the general family of microprocessors. Subsequent to a review of the family characteristics will be sections devoted to each member of the group with specific features of each.

SY6500 INTERNAL ARCHITECTURE



NOTE:
1. CLOCK GENERATOR IS NOT INCLUDED ON SY651X.
2. ADDRESSING CAPABILITY AND CONTROL OPTIONS VARY WITH EACH OF THE SY6500 PRODUCTS.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 to +7.0	V
Input Voltage	V_{in}	-0.3 to +7.0	V
Operating Temperature	T_A	0 to +70	°C
Storage Temperature	T_{STG}	-55 to +150	°C

COMMENT

This device contains input protection against damage due to high static voltages or electric fields; however, precautions should be taken to avoid application of voltages higher than the maximum rating.

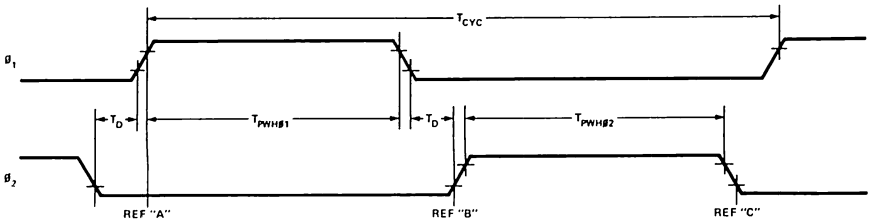
D.C. CHARACTERISTICS (V_{CC} = 5.0V ±5%, T_A = 0-70°C)

(ϕ_1, ϕ_2 applies to SY651X, $\phi_{o(in)}$ applies to SY650X)

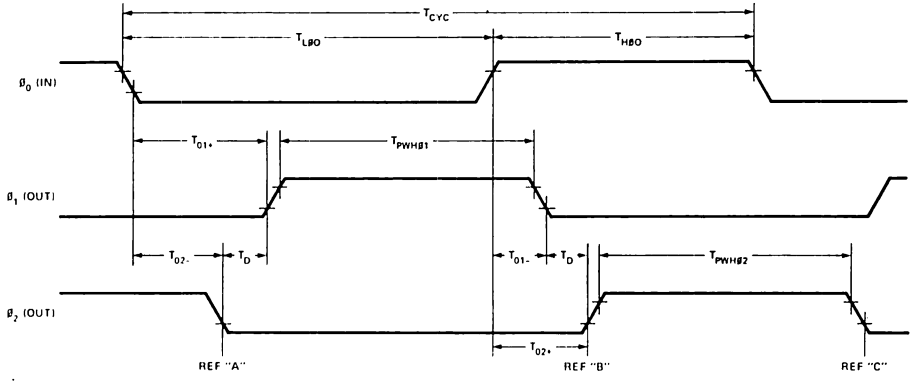
Symbol	Characteristic		Min.	Max.	Unit
V_{IH}	Input High Voltage				
	Logic and ϕ_o (in) for all 650X devices	1,2,3 MHz	+2.4	V_{CC}	V
		4 MHz	+3.3	V_{CC}	V
	ϕ_1 and ϕ_2 only for all 651X devices. Logic as 650X	All Speeds	V_{CC} -0.5	V_{CC} + 0.25	V
V_{IL}	Input Low Voltage				
	Logic, $\phi_{o(in)}$	(650X)	-0.3	+0.4	
	ϕ_1, ϕ_2	(651X)	-0.3	+0.2	V
I_{IL}	Input Loading (V_{in} = 0 V, V_{CC} = 5.25 V) RDY, S.O.		-10	-300	μA
I_{in}	Input Leakage Current (V_{in} = 0 to 5.25 V, V_{CC} = 0)				
	Logic (Excl. RDY, S.O.)		–	2.5	μA
	ϕ_1, ϕ_2	(651X)	–	100	μA
	$\phi_{o(in)}$	(650X)	–	10.0	μA
I_{TSI}	Three-State (Off State) Input Current (V_{in} = 0.4 to 2.4 V, V_{CC} = 5.25 V) DB0-DB7		–	±10	μA
V_{OH}	Output High Voltage				
	(I_{LOAD} = -100μAdc, V_{CC} = 4.75 V)	1,2,3 MHz	2.4	–	V
	SYNC, DB0-DB7, A0-A15, R/$\overline{W}$	4 MHz	2.0	–	V
V_{OL}	Output Low Voltage				
	(I_{LOAD} = 1.6mAdc, V_{CC} = 4.75 V)	1,2,3 MHz	–	0.4	V
	SYNC, DB0-DB7, A0-A15, R/$\overline{W}$	4 MHz	–	0.8	V
P_D	Power Dissipation	1 MHz and 2 MHz	–	700	mW
	(V_{CC} = 5.25V)	3 MHz	–	800	mW
		4 MHz	–	900	mW
C	Capacitance (V_{in} = 0, T_A = 25°C, f = 1 MHz)				
C_{in}	$\overline{RES}$, $\overline{NMI}$, RDY, $\overline{IRQ}$, S.O., DBE		–	10	
	DB0-DB7		–	15	
C_{out}	A0-A15, R/$\overline{W}$, SYNC		–	12	pF
$C_{\phi_{o(in)}}$	$\phi_{o(in)}$	(650X)	–	15	
C_{ϕ_1}	ϕ_1	(651X)	–	50	
C_{ϕ_2}	ϕ_2	(651X)	–	80	

TIMING DEFINITIONS, COMPOSITE TIMING DIAGRAM (See note at bottom of page 5)

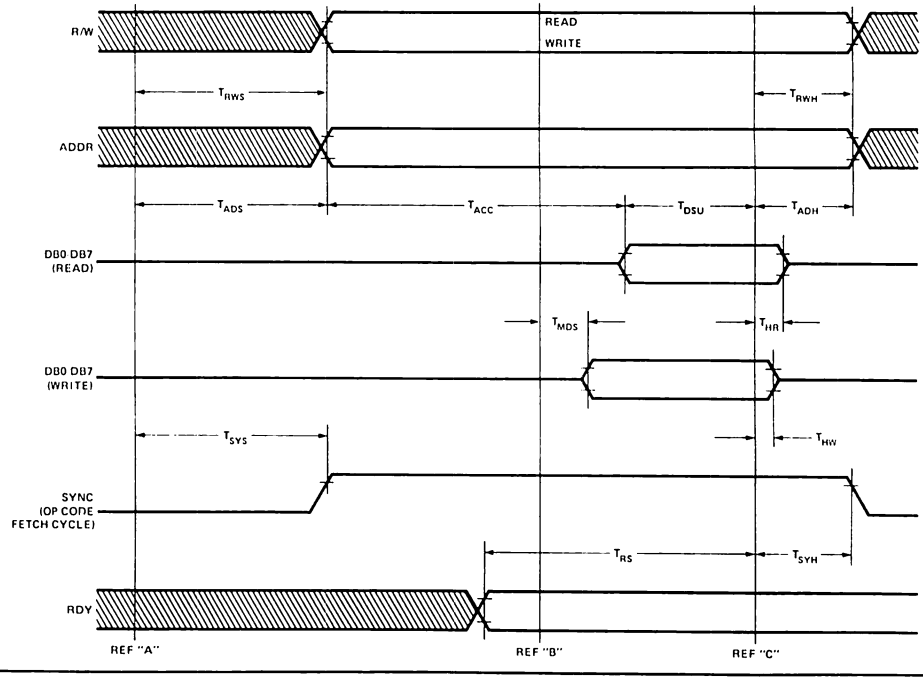
SY651X INPUT CLOCK TIMING



SY650X INPUT CLOCK TIMING



SY65XX TIMING (See note at bottom of page 5)



DYNAMIC OPERATING CHARACTERISTICS

(V_{CC} = 5.0 ± 5%, T_A = 0° to 70°C)

Parameter	Symbol	1 MHz		2 MHz		3 MHz		4 MHz		Units
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
651X										
Cycle Time	T_{CYC}	1.00	40	0.50	40	0.33	40	0.25	40	µs
$Ø_1$ Pulse Width	$T_{PWHØ_1}$	430	—	215	—	150	—			ns
$Ø_2$ Pulse Width	$T_{PWHØ_2}$	470	—	235	—	160	—			ns
Delay Between $Ø_1$ and $Ø_2$	T_D	0	—	0	—	0	—			ns
$Ø_1$ and $Ø_2$ Rise and Fall Times[1]	T_R, T_F	0	25	0	20	0	15			ns
650X										
Cycle Time	T_{CYC}	1.00	40	0.50	40	0.33	40	0.25	40	µs
$Ø_{o(IN)}$ Low Time[2]	$T_{LØ_o}$	480	—	240	—	160	—	110	—	ns
$Ø_{o(IN)}$ High Time[2]	$T_{HØ_o}$	460	—	240	—	160	—	115	—	ns
$Ø_o$ Neg to $Ø_1$ Pos Delay[5]	T_{O1}	10	70	10	70	10	70	10	70	ns
$Ø_o$ Neg to $Ø_2$ Neg Delay[5]	T_{O2}	5	65	5	65	5	65	5	65	ns
$Ø_o$ Pos to $Ø_1$ Neg Delay[5]	T_{O1}	5	65	5	65	5	65	5	65	ns
$Ø_o$ Pos to $Ø_2$ Pos Delay[5]	T_{O2}	15	75	15	75	15	75	15	75	ns
$Ø_{o(IN)}$ Rise and Fall Time[1]	T_{RO}, T_{FO}	0	30	0	20	0	15	0	10	ns
$Ø_{1(OUT)}$ Pulse Width	$T_{PWHØ_1}$	$T_{LØ_o}$-20	$T_{LØ_o}$	$T_{LØ_o}$-20	$T_{LØ_o}$	$T_{LØ_o}$-20	$T_{LØ_o}$	$T_{LØ_o}$-20	$T_{LØ_o}$	ns
$Ø_{2(OUT)}$ Pulse Width	$T_{PWHØ_2}$	$T_{LØ_o}$-40	$T_{LØ_o}$-10	$T_{LØ_o}$-40	$T_{LØ_o}$-10	$T_{LØ_o}$-40	$T_{LØ_o}$-10	$T_{LØ_o}$-40	$T_{LØ_o}$-10	ns
Delay Between $Ø_1$ and $Ø_2$	T_D	5	—	5	—	5	—	5	—	ns
$Ø_1$ and $Ø_2$ Rise and Fall Times[1,3]	T_R, T_F	—	25	—	25	—	15	—	15	ns
650X, 651X										
R/$\overline{W}$ Setup Time	T_{RWS}	—	225	—	140	—	110	—	90	ns
R/$\overline{W}$ Hold Time	T_{RWH}	30	—	30	—	15	—	10	—	ns
Address Setup Time	T_{ADS}	—	225	—	140	—	110	—	90	ns
Address Hold Time	T_{ADH}	30	—	30	—	15	—	10	—	ns
Read Access Time	T_{ACC}	—	650	—	310	—	170	—	110	ns
Read Data Setup Time	T_{DSU}	100	—	50	—	50	—	50	—	ns
Read Data Hold Time	T_{HR}	10	—	10	—	10	—	10	—	ns
Write Data Setup Time	T_{MDS}	20	175	20	100	20	75	—	70	ns
Write Data Hold Time	T_{HW}	60	150	60	150	30	130	20	—	ns
Sync Setup Time	T_{SYS}	—	350	—	175	—	100	—	90	ns
Sync Hold Time	T_{SYH}	30	—	30	—	15	—	15	—	ns
RDY Setup Time[4]	T_{RS}	200	—	200	—	150	—	120	—	ns

NOTES:

1. Measured between 10% and 90% points on waveform.
2. Measured at 50% points.
3. Load = 1 TTL load +30 pF.
4. RDY must never switch states within T_{RS} to end of $Ø_2$.
5. Load = 100 pF.
6. The 2 MHz devices are identified by an "A" suffix.
7. The 3 MHz devices are identified by a "B" suffix.
8. The 4 MHz devices are identified by a "C" suffix.

TIMING DIAGRAM NOTE:

Because the clock generation for the SY650X and SY651X is different, the two clock timing sections are referenced to the main timing diagram by three reference lines marked REF 'A', REF 'B' and REF 'C'. Reference between the two sets of clock timings is without meaning. Timing parameters are referred to these lines and scale variations in the diagrams are of no consequence.

PIN FUNCTIONS

Clocks (ϕ_1, ϕ_2)

The SY651X requires a two phase non-overlapping clock that runs at the V_{CC} voltage level.

The SY650X clocks are supplied with an internal clock generator. The frequency of these clocks is externally controlled. Clock generator circuits are shown elsewhere in this data sheet.

Address Bus (A_0-A_{15}) (See sections on each micro for respective address lines on those devices.)

These outputs are TTL compatible, capable of driving one standard TTL load and 130 pF.

Data Bus (DB_0-DB_7)

Eight pins are used for the data bus. This is a bi-directional bus, transferring data to and from the device and peripherals. The outputs are three-state buffers, capable of driving one standard TTL load and 130 pF.

Data Bus Enable (DBE)

This TTL compatible input allows external control of the three-state data output buffers and will enable the microprocessor bus driver when in the high state. In normal operation DBE would be driven by the phase two (ϕ_2) clock, thus allowing data output from microprocessor only during ϕ_2. During the read cycle, the data bus drivers are internally disabled, becoming essentially an open circuit. To disable data bus drivers externally, DBE should be held low. This signal is available on the SY6512, only.

Ready (RDY)

This input signal allows the user to halt the microprocessor on all cycles except write cycles. A negative transition to the low state during or coincident with phase one (ϕ_1) will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two (ϕ_2) in which the Ready signal is low. This feature allows microprocessor interfacing with low speed PROMS as well as fast (max. 2 cycle) Direct Memory Access (DMA). If ready is low during a write cycle, it is ignored until the following read operation. Ready transitions must not be permitted during ϕ_2 time.

Interrupt Request ($\overline{IRQ}$)

This TTL level input requests that an interrupt sequence begin within the microprocessor. The microprocessor will complete the current instruction being executed before recognizing the request. At that time, the interrupt mask bit in the Status Code Register will be examined. If the interrupt mask flag is not set, the microprocessor will begin an interrupt sequence. The Program Counter and Processor Status Register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further interrupts may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, therefore transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3KΩ external resistor should be used for proper wire-OR operation.

Non-Maskable Interrupt ($\overline{NMI}$)

A negative going transition on this input requests that a non-maskable interrupt sequence be generated within the microprocessor.

$\overline{NMI}$ is an unconditional interrupt. Following completion of the current instruction, the sequence of operations defined for $\overline{IRQ}$ will be performed, regardless of the state interrupt mask flag. The vestor address loaded into the program counter, low and high, are locations FFFA and FFFB respectively, thereby transferring program control to the memory vector located at these addresses. The instructions loaded at these locations cause the microprocessor to branch to a non-maskable interrupt routine in memory.

$\overline{NMI}$ also requires an external 3KΩ resistor to V_{CC} for proper wire-OR operations.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupts lines that are sampled during ϕ_2 (phase 2) and will begin the appropriate interrupt routine on the ϕ_1 (phase 1) following the completion of the current instruction.

Set Overflow Flag (S.O.)

A NEGATIVE going edge on this input sets the overflow bit in the Status Code Register. This signal is sampled on the trailing edge of ϕ_1.

SYNC

This output line is provided to identify those cycles in which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during ϕ_1 of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the ϕ_1 clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

Reset ($\overline{RES}$)

This input is used to reset or start the microprocessor from a power down condition. During the time that this line is held low, writing to or from the microprocessor is inhibited. When a positive edge is detected on the input, the microprocessor will immediately begin the reset sequence.

After a system initialization time of six clock cycles, the mask interrupt flag will be set and the microprocessor will load the program counter from the memory vector locations FFFC and FFFD. This is the start location for program control.

After V_{CC} reaches 4.75 volts in a power up routine, reset must be held low for at least two clock cycles. At this time the R/$\overline{W}$ and SYNC signal will become valid.

When the reset signal goes high following these two clock cycles, the microprocessor will proceed with the normal reset procedure detailed above.

Read/Write (R/$\overline{W}$)

This output signal is used to control the direction of data transfers between the processor and other circuits on the data bus. A high level on R/$\overline{W}$ signifies data into the processor; a low is for data transfer out of the processor.

PROGRAMMING CHARACTERISTICS

INSTRUCTION SET – ALPHABETIC SEQUENCE

ADC	Add Memory to Accumulator with Carry	DEC	Decrement Memory by One	PHA	Push Accumulator on Stack
AND	"AND" Memory with Accumulator	DEX	Decrement Index X by One	PHP	Push Processor Status on Stack
ASL	Shift left One Bit (Memory or Accumulator)	DEY	Decrement Index Y by One	PLA	Pull Accumulator from Stack
				PLP	Pull Processor Status from Stack
BCC	Branch on Carry Clear	EOR	"Exclusive or" Memory with Accumulator		
BCS	Branch on Carry Set			ROL	Rotate One Bit Left (Memory or Accumulator)
BEQ	Branch on Result Zero	INC	Increment Memory by One	ROR	Rotate One Bit Right (Memory or Accumulator)
BIT	Test Bits in Memory with Accumulator	INX	Increment Index X by One	RTI	Return from Interrupt
BMI	Branch on Result Minus	INY	Increment Index Y by One	RTS	Return from Subroutine
BNE	Branch on Result not Zero				
BPL	Branch on Result Plus	JMP	Jump to New Location	SBC	Subtract Memory from Accumulator with Borrow
BRK	Force Break	JSR	Jump to New Location Saving Return Address	SEC	Set Carry Flag
BVC	Branch on Overflow Clear			SED	Set Decimal Mode
BVS	Branch on Overflow Set	LDA	Load Accumulator with Memory	SEI	Set Interrupt Disable Status
		LDX	Load Index X with Memory	STA	Store Accumulator in Memory
CLC	Clear Carry Flag	LDY	Load Index Y with Memory	STX	Store Index X in Memory
CLD	Clear Decimal Mode	LSR	Shift One Bit Right (Memory or Accumulator)	STY	Store Index Y in Memory
CLI	Clear Interrupt Disable Bit				
CLV	Clear Overflow Flag	NOP	No Operation	TAX	Transfer Accumulator to Index X
CMP	Compare Memory and Accumulator			TAY	Transfer Accumulator to Index Y
CPX	Compare Memory and Index X	ORA	"OR" Memory with Accumulator	TSX	Transfer Stack Pointer to Index X
CPY	Compare Memory and Index Y			TXA	Transfer Index X to Accumulator
				TXS	Transfer Index X to Stack Pointer
				TYA	Transfer Index Y to Accumulator

ADDRESSING MODES

Accumulator Addressing

This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

Immediate Addressing

In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing

In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65K bytes of addressable memory.

Zero Page Addressing

The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Zero Page Addressing – (X, Y indexing)

This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y." The effective address is calcuated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing – (X, Y indexing)

This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X," and "Absolute, Y." The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing

In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing

Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.

The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is −128 to +127 bytes from the next instruction.

Indexed Indirect Addressing

In indexed indirect addressing (referred to as (Indirect,X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of this addition points to a memory location on page zero whose contents is the low order eight bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

Indirect Indexed Addressing

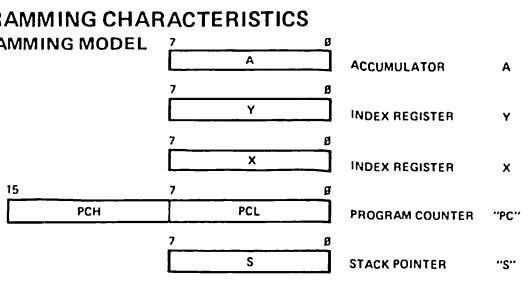
In indirect indexed addressing (referred to as (Indirect),Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y index register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

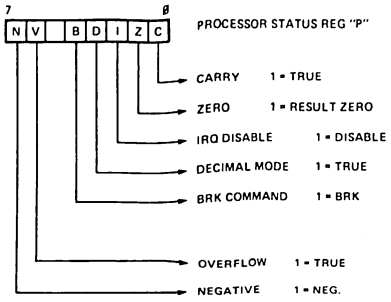
Absolute Indirect

The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address. The next memory location contains the high order byte of the effective address which is loaded into the sixteen bits of the program counter

PROGRAMMING CHARACTERISTICS

PROGRAMMING MODEL





INSTRUCTION SET – OP CODES, EXECUTION TIME, MEMORY REQUIREMENTS

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			INDX X			INDX Y			Z PAGE X			ABS X			ABS Y			RELATIVE			INDIRECT			Z PAGE Y			CONDITION CODES					
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
ADC	A + M + C → A (4) (11)	69	2	2	6D	4	3	65	3	2							61	6	2	71	5	2	75	4	2	7D	4	3	79	4	3										✓	✓	✓	-	-	✓
AND	A ∧ M → A (11)	29	2	2	2D	4	3	25	3	2							21	6	2	31	5	2	35	4	2	3D	4	3	39	4	3										✓	✓	-	-	-	-
ASL	C ← 7 0 ← 0				0E	6	3	06	5	2	0A	2	1										16	6	2	1E	7	3													✓	✓	✓	-	-	-
BCC	BRANCH ON C=0 (2)																															90	2	2							-	-	-	-	-	-
BCS	BRANCH ON C=1 (2)																															B0	2	2							-	-	-	-	-	-
BEQ	BRANCH ON Z=1 (2)																															F0	2	2							-	-	-	-	-	-
BIT	A ∧ M				2C	4	3	24	3	2																															M7	✓	-	-	-	M6
BMI	BRANCH ON N=1 (2)																															30	2	2							-	-	-	-	-	-
BNE	BRANCH ON Z=0 (2)																															D0	2	2							-	-	-	-	-	-
BPL	BRANCH ON N=0 (2)																															10	2	2							-	-	-	-	-	-
BRK	(See Fig. 11)													00	7	1																									-	-	-	✓	-	-
BVC	BRANCH ON V=0 (2)																															50	2	2							-	-	-	-	-	-
BVS	BRANCH ON V=1 (2)																															70	2	2							-	-	-	-	-	-
CLC	0 → C													18	2	1																									-	-	0	-	-	-
CLD	0 → D													D8	2	1																									-	-	-	-	0	-
CLI	0 → I													58	2	1																									-	-	-	0	-	-
CLV	0 → V													B8	2	1																									-	-	-	-	-	0
CMP	A - M (1)	C9	2	2	CD	4	3	C5	3	2							C1	6	2	D1	5	2	D5	4	2	DD	4	3	D9	4	3										✓	✓	✓	-	-	-
CPX	X - M	E0	2	2	EC	4	3	E4	3	2																															✓	✓	✓	-	-	-
CPY	Y - M	C0	2	2	CC	4	3	C4	3	2																															✓	✓	✓	-	-	-
DEC	M - 1 → M				CE	6	3	C6	5	2													D6	6	2	DE	7	3													✓	✓	-	-	-	-
DEX	X - 1 → X													CA	2	1																									✓	✓	-	-	-	-
DEY	Y - 1 → Y													88	2	1																									✓	✓	-	-	-	-
EOR	A ⊻ M → A (11)	49	2	2	4D	4	3	45	3	2							41	6	2	51	5	2	55	4	2	5D	4	3	59	4	3										✓	✓	-	-	-	-
INC	M + 1 → M				EE	6	3	E6	5	2													F6	6	2	FE	7	3													✓	✓	-	-	-	-
INX	X + 1 → X													E8	2	1																									✓	✓	-	-	-	-
INY	Y + 1 → Y													C8	2	1																									✓	✓	-	-	-	-
JMP	JUMP TO NEW LOC				4C	3	3																													6C	5	3				-	-	-	-	-	-
JSR	(See Fig. 2) JUMP SUB				20	6	3																																		-	-	-	-	-	-
LDA	M → A (11)	A9	2	2	AD	4	3	A5	3	2							A1	6	2	B1	5	2	B5	4	2	BD	4	3	B9	4	3										✓	✓	-	-	-	-

INSTRUCTIONS		IMMEDIATE			ABSOLUTE			ZERO PAGE			ACCUM			IMPLIED			INDX X			INDX Y			Z PAGE X			ABS X			ABS Y			RELATIVE			INDIRECT			Z PAGE Y			CONDITION CODES					
MNEMONIC	OPERATION	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	OP	N	#	N	Z	C	I	D	V
LDX	M → X (11)	A2	2	2	AE	4	3	A6	3	2																			BE	4	3										✓	✓	-	-	-	-
LDY	M → Y (11)	A0	2	2	AC	4	3	A4	3	2													B4	4	2	BC	4	3													✓	✓	-	-	-	-
LSR	0 → 7 0 → C				4E	6	3	46	5	2	4A	2	1										56	6	2	5E	7	3													0	✓	✓	-	-	-
NOP	NO OPERATION													EA	2	1																									-	-	-	-	-	-
ORA	A ∨ M → A	09	2	2	0D	4	3	05	3	2							01	6	2	11	5	2	15	4	2	1D	4	3	19	4	3										✓	✓	-	-	-	-
PHA	A → Ms S-1 → S													48	3	1																									-	-	-	-	-	-
PHP	P → Ms S-1 → S													08	3	1																									-	-	-	-	-	-
PLA	S + 1 → S Ms → A													68	4	1																									✓	✓	-	-	-	-
PLP	S + 1 → S Ms → P													28	4	1																									(RESTORED)					
ROL	7 0 C				2E	6	3	26	5	2	2A	2	1										36	6	2	3E	7	3													✓	✓	✓	-	-	-
ROR	C 7 0 C				6E	6	3	66	5	2	6A	2	1										76	6	2	7E	7	3													✓	✓	✓	-	-	-
RTI	(See Fig. 11) RTRN INT													40	6	1																									(RESTORED)					
RTS	(See Fig. 2) RTRN SUB													60	6	1																									-	-	-	-	-	-
SBC	A - M - C̄ → A (11)	E9	2	2	ED	4	3	E5	3	2							E1	6	2	F1	5	2	F5	4	2	FD	4	3	F9	4	3										✓	✓	(3)	-	-	✓
SEC	1 → C													38	2	1																									-	-	1	-	-	-
SED	1 → D													F8	2	1																									-	-	-	-	1	-
SEI	1 → I													78	2	1																									-	-	-	1	-	-
STA	A → M				8D	4	3	85	3	2							81	6	2	91	6	2	95	4	2	9D	5	3	99	5	3										-	-	-	-	-	-
STX	X → M				8E	4	3	86	3	2																															96	4	2	-	-	-	-	-	-
STY	Y → M				8C	4	3	84	3	2													94	4	2																	-	-	-	-	-	-
TAX	A → X													AA	2	1																									✓	✓	-	-	-	-
TAY	A → Y													A8	2	1																									✓	✓	-	-	-	-
TSX	S → X													BA	2	1																									✓	✓	-	-	-	-
TXA	X → A													8A	2	1																									✓	✓	-	-	-	-
TXS	X → S													9A	2	1																									-	-	-	-	-	-
TYA	Y → A													98	2	1																									✓	✓	-	-	-	-

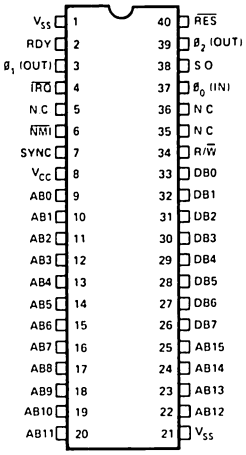
(1) ADD 1 TO "N" IF PAGE BOUNDARY IS CROSSED
(2) ADD 1 TO "N" IF BRANCH OCCURS TO SAME PAGE
ADD 2 TO "N" IF BRANCH OCCURS TO DIFFERENT PAGE
(3) CARRY NOT = BELOW
(4) IF IN DECIMAL MODE Z FLAG IS INVALID
ACCUMULATOR MUST BE CHECKED FOR ZERO RESULT

X INDEX X
Y INDEX Y
A ACCUMULATOR
M MEMORY PER EFFECTIVE ADDRESS
Ms MEMORY PER STACK POINTER

+ ADD
-- SUBTRACT
∧ AND
∨ OR
⊻ EXCLUSIVE OR
✓ MODIFIED

- NOT MODIFIED
M7 MEMORY BIT 7
M6 MEMORY BIT 6
N NO CYCLES
NO BYTES

SY6502 – 40 Pin Package



Features

- 65K Addressable Bytes of Memory
- $\overline{IRQ}$ Interrupt
- $\overline{NMI}$ Interrupt
- On-the-chip Clock
  - √ TTL Level Single Phase Input
  - √ Crystal Time Base Input
- SYNC Signal
  (can be used for single instruction execution)
- RDY Signal
  (can be used for single cycle execution)
- Two Phase Output Clock for Timing of Support Chips

APPENDIX F

6551 ACIA

This Appendix contains information about the 6651's architecture, operation and interface capabilities.

Thank you to Synertek for permission to reprint their material.

Asynchronous Communication Interface Adapter

Synertek

SY6551

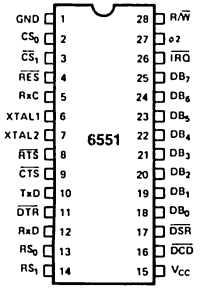
MICROPROCESSOR PRODUCTS

DECEMBER 1980

- On-chip baud rate generator: 15 programmable baud rates derived from a standard 1.8432 MHz external crystal (50 to 19,200 baud).
- Programmable interrupt and status register to simplify software design.
- Single +5 volt power supply.
- Serial echo mode.
- False start bit detection.
- 8-bit bi-directional data bus for direct communication with the microprocessor.
- External 16x clock input for non-standard baud rates (up to 125 Kbaud).
- Programmable: word lengths; number of stop bits; and parity bit generation and detection.
- Data set and modem control signals provided.
- Parity: (odd, even, none, mark, space).
- Full-duplex or half-duplex operation.
- 5, 6, 7, 8 and 9 bit transmission.

The SY6551 is an Asynchronous Communication Adapter (ACIA) intended to provide for interfacing the 6500/6800 microprocessor families to serial communication data sets and modems. A unique feature is the inclusion of an on-chip programmable baud rate generator, with a crystal being the only external component required.

PIN CONFIGURATION



ORDERING INFORMATION

Part No.	Package	Clock Rate
SYC6551	Ceramic	1 MHz
SYP6551	Plastic	1 MHz
SYC6551A	Ceramic	2 MHz
SYP6551A	Plastic	2 MHz

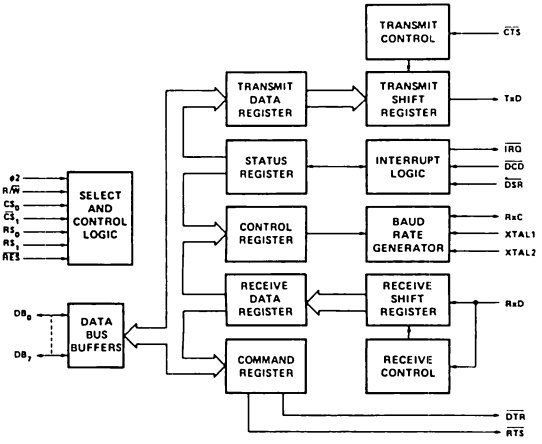


Figure 1. Block Diagram

ABSOLUTE MAXIMUM RATINGS

Rating	Symbol	Allowable Range
Supply Voltage	V_{CC}	-0.3V to +7.0V
Input/Output Voltage	V_{IN}	-0.3V to +7.0V
Operating Temperature	T_{OP}	0°C to 70°C
Storage Temperature	T_{STG}	-55°C to 150°C

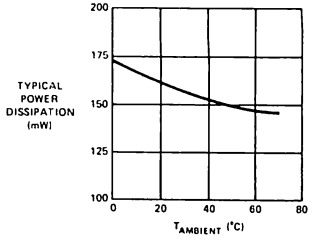
All inputs contain protection circuitry to prevent damage to high static charges. Care should be exercised to prevent unnecessary application of voltages in excess of the allowable limits.

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0V ± 5%, T_A = 0-70°C, unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Input High Voltage	V_{IH}	2.0	–	V_{CC}	V
Input Low Voltage	V_{IL}	-0.3	–	0.8	V
Input Leakage Current: V_{IN} = 0 to 5V (ϕ2, R/$\overline{W}$, $\overline{RES}$, CS_0, $\overline{CS_1}$, RS_0, RS_1, $\overline{CTS}$, RxD, $\overline{DCD}$, $\overline{DSR}$)	I_{IN}	–	±1.0	±2.5	μA
Input Leakage Current for High Impedance State (Three State)	I_{TSI}	–	±2.0	±10.0	μA
Output High Voltage: I_{LOAD} = -100μA (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$)	V_{OH}	2.4	–	–	V
Output Low Voltage: I_{LOAD} = 1.6mA (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$, $\overline{IRQ}$)	V_{OL}	–	–	0.4	V
Output High Current (Sourcing): V_{OH} = 2.4V (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$)	I_{OH}	-250	–	–	μA
Output Low Current (Sinking): V_{OL} = 0.4V (DB_0 - DB_7, TxD, RxC, $\overline{RTS}$, $\overline{DTR}$, $\overline{IRQ}$)	I_{OL}	1.6	–	–	mA
Output Leakage Current (Off State): V_{OUT} = 5V ($\overline{IRQ}$)	I_{OFF}	–	1.0	10.0	μA
Clock Capacitance (ϕ2)	C_{CLK}	–	–	20	pF
Input Capacitance (Except XTAL1 and XTAL2)	C_{IN}	–	–	10	pF
Output Capacitance	C_{OUT}	–	–	10	pF
Power Dissipation (See Graph) (T_A = 0°C)	P_D	–	170	300	mW

POWER DISSIPATION vs TEMPERATURE



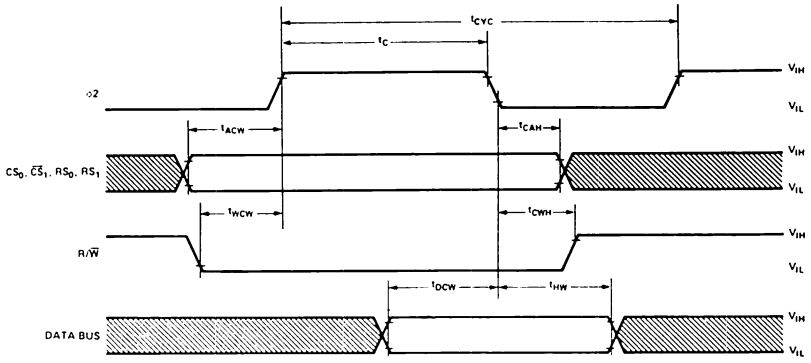


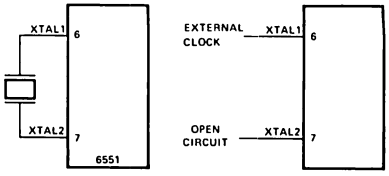
Figure 2. Write Timing Characteristics

WRITE CYCLE (V_{CC} = 5.0V ± 5%, T_A = 0 to 70°C, unless otherwise noted)

Characteristic	Symbol	SY6551		SY6551A		Unit
		Min	Max	Min	Max	
Cycle Time	t_{CYC}	1.0	–	0.5	–	μs
$\phi 2$ Pulse Width	t_C	400	–	200	–	ns
Address Set-Up Time	t_{ACW}	120	–	70	–	ns
Address Hold Time	t_{CAH}	0	–	0	–	ns
R/$\overline{W}$ Set-Up Time	t_{WCW}	120	–	70	–	ns
R/$\overline{W}$ Hold Time	t_{CWH}	0	–	0	–	ns
Data Bus Set-Up Time	t_{DCW}	150	–	60	–	ns
Data Bus Hold Time	t_{HW}	20	–	20	–	ns

(t_r and t_f = 10 to 30 ns)

CLOCK GENERATION



INTERNAL CLOCK EXTERNAL CLOCK

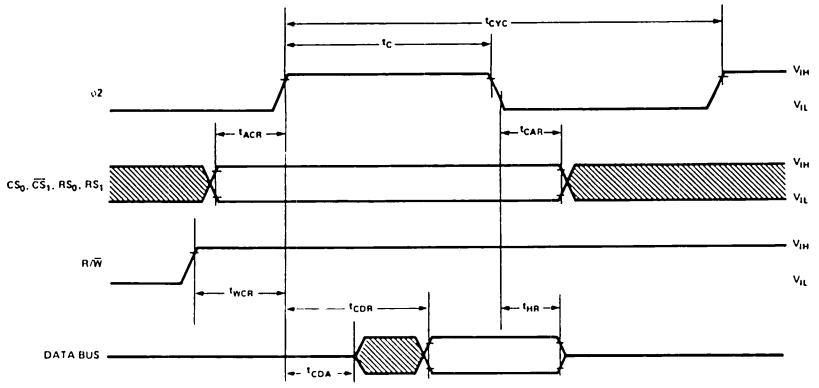
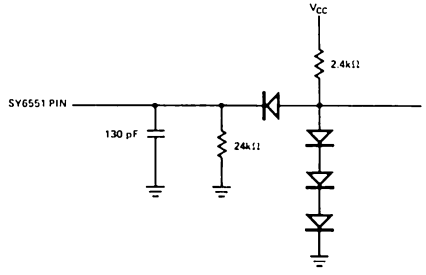


Figure 3. Read Timing Characteristics

READ CYCLE (V_{CC} = 5.0V ± 5%, T_A = 0 to 70°C, unless otherwise noted)

Characteristic	Symbol	SY6551 Min	SY6551 Max	SY6551A Min	SY6551A Max	Unit
Cycle Time	t_{CYC}	1.0	–	0.5	–	μs
Pulse Width (ϕ2)	t_C	400	–	200	–	ns
Address Set-Up Time	t_{ACR}	120	–	70	–	ns
Address Hold Time	t_{CAR}	0	–	0	–	ns
R/$\overline{W}$ Set-Up Time	t_{WCR}	120	–	70	–	ns
Read Access Time (Valid Data)	t_{CDR}	–	200	–	150	ns
Read Hold Time	t_{HR}	20	–	20	–	ns
Bus Active Time (Invalid Data)	t_{CDA}	40	–	40	–	ns



TEST LOAD FOR DATA BUS (DB_0 – DB_7), $\overline{TxD}$, $\overline{DTR}$, RTS OUTPUTS

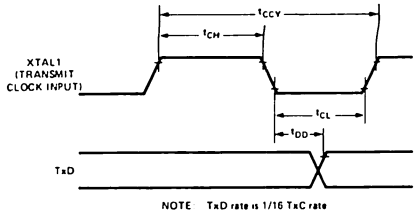


Figure 4a. Transmit Timing with External Clock

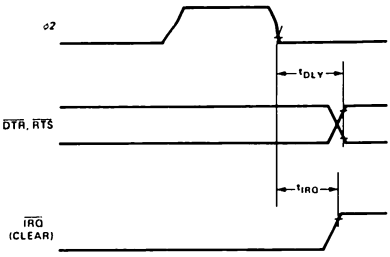


Figure 4b. Interrupt and Output Timing

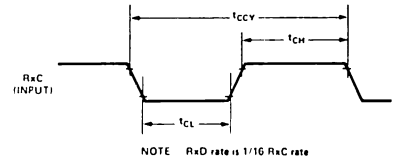


Figure 4c. Receive External Clock Timing

TRANSMIT/RECEIVE CHARACTERISTICS

Characteristic	Symbol	SY6551 Min	SY6551 Max	SY6551A Min	SY6551A Max	Unit
Transmit/Receive Clock Rate	t_{CCY}	400*	–	400*	–	ns
Transmit/Receive Clock High Time	t_{CH}	175	–	175	–	ns
Transmit/Receive Clock Low Time	t_{CL}	175	–	175	–	ns
XTAL1 to TxD Propagation Delay	t_{DD}	–	500	–	500	ns
Propagation Delay ($\overline{RTS}$, $\overline{DTR}$)	t_{DLY}	–	500	–	500	ns
$\overline{IRQ}$ Propagation Delay (Clear)	t_{IRQ}	–	500	–	500	ns

(t_r, t_f = 10 to 30 nsec)

*The baud rate with external clocking is: Baud Rate = $\frac{1}{16 \times T_{CCY}}$

INTERFACE SIGNAL DESCRIPTION

$\overline{RES}$ (Reset)

During system initialization a low on the $\overline{RES}$ input will cause internal registers to be cleared.

ϕ2 (Input Clock)

The input clock is the system ϕ2 clock and is used to trigger all data transfers between the system microprocessor and the SY6551.

R/$\overline{W}$ (Read/Write)

The R/$\overline{W}$ is generated by the microprocessor and is used to control the direction of data transfers. A high on the R/$\overline{W}$ pin allows the processor to read the data supplied by the SY6551. A low on the R/$\overline{W}$ pin allows a write to the SY6551.

$\overline{IRQ}$ (Interrupt Request)

The $\overline{IRQ}$ pin is an interrupt signal from the interrupt control logic. It is an open drain output, permitting several devices to be connected to the common $\overline{IRQ}$ microprocessor input. Normally a high level, $\overline{IRQ}$ goes low when an interrupt occurs.

DB_0 - DB_7 (Data Bus)

The DB_0-DB_7 pins are the eight data lines used for transfer of data between the processor and the SY6551. These lines are bi-directional and are normally high-impedance except during Read cycles when selected.

CS_0, $\overline{CS_1}$ (Chip Selects)

The two chip select inputs are normally connected to the processor address lines either directly or through decoders. The SY6551 is selected when CS_0 is high and $\overline{CS_1}$ is low.

RS_ϕ, RS_1 (Register Selects)

The two register select lines are normally connected to the processor address lines to allow the processor to select the various SY6551 internal registers. The following table indicates the internal register select coding:

RS_1	RS_0	Write	Read
0	0	Transmit Data Register	Receiver Data Register
0	1	Programmed Reset (Data is "Don't Care")	Status Register
1	0	Command Register	
1	1	Control Register	

The table shows that only the Command and Control registers are read/write. The Programmed Reset operation does not cause any data transfer, but is used to clear the SY6551 registers. The Programmed Reset is slightly different from the Hardware Reset ($\overline{RES}$) and these differences are described in the individual register definitions.

ACIA/MODEM INTERFACE SIGNAL DESCRIPTION

XTAL1, XTAL2 (Crystal Pins)

These pins are normally directly connected to the external crystal (1.8432 MHz) used to derive the various baud rates. Alternatively, an externally generated clock may be used to drive the XTAL1 pin, in which case the XTAL2 pin must float. The choice of crystal is not critical, but CTS Knights MPO18 is recommended.

TxD (Transmit Data)

The TxD output line is used to transfer serial NRZ (non-return-to-zero) data to the modem. The LSB (least significant bit) of the Transmit Data Register is the first data bit transmitted and the rate of data transmission is determined by the baud rate selected.

RxD (Receive Data)

The RxD input line is used to transfer serial NRZ data into the ACIA from the modem, LSB first. The receiver data rate is either the programmed baud rate or the rate of an externally generated receiver clock. This selection is made by programming the Control Register.

RxC (Receive Clock)

The RxC is a bi-directional pin which serves as either the receiver 16x clock input or the receiver 16x clock output. The latter mode results if the internal baud rate generator is selected for receiver data clocking.

$\overline{RTS}$ (Request to Send)

The $\overline{RTS}$ output pin is used to control the modem from the processor. The state of the $\overline{RTS}$ pin is determined by the contents of the Command Register.

$\overline{CTS}$ (Clear to Send)

The $\overline{CTS}$ input pin is used to control the transmitter operation. The enable state is with $\overline{CTS}$ low. The transmitter is automatically disabled if $\overline{CTS}$ is high.

$\overline{DTR}$ (Data Terminal Ready)

This output pin is used to indicate the status of the SY6551 to the modem. A low on $\overline{DTR}$ indicates the SY6551 is enabled and a high indicates it is disabled. The processor controls this pin via bit 0 of the Command Register.

$\overline{DSR}$ (Data Set Ready)

The $\overline{DSR}$ input pin is used to indicate to the SY6551 the status of the modem. A low indicates the "ready" state and a high, "not-ready." $\overline{DSR}$ is a high-impedance input and must not be a no-connect. If unused, it should be driven high or low, but not switched.

Note: If Command Register Bit 0 = 1 and a change of state on $\overline{DSR}$ occurs, $\overline{IRQ}$ will be set, and Status Register Bit 6 will reflect the new level. The state of $\overline{DSR}$ does not affect either Transmitter or Receiver operation.

$\overline{DCD}$ (Data Carrier Detect)

The $\overline{DCD}$ input pin is used to indicate to the SY6551 the status of the carrier-detect output of the modem. A low indicates that the modem carrier signal is present and a high, that it is not. $\overline{DCD}$, like $\overline{DSR}$, is a high-impedance input and must not be a no-connect.

Note: If Command Register Bit 0 = 1 and a change of state on $\overline{DCD}$ occurs, $\overline{IRQ}$ will be set, and Status Register Bit 5 will reflect the new level. The state of $\overline{DCD}$ does not affect Transmitter operation, but must be low for the Receiver to operate.

INTERNAL ORGANIZATION

The Transmitter/Receiver sections of the SY6551 are depicted by the block diagram in Figure 5.

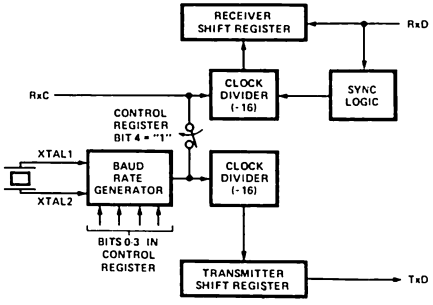


Figure 5. Transmitter/Receiver Clock Circuits

Bits 0-3 of the Control Register select the divisor used to generate the baud rate for the Transmitter. If the Receiver clock is to use the same baud rate as the Transmitter, then RxC becomes an output pin and can be used to slave other circuits to the SY6551.

CONTROL REGISTER

The Control Register is used to select the desired mode for the SY6551. The word length, number of stop bits, and clock controls are all determined by the Control Register, which is depicted in Figure 6.

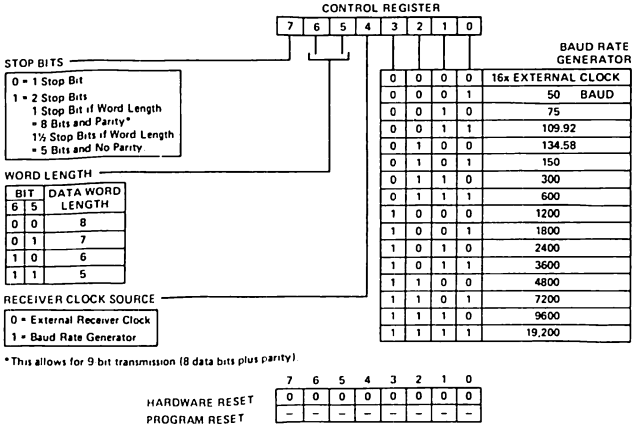


Figure 6. Control Register Format

COMMAND REGISTER

The Command Register is used to control Specific Transmit/Receive functions and is shown in Figure 7.

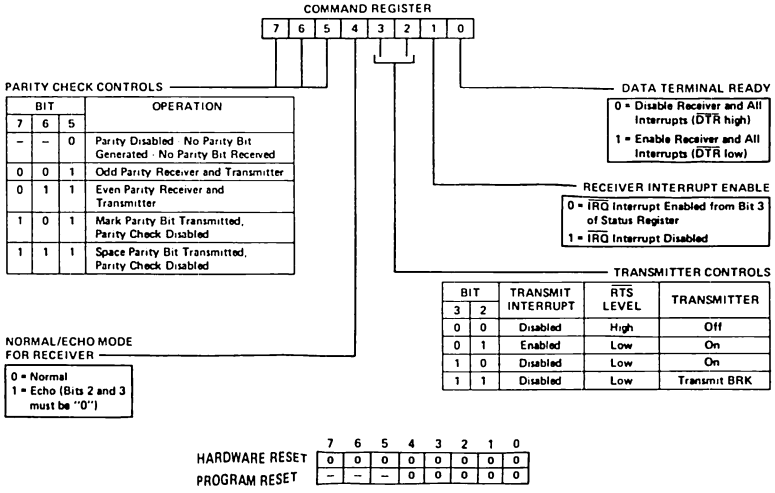
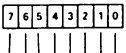


Figure 7. Command Register Format

STATUS REGISTER

The Status Register is used to indicate to the processor the status of various SY6551 functions and is outlined in Figure 8.



STATUS	SET BY	CLEARED BY
Parity Error*	0 = No Error 1 = Error	Self Clearing**
Framing Error*	0 = No Error 1 = Error	Self Clearing**
Overrun*	0 = No Error 1 = Error	Self Clearing**
Receive Data Register Full	0 = Not Full 1 = Full	Read Receive Data Register
Transmit Data Register Empty	0 = Not Empty 1 = Empty	Write Transmit Data Register
$\overline{DCD}$	0 = $\overline{DCD}$ Low 1 = $\overline{DCD}$ High	Not Resettable Reflects $\overline{DCD}$ State
$\overline{DSR}$	0 = $\overline{DSR}$ Low 1 = $\overline{DSR}$ High	Not Resettable Reflects $\overline{DSR}$ State
IRQ	0 = No Interrupt 1 = Interrupt	Read Status Register

*NO INTERRUPT GENERATED FOR THESE CONDITIONS.
**CLEARED AUTOMATICALLY AFTER A READ OF RDR AND THE NEXT ERROR FREE RECEIPT OF DATA

	7	6	5	4	3	2	1	0
HARDWARE RESET	0	–	–	1	0	0	0	0
PROGRAM RESET	–	–	–	–	–	0	–	–

Figure 8. Status Register Format

TRANSMIT AND RECEIVE DATA REGISTERS

These registers are used as temporary data storage for the 6551 Transmit and Receive circuits. The Transmit Data Register is characterized as follows:

- Bit 0 is the leading bit to be transmitted.
- Unused data bits are the high-order bits and are "don't care" for transmission.

The Receive Data Register is characterized in a similar fashion:

- Bit 0 is the leading bit received.
- Unused data bits are the high-order bits and are "0" for the receiver.
- Parity bits are not contained in the Receive Data Register, but are stripped-off after being used for external parity checking. Parity and all unused high-order bits are "0".

Figure 9 illustrates a single transmitted or received data word, for the example of 8 data bits, parity, and 1 stop bit.

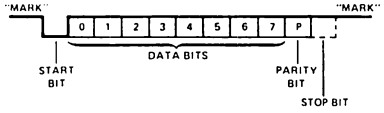
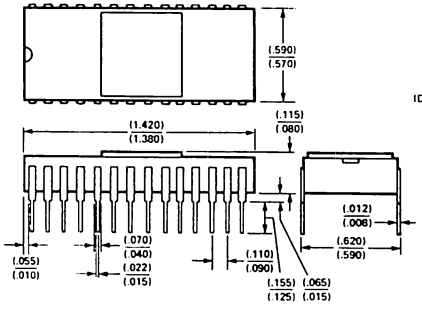


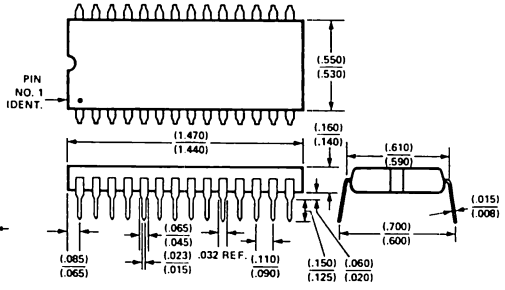
Figure 9. Serial Data Stream Example

PACKAGE OUTLINES

28 LEAD CERAMIC



28 LEAD PLASTIC



APPENDIX G

Z-80 MICROPROCESSOR

This Appendix contains information about the Z-80 microprocessor's architecture, operation and instruction set.

Z8400
Z80® CPU Central Processing Unit

Zilog

Product Specification

March 1981

Features

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Six MHz, 4 MHz and 2.5 MHz clocks for the Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high speed interrupt processing: 8080 compatible, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

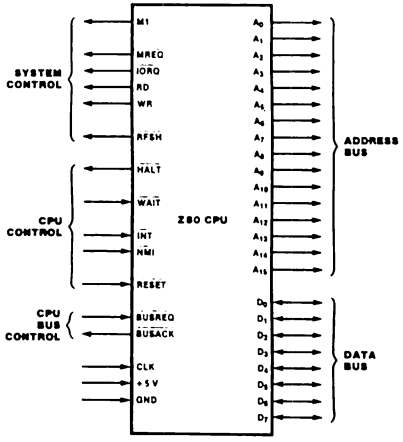


Figure 1. Pin Functions

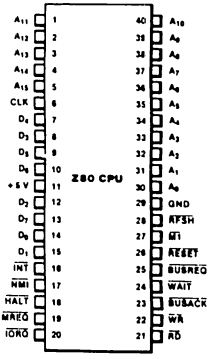


Figure 2. Pin Assignments

General Description

The Z80, Z80A, and Z80B CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second- and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source, all output signals are fully decoded and timed to control standard memory or peripheral circuits, and is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

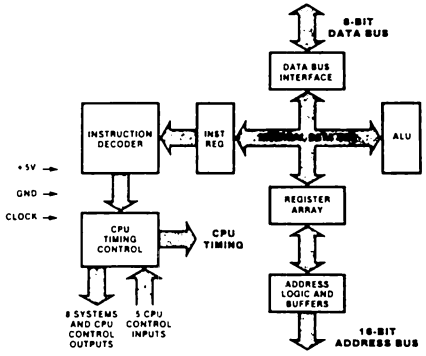


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

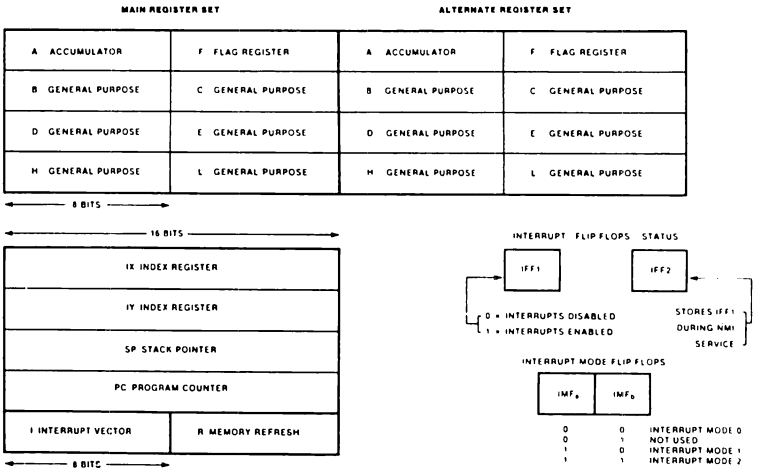


Figure 4. CPU Registers

Z80 CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Automatically incremented and placed on the address bus during each instruction fetch cycle.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Stores addresses or data temporarily. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{NMI}$ and $\overline{INT}$. The $\overline{NMI}$ is a non-maskable interrupt and has the highest priority. $\overline{INT}$ is a lower priority interrupt since it requires that interrupts be enabled in software in order to operate. Either $\overline{NMI}$ or $\overline{INT}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{INT}$, has three programmable response modes available. These are:

- Mode 0 — compatible with the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt ($\overline{NMI}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at at all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in a normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request and to place the interrupt vector on the bus.

Mode 0 Interrupt Operation. This mode is compatible with the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus, which is then acted on six times by the CPU. This is normally a Restart Instruction, which will initiate an unconditional jump to the selected one of eight restart locations in page zero of memory.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has a vector address of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit address vector on the data bus during the interrupt acknowledge cycle. The high-order byte of the interrupt service routine address is supplied by the I (Interrupt) register. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwared to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution	•	•	IFF_2 – Parity flag
LD A,R instruction execution	•	•	IFF_2 – Parity flag
Accept $\overline{NMI}$	0	IFF_1	IFF_1 – IFF_2 (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2	•	IFF_2 – IFF_1 at completion of an $\overline{NMI}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- 8-bit loads
- 16-bit loads
- Exchanges, block transfers, and searches
- 8-bit arithmetic and logic operations
- General-purpose arithmetic and CPU control
- 16-bit arithmetic operations
- Rotates and shifts
- Bit set, reset, and test operations
- Jumps
- Calls, returns, and restarts
- Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- Immediate
- Immediate extended
- Modified page zero
- Relative
- Extended
- Indexed
- Register
- Register indirect
- Implied
- Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r ← r'	•	•	X	X	•	•	01 r r'		1	1	4	r, r' Reg
LD r, n	r ← n	•	•	X	X	•	•	00 r 110		2	2	7	000 B
								— n —					001 C
LD r, (HL)	r ← (HL)	•	•	X	X	•	•	01 r 110		1	2	7	010 D
LD r, (IX + d)	r ← (IX + d)	•	•	X	X	•	•	11 011 101	DD	3	5	19	011 E
								01 r 101					100 H
								— d —					101 L
LD r (IY + d)	r ← (IY + d)	•	•	X	X	•	•	11 111 101	FD	3	5	19	111 A
								01 r 110					
								— d —					
LD (HL), r	(HL) ← r	•	•	X	X	•	•	01 110 r		1	2	7	
LD (IX + d), r	(IX + d) ← r	•	•	X	X	•	•	11 011 101	DD	3	5	19	
								01 110 r					
								— d —					
LD (IY + d), r	(IY + d) ← r	•	•	X	X	•	•	11 111 101	FD	3	5	19	
								01 110 r					
								— d —					
LD (HL), n	(HL) ← n	•	•	X	X	•	•	00 110 110	36	2	3	10	
								— n —					
LD (IX + d), n	(IX + d) ← n	•	•	X	X	•	•	11 011 101	DD	4	5	19	
								00 110 110	36				
								— d —					
								— n —					
LD (IY + d), n	(IY + d) ← n	•	•	X	X	•	•	11 111 101	FD	4	5	19	
								00 110 110	36				
								— d —					
								— n —					
LD A, (BC)	A ← (BC)	•	•	X	X	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	X	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	X	•	•	00 111 010	3A	3	4	13	
								— n —					
								— n —					
LD (BC), A	(BC) ← A	•	•	X	X	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	X	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	X	•	•	00 110 010	32	3	4	13	
								— n —					
								— n —					
LD A, I	A ← I	↕	↕	X	0	X	IFF	11 101 101	ED	2	2	9	
								01 010 111	57				
LD A, R	A ← R	↕	↕	X	0	X	IFF	11 101 101	ED	2	2	9	
								01 011 111	5F				
LD I, A	I ← A	•	•	X	•	X	•	11 101 101	ED	2	2	9	
								01 000 111	47				
LD R, A	R ← A	•	•	X	•	X	•	11 101 101	ED	2	2	9	
								01 001 111	4F				

NOTES: r, r' means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip flop (IFF) is copied into the P/V flag
For an explanation of flag notation and symbols for mnemonic tables see Symbolic Notation section following tables

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n ← ← n ←	3	3	10	dd Pair 00 BC 01 DE
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 001 21 ← n ← ← n ←	4	4	14	10 HL 11 SP
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 001 21 ← n ← ← n ←	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 2A ← n ← ← n ←	3	5	16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 ED 01 dd1 011 ← n ← ← n ←	4	6	20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 DD 00 101 010 2A ← n ← ← n ←	4	6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 FD 00 101 010 2A ← n ← ← n ←	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 22 ← n ← ← n ←	3	5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 ED 01 dd0 011 ← n ← ← n ←	4	6	20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 010 22 ← n ← ← n ←	4	6	20	
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 010 22 ← n ← ← n ←	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001 F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 DD 11 111 001 F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 FD 11 111 001 F9	2	2	10	
PUSH qq	(SP - 2) ← qq_L (SP - 1) ← qq_H SP ← SP - 2	•	•	X	•	X	•	•	•	11 qq0 101	1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP - 2) ← IX_L (SP - 1) ← IX_H SP ← SP - 2	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 101 E5	2	4	15	
PUSH IY	(SP - 2) ← IY_L (SP - 1) ← IY_H SP ← SP - 2	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 101 E5	2	4	15	
POP qq	qq_H ← (SP + 1) qq_L ← (SP) SP ← SP + 2	•	•	X	•	X	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP ← SP + 2	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 001 E1	2	4	14	
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP ← SP + 2	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 001 E1	2	4	14	

NOTES dd is any of the register pairs BC DE HL SP
qq is any of the register pairs AF BC DE HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively
e.g. BC_L = C AF_H = A

Exchange, Block Transfer, Block Search Groups

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ← HL	•	•	X	•	X	•	•	•	11 101 011 EB	1	1	4	
EX AF, AF'	AF ← AF'	•	•	X	•	X	•	•	•	00 001 000 08	1	1	4	
EXX	BC ← BC' DE ← DE' HL ← HL'	•	•	X	•	X	•	•	•	11 011 001 D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ← (SP + 1) L ← (SP)	•	•	X	•	X	•	•	•	11 100 011 E3	1	5	19	
EX (SP), IX	IX_H ← (SP + 1) IX_L ← (SP)	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 011 E3	2	6	23	
EX (SP), IY	IY_H ← (SP + 1) IY_L ← (SP)	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1	•	•	X	①	X	0	0	•	11 101 101 ED 10 100 000 A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 ED 10 110 000 B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE ① P/V flag is 0 if the result of BC - 1 = 0, otherwise P/V = 1

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LDD	(DE) ← (HL) DE ← DE - 1 HL ← HL - 1 BC ← BC - 1	•	•	X 0	X	① 1	0	•	11 101 101 ED 10 101 000 A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE - 1 HL ← HL - 1 BC ← BC - 1 Repeat until BC = 0	•	•	X 0	X	0	0	•	11 101 101 ED 10 111 000 B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A - (HL) HL ← HL + 1 BC ← BC - 1	② ↕	↕	X ↕	X	① 1	↕	•	11 101 101 ED 10 100 001 A1	2	4	16	
CPIR	A - (HL) HL ← HL + 1 BC ← BC - 1 Repeat until A = (HL) or BC = 0	② ↕	↕	X ↕	X	① 1	↕	•	11 101 101 ED 10 110 001 B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A - (HL) HL ← HL - 1 BC ← BC - 1	② ↕	↕	X ↕	X	① 1	↕	•	11 101 101 ED 10 101 001 A9	2	4	16	
CPDR	A - (HL) HL ← HL - 1 BC ← BC - 1 Repeat until A = (HL) or BC = 0	② ↕	↕	X ↕	X	① 1	↕	•	11 101 101 ED 10 111 001 B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES: ① P/V flag is 0 if the result of BC - 1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0

8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	↕	↕	X ↕	X	V 0	↕	10 000 r	1	1	4	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X ↕	X	V 0	↕	11 000 110 — n —	2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X ↕	X	V 0	↕	10 000 110	1	2	7	
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X ↕	X	V 0	↕	11 011 101 DD 10 000 110 — d —	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X ↕	X	V 0	↕	11 111 101 FD 10 000 110 — d —	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	X ↕	X	V 0	↕	001				s is any of r, n (HL), (IX + d) (IY + d) as shown for ADD instruction The indicated bits replace the 000 in the ADD set above
SUB s	A ← A - s	↕	↕	X ↕	X	V 1	↕	010				
SBC A, s	A ← A - s - CY	↕	↕	X ↕	X	V 1	↕	011				
AND s	A ← A ∧ s	↕	↕	X 1	X	P 0	0	100				
OR s	A ← A ∨ s	↕	↕	X 0	X	P 0	0	110				
XOR s	A ← A ● s	↕	↕	X 0	X	P 0	0	101				
CP s	A - s	↕	↕	X ↕	X	V 1	↕	111				
INC r	r ← r + 1	↕	↕	X ↕	X	V 0	•	00 r 100	1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X ↕	X	V 0	•	00 110 100	1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X ↕	X	V 0	•	11 011 101 DD 00 110 100 — d —	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X ↕	X	V 0	•	11 111 101 FD 00 110 100 — d —	3	6	23	
DEC m	m ← m - 1	↕	↕	X ↕	X	V 1	•	101				m is any of r, (HL) (IX + d), (IY + d) as shown for INC DEC same format and states as INC Replace 100 with 101 in opcode

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	P	•	↕	00 100 111 27	1	1	4	Decimal adjust accumulator
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111 2F	1	1	4	Complement accumulator (one's complement)
NEG	A ← 0 − A	↕	↕	X	↕	X	V	1	↕	11 101 101 ED 01 000 100 44	2	2	8	Negate acc. (two's complement)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111 3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111 37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000 00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110 76	1	1	4	
DI	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011 F3	1	1	4	
EI	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011 FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 110 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 ED 01 010 110 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 ED 01 011 110 5E	2	2	8	

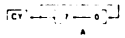
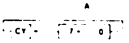
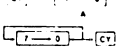
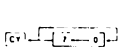
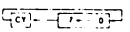
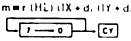
NOTES: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop
• indicates interrupts are not sampled at the end of EI or DI

16-Bit Arithmetic Group

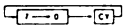
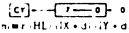
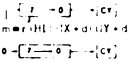
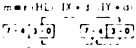
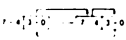
Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001	1	3	11	ss Reg 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 ED 01 ss1 010	2	4	15	
SBC HL, ss	HL ← HL − ss − CY	↕	↕	X	X	X	V	1	↕	11 101 101 ED 01 ss0 010	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 DD 00 ppl 001	2	4	15	pp Reg 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 FD 00 rr1 001	2	4	15	rr Reg 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011	1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 011 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 011 23	2	2	10	
DEC ss	ss ← ss − 1	•	•	X	•	X	•	•	•	00 ss1 011	1	1	6	
DEC IX	IX ← IX − 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 101 011 2B	2	2	10	
DEC IY	IY ← IY − 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 101 011 2B	2	2	10	

NOTE: [illegible]

Rotate and Shift Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111 07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	↕	00 010 111 17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111 0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111 1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 r	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 110 06	2	4	15	r Reg 111 A 000 B 001 C 010 D 011 E 100 H 101 L
RLC (IX + d) r ← (HL) (IX + d) (IY + d)		↕	↕	X	0	X	P	0	↕	11 011 101 DD 11 001 011 CB ← d → 00 000 110 06	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 FD 11 001 011 CB ← d → 00 000 110 06	4	6	23	
RL m m = r, (HL), (IX + d), (IY + d)		↕	↕	X	0	X	P	0	↕	010				Instruction format and states are as shown for RLC's. To form new opcode replace 000 of RLC's with shown code
RRC m m = r, (HL), (IX + d), (IY + d)		↕	↕	X	0	X	P	0	↕	001				

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
RR m	 m = r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	11 001 011 – d – 00 011 m	CB	2	4	15	
SLA m	 m = r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	11 001 011 – d – 00 100 m	CB	2	4	15	
SRA m	 m = r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	11 001 011 – d – 00 101 m	CB	2	4	15	
SRL m	 m = r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	11 001 011 – d – 00 111 m	CB	2	4	15	
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected.
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z = \overline{r_b}$	X	↕	X	↕	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A b: 000 0, 001 1, 010 2, 011 3, 100 4, 101 5, 110 6, 111 7
BIT b, (HL)	$Z = \overline{(HL)_b}$	X	↕	X	↕	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX + d)	$Z = \overline{(IX+d)_b}$	X	↕	X	↕	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, (IY + d)	$Z = \overline{(IY+d)_b}$	X	↕	X	↕	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	
SET b, r	$r_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b = 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX + d)	$(IX+d)_b = 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – 11 b 110	DD CB	4	6	23	
SET b, (IY + d)	$(IY+d)_b = 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – 11 b 110	FD CB	4	6	23	
RES b, m	$m_b = 0$ m = r, (HL) (IX + d) (IY + d)	•	•	X	•	X	•	•	•	10					To form new opcode replace 11 of SET b, m with 10. Flags and time states for SET instruction.

Jump Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 – n – – n –	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 – n – – n –		3	3	10	cc: 000 NZ non zero, 001 Z zero, 010 NC non carry, 011 C carry, 100 PO parity odd, 101 PE parity even, 110 P sign positive, 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 – e - 2 –	18	2	3	12	
JR C, e	If C = 0 continue If C = 1 PC ← PC + e	•	•	X	•	X	•	•	•	00 111 000 – e - 2 –	38	2 2	2 3	7 12	If condition not met If condition is met
JR NC, e	If C = 1 continue If C = 0 PC ← PC + e	•	•	X	•	X	•	•	•	00 110 000 – e - 2 –	30	2 2	2 3	7 12	If condition not met If condition is met
JP Z, e	If Z = 0 continue If Z = 1 PC ← PC + e	•	•	X	•	X	•	•	•	00 101 000 – e - 2 –	28	2 2	2 3	7 12	If condition not met If condition is met
JR NZ, e	If Z = 1 continue If Z = 0 PC ← PC + e	•	•	X	•	X	•	•	•	00 100 000 – e - 2 –	20	2 2	2 3	7 12	If condition not met If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z	Flags H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 FD 11 101 001 E9	2	2	8	
DJNZ, e	B ← B - 1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 10 – e – 2 –	2	2	8	If B ≠ 0
	If B ≠ 0, PC ← PC + e										2	3	13	If B ≠ 0

NOTES
- e represents the extension in the relative addressing mode
- e is a signed two's complement number in the range < – 126 , 129 >
- e – 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Call and Return Group

Mnemonic	Symbolic Operation	S	Z	Flags H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	(SP - 1) ← PC_H (SP - 2) ← PC_L PC ← nn	•	•	X	•	X	•	•	•	11 001 101 CD – n – – n –	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	•	X	•	•	•	11 cc 100 – n – – n –	3	3	10	If cc is false
											3	5	17	If cc is true
RET	PC_L ← (SP) PC_H ← (SP + 1)	•	•	X	•	X	•	•	•	11 001 001 C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	•	11 cc 000	1	1	5	If cc is false
											1	3	11	If cc is true
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 ED 01 001 101 4D	2	4	14	
RETN[1]	Return from non maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 101 45	2	4	14	
RST p	(SP - 1) ← PC_H (SP - 2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	•	X	•	•	•	11 t 111	1	3	11	

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

NOTE [1]RETN loads IFF_2 → IFF_1

Input and Output Group

Mnemonic	Symbolic Operation	S	Z	Flags H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 DB – n –	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 ED 01 r 000	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	↕ ①	X	X	X	X	↕	•	11 101 101 ED 10 100 010 A2	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 110 010 B2	2	5 (If B ≠ 0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
											2	4 (If B = 0)	16	
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 101 010 AA	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 111 010 BA	2	5 (If B ≠ 0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
											2	4 (If B = 0)	16	
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011 D3 – n –	2	3	11	n to A_0 ~ A_7 Acc to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 ED 01 r 001	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 100 011 A3	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 110 011 B3	2	5 (If B ≠ 0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
											2	4 (If B = 0)	16	
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	↕ ①	X	X	X	X	1	•	11 101 101 ED 10 101 011 AB	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

NOTE ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	?	X	X	X	1	•	10 101 101 ED 10 111 011	2 2	5 (If B ≠ 0) 4 If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

Summary of Flag Operation

Instruction	D_7 S	Z	H	P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	V	0	↕	8-bit add or add with carry
SUB s; SBC A, s; CP s; NEG	↕	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	P	0	0	Logical operations
OR s; XOR s	↕	↕	X	P	0	0	
INC s	↕	↕	X	V	0	•	8-bit increment
DEC s	↕	↕	X	V	1	•	8-bit decrement
ADD DD, ss	•	•	X	X	0	↕	16-bit add
ADC HL, ss	↕	↕	X	V	0	↕	16-bit add with carry
SBC HL, ss	↕	↕	X	V	1	↕	16-bit subtract with carry
RLA; RLCA; RRA; RRCA	•	•	X	•	0	↕	Rotate accumulator
RL m; RLC m; RR m RRC m; SLA m SRA m; SRL m	↕	↕	X	P	0	↕	Rotate and shift operations
RLD; RRD	↕	↕	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	•	1	•	Complement accumulator
SCF	•	•	0	X	0	1	Set carry
CCF	•	•	X	X	0	↕	Complement carry
IN r, (C)	↕	↕	0	P	0	•	Input register indirect
INI; IND; OUTI; OUTD	X	↕	X	X	1	•	Block input and output Z = 0 if B ≠ 0, otherwise Z = 0
INIR; INDR; OTIR; OTDR	X	1	X	X	1	•	
LDI; LDD	X	X	0	↕	0	•	Block transfer instructions P/V = 1 if BC ≠ 0, otherwise P/V = 0
LDIR; LDDR	X	0	0	0	0	•	
CPI; CPIR; CPD; CPDR	X	↕	↕	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	0	0	•	The contents of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	1	X	0	•	The state of bit b of location s is copied into the Z flag

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result, while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle. $\overline{BUSREQ}$ forces the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ to go to a high-impedance state so that other devices can control these lines. $\overline{BUSREQ}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{BUSREQ}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{HALT}$. *Halt State* (output, active Low). $\overline{HALT}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{INT}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{INT}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{IORQ}$. *Input/Output Request* (output, active Low, 3-state). $\overline{IORQ}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{IORQ}$ is also generated concurrently with $\overline{M1}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{M1}$. *Machine Cycle One* (output, active Low). $\overline{M1}$, together with $\overline{MREQ}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{M1}$, together with $\overline{IORQ}$, indicates an interrupt acknowledge cycle.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). $\overline{MREQ}$ indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{NMI}$. *Non-Maskable Interrupt* (input, active Low). $\overline{NMI}$ has a higher priority than $\overline{INT}$. $\overline{NMI}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{RD}$. *Memory Read* (output, active Low, 3-state). $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{RESET}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{RFSH}$. *Refresh* (output, active Low). $\overline{RFSH}$, together with $\overline{MREQ}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{WAIT}$. *Wait* (input, active Low). $\overline{WAIT}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{WAIT}$ periods can prevent the CPU from refreshing dynamic memory properly.

$\overline{WR}$. *Memory Write* (output, active Low, 3-state). $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

APPENDIX H

SPECIAL ADDRESSES

NUMBERS

Addresses can be expressed in different ways. When working in the System Monitor, hexadecimal addresses are used. When working in Floating Point Basic or Integer Basic, decimal addresses are used. Decimal addresses can be expressed in two forms: hexadecimal equivalent or for numbers greater than $7FFF ($32767_{10}$) hexadecimal equivalent minus $10000 ($65536_{10}$). This scheme was brought about by the inability of Integer Basic to use numbers in excess of 32767_{10}. A method had to be devised to allow access to the full range of memory addresses $0000 - $FFFF ($0_{10}$ - 65535_{10}) and do it with numbers in the the range of -32767_{10} to 32767_{10}.

The highest or most significant bit (2^{15}) is used as an normal address bit for hexadecimal addressing and is expressed as a sign bit when used with either of the basics. This is twos complement mathematics.

Numbers in this manual, for the most part, are expressed in hexadecimal and at times in equivalent decimal values. They can be converted to the negative number form by subtracting 65536_{10} from the decimal equivalent.

CONVERSION

The chart in Table H-1 can be used to convert hexadecimal numbers to decimal and decimal numbers to hexadecimal.

Table H-1. Number Conversions

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

Converting Numbers

Numbers between $0 - $FF or 0 - 255 are converted in one step. Numbers between $100 - $FFF or 256 - 4095 require one or two steps. Larger numbers require two or three steps. The following examples show how to use the chart to convert numbers.

Example 1

Let's use the first entry in Table H-2 as an example. To convert $C000 to decimal, first find **C** in the leftmost column. Then locate the **000** across the top of the chart. Locate the point in the chart where the column and row for the values located intersect. The answer is **49152.**

Example 2

To convert the number $A52F to decimal requires a few steps. You need to break the number into three parts: **$A000, $500, and $2F.**

Next you need to find the equivalent value for each part:

$A000 = 40960, $500 = 1280, and $2F = 47.

The last step is to add the three parts together: **40960 + 1280 + 47 = 42287.**

Example 3

To convert 42287 to a hexadecimal value requires a slightly different process. First you need to subtract the largest possible decimal number on the chart that that will leave a positive or zero answer and find the hexadecimal value for that number. **42287 - 40960 = 1327** **40960 = $A000**

Repeat the step for 1327. **1327 - 1280 = 47** **1280 = $500**

Repeat the step for 47. **47 - 47 = 0** **47 = $2F**

Add the hexadecimal numbers together. **$A52F**

When the subtraction equals zero (=0), you have found the last required part of the number for conversion.

SPECIAL ADDRESSES

Table H-2 contains the hexadecimal and decimal equivalent values for the BASIS 108 addresses that perform special functions.

Table H-2. Special Addresses

HEX ADDRESS	DECIMAL ADDRESS	READ	WRITE
$C000	49152	Keyboard Input	SW3 Inverse
$C001	49153		SW3 Flash
$C002	49154		SW2 Off (0)
$C003	49155		SW2 On (1)
$C004	49156		SW1 Off (0)
$C005	49157		SW1 On (1)
$C006	49158		SW0 Off (0)
$C007	49159		SW0 On (1)
$C008	49160	Keyboard Status	Keyboard Interrupt Off
$C009	49161		Keyboard Interrupt On
$C00A	49162		40 Character Display On
$C00B	49163		80 Character Display On
$C00C	49164		Dynamic RAM On
$C00D	49165		Static RAM On
$C00E	49166		$C08x Addresses Active
$C00F	49167		$C08x Addresses Inactive
$C010	49168	Clear Keyboard Strobe	
$C020	49184	Toggle Cassette Output	
$C030	49200	Toggle Speaker Output	
$C04x	49216	Utility Strobe	Utility Strobe
$C050	49232	Set Graphic Mode On	
$C051	49233	Set Text Mode On	
$C052	49234	Set Full Graphics Display	
$C053	49235	Set Mixed Graphics Display	
$C054	49236	Select Page 1 for Display	
$C055	49237	Select Page 2 for Display	
$C056	49238	Select Lo-Res or Me-Res Graphics	
$C057	49239	Select Hi-Res Graphics	
$C058	49240	Clear TTL OUT1 (low)	Clear TTL OUT1 (low)
$C059	49241	Set TTL OUT1 (high)	Set TTL OUT1 (high)
$C05A	49242	Clear TTL OUT2 (low)	Clear TTL OUT2 (low)
$C05B	49243	Set TTL OUT2 (high)	Set TTL OUT2 (high)
$C05C	49244	Clear TTL OUT3 (low)	Clear TTL OUT3 (low)
$C05D	49245	Set TTL OUT3 (high)	Set TTL OUT3 (high)
$C05E	49246	Clear TTL OUT4 (low)	Clear TTL OUT4 (low)
$C05F	49247	Set TTL OUT4 (high)	Set TTL OUT4 (high)
$C060	49248	Cassette Input	Select $0000-$1FFF Bank 0
$C061	49249	TTL IN1	Select $0000-$1FFF Bank 1
$C062	49250	TTL IN2	Select $2000-$3FFF Bank 0
$C063	49251	TTL IN3	Select $2000-$3FFF Bank 1
$C064	49252	ANALOG 1 Timer Output	Select $4000-$5FFF Bank 0
$C065	49253	ANALOG 2 Timer Output	Select $4000-$5FFF Bank 1
$C066	49254	ANALOG 3 Timer Output	Select $6000-$7FFF Bank 0
$C067	49255	ANALOG 4 Timer Output	Select $6000-$7FFF Bank 1

Table H-2. Special Addresses (Continued)

HEX ADDRESS	DECIMAL ADDRESS	READ	WRITE
$C068	49256		Select $8000-$9FFF Bank 0
$C069	49257		Select $8000-$9FFF Bank 1
$C06A	49258		Select $A000-$BFFF Bank 0
$C06B	49259		Select $A000-$BFFF Bank 1
$C06C	49260		Select $D000-$DFFF Bank 0
$C06D	49261		Select $D000-$DFFF Bank 1
$C06E	49262		Select $E000-$FFFF Bank 0
$C06F	49263		Select $E000-$FFFF Bank 1
$C070	49264	Start ANALOG x Timers	Start ANALOG x Timers
$C08x	49280	LC Control Addresses	
$C090	49296		Parallel Printer Output
$C098	49304	Serial Interface Input	Serial Interface Output
$C099	49305	Serial Interface Status	Serial Interface RESET
$C09A	49306	Serial Interface Command	Serial Interface Command
$C09B	49307	Serial Interface Control	Serial Interface Control
$C0Ax	49312	Slot 2 Device Select	Slot 2 Device Select
$C0Bx	49328	Slot 3 Device Select	Slot 3 Device Select
$C0Cx	49344	Slot 4 Device Select	Slot 4 Device Select
$C0Dx	49360	Slot 5 Device Select	Slot 5 Device Select
$C0Ex	49376	Slot 6 Device Select	Slot 6 Device Select
$C0Fx	49392	Slot 7 Device Select	Slot 7 Device Select
$C100	49408		Z80 On/Off
$C1C1	49601	Printer Acknowledge	

APPENDIX I

SPECIFICATIONS

POWER SUPPLY

The Switch-Mode power supply has dual line inputs and regulated outputs of:

+5V at 6A
-5V at 0.5A
+12V at 2.5A
-12V at 0.5A

Connector Pin Assignments

AC Input	Pin 1	LIVE	Mating connector:
	Pin 2	NEUTRAL	Molex Type 09-50-3031
DC Outputs	Pin 1	NC	
	Pin 2	KEY	
	Pin 3,4	+12V	
	Pin 5,6	+5V	
	Pin 7,8,9	COMMON	
	Pin 10	+5V	(-5V supply)
	Pin 11	-5V	
	Pin 12	+12V	(-12V supply)
	Pin 13	-12V	

Electrical Specifications

Tables I-1 and I-2 list the power supply specifications.

Table I-1. Power Supply Input and Output Voltages

PARAMETER		MIN	TYP	MAX	UNIT	NOTES
Input Voltage		90	115	135	Vac	
		180	230	270	Vac	
Input Frequency		47	50/60	400	Hz	
Outputs:	VO1	4.9	5.0	5.1	Vdc	+5V Output
	IO1	1.2	3.0	6.0	A	
	VO2	11.4	12.0	12.6	Vdc	+12V Output
	IO2	0.5	1.25	2.5	A	

Table I-1. Power Supply Input and Output Voltages (Continued)

PARAMETER	MIN	TYP	MAX	UNIT	NOTES
VO3	-11.4	-12.0	-12.6	Vdc	-12V Output
IO3	0	0.25	0.5	A	
VO4	-4.75	-5.0	-5.25	Vdc	-5V Output
IO4	0	0.25	0.5	A	

Table I-2. Power Supply Operating Parameters

PARAMETER	MIN	TYP	MAX	UNIT	NOTES
Efficiency	65			%	At full load 115/230 Vac in
Operating Temperature	0°		50°	°C	Ambient Temperature
Output Power			50	W	Maximum Continuous
Output Ripple			1	%	1 Hz to 10 MHz
Line Regulation		0.1	0.2	%	
Load Regulation:					
VO1		0.2	2.0	%	
VO2, VO3, VO4			5.0	%	
Over Voltage Protection	5.9		6.9	V	+5V Supply
Hold-up Time	16	24		mS	Full load at 115/230 Vac

- - - - - - - - - - - - - - - -

Short Circuit Loads	Indefinite period on all outputs
Open Circuit Loads	Indefinite period on all outputs
EMI Requirements	Meets the conduction limits of VDE 0871 'B' for 230 Vac in and FCC 'B' rules for 115 Vac in.
Safety Requirements	Meets UL 1012 safety standard for power supplies.

APPENDIX J

BIBLIOGRAPHY

Just a list of a few publications to read and enjoy.

1. APPLE II REFERENCE MANUAL (Product number A2L0001A)

   The reference manual for the Apple II. It provides a little insight of what the BASIS 108 is all about.

   Apple Computer Inc.
   10260 Bandley Drive
   Cupertino CA 95014
   (408) 996-1010

2. BASIC PROGRAMMING REFERENCE MANUAL (Product number A2L0006)

   The Applesoft Reference Manual. Contains detail about the language.

   Same address as 1.

3. APPLE II BASIC PROGRAMMING MANUAL (Product number A2L0005X)

   The Integer Basic Reference Manual. Contains detail about the language.

   Same address as 1.

4. THE DOS MANUAL (Product number A2L0036)

   The Disk Operating System 3.3 manual. Contains detail about the operating system.

   Same address as 1.

5. APPLE PASCAL OPERATING SYSTEM REFERENCE MANUAL (Product number A2L0028)

   The Apple Pascal Operating System Manual. Contains detail on the operation of Pascal on the Apple II.

   Same address as 1.

6. APPLE PASCAL LANGUAGE REFERENCE MANUAL (Product number A2L0027)

   The Apple Pascal language reference manual. Provides detail about Apple's use and adaptation of Pascal II.1.1 language.

   Same address as 1.

7. UCSD PASCAL USERS MANUAL VERSION IV.0

   This manual provides detail about Pascal version IV.0.

   SofTech Microsystems, Inc.
   9494 Black Mountain Road
   San Diego CA 92126
   (714) 578-6105

8. 6502 ASSEMBLY LANGUAGE PROGRAMMING by Lance A. Leventhal

   This book provides detail about assembly language programming of the 6502 microprocessor.

   OSBORNE/McGraw-Hill
   630 Bancroft Way
   Berkeley CA 94710

9. PROGRAMMING THE Z-80 by Rodnay Zaks

   This book provides detail about assembly language programming of the Z-80 microprocessor.

   Sybex Inc.
   2344 Sixth Street
   Berkeley CA 94710
   (415) 848-8233 or (800) 227-2346

10. APPLE PASCAL: A HANDS-ON APPROACH by Arthur Luehrmann and Herbert Peckham

    A very good book to use if you'd like to learn about Pascal.

    McGraw-Hill Book Company
    No address listed. (In all major cities)

11. SOFTCARD MANUAL

    Detail about the use of CP/M on the Apple II. Provides detail on many facets of CP/M and the Microsoft Languages.

    Microsoft Consumer Products
    400 108th Avenue Northeast, Suite 200
    Bellevue WA 98004

12. THE CP/M HANDBOOK WITH MP/M by Rodnay Zaks

This book provides detail about the CP/M Operating System.

Same address as 9.

13. ASSEMBLY LANGUAGE DEVELOPMENT SYSTEM MANUAL

This manual provides detail on the use of the ALDS software. Very helpful if you are trying to program the 6502 or the Z-80 as configured on the BASIS 108.

Same address as 11.

There are many more books and manuals that are very helpful. Digital Research, the owner of CP/M, has many publications about CP/M and the utility programs used with CP/M. They are located at:

Digital Research
Post Office Box 579
160 Central Avenue
Pacific Grove CA 93950
(408) 649-3896

Some of the listed publications have a Bibliography. More and more books are being published every day. Magazine articles about the care and feeding of the BASIS 108 are appearing. Read, read, read, and enjoy.

APPENDIX K

SCHEMATICS

SCHEMATICS

This appendix contains the computerboard schematic, keyboard schematic, and computerboard layout.

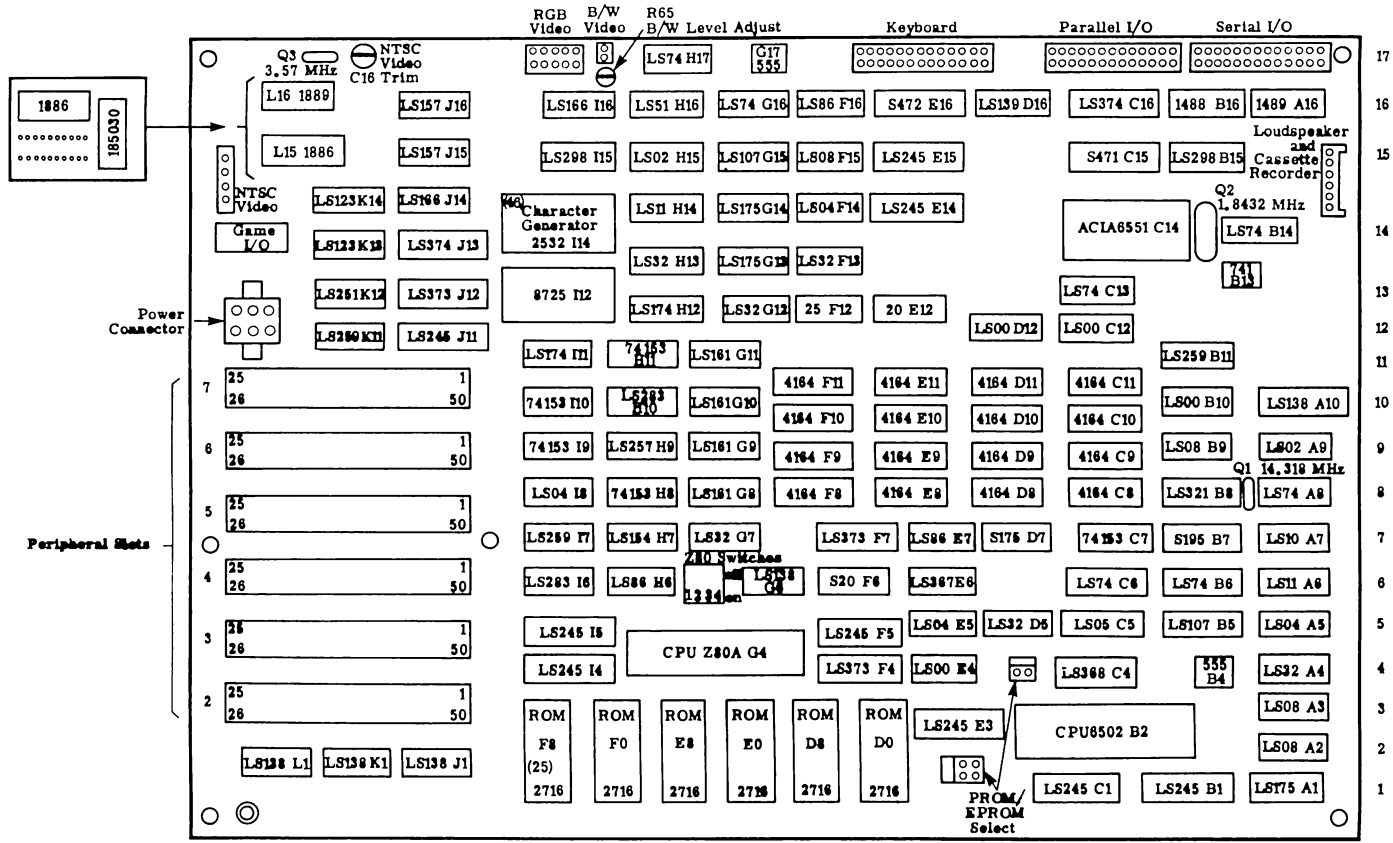


Figure K-1. Computerboard Layout

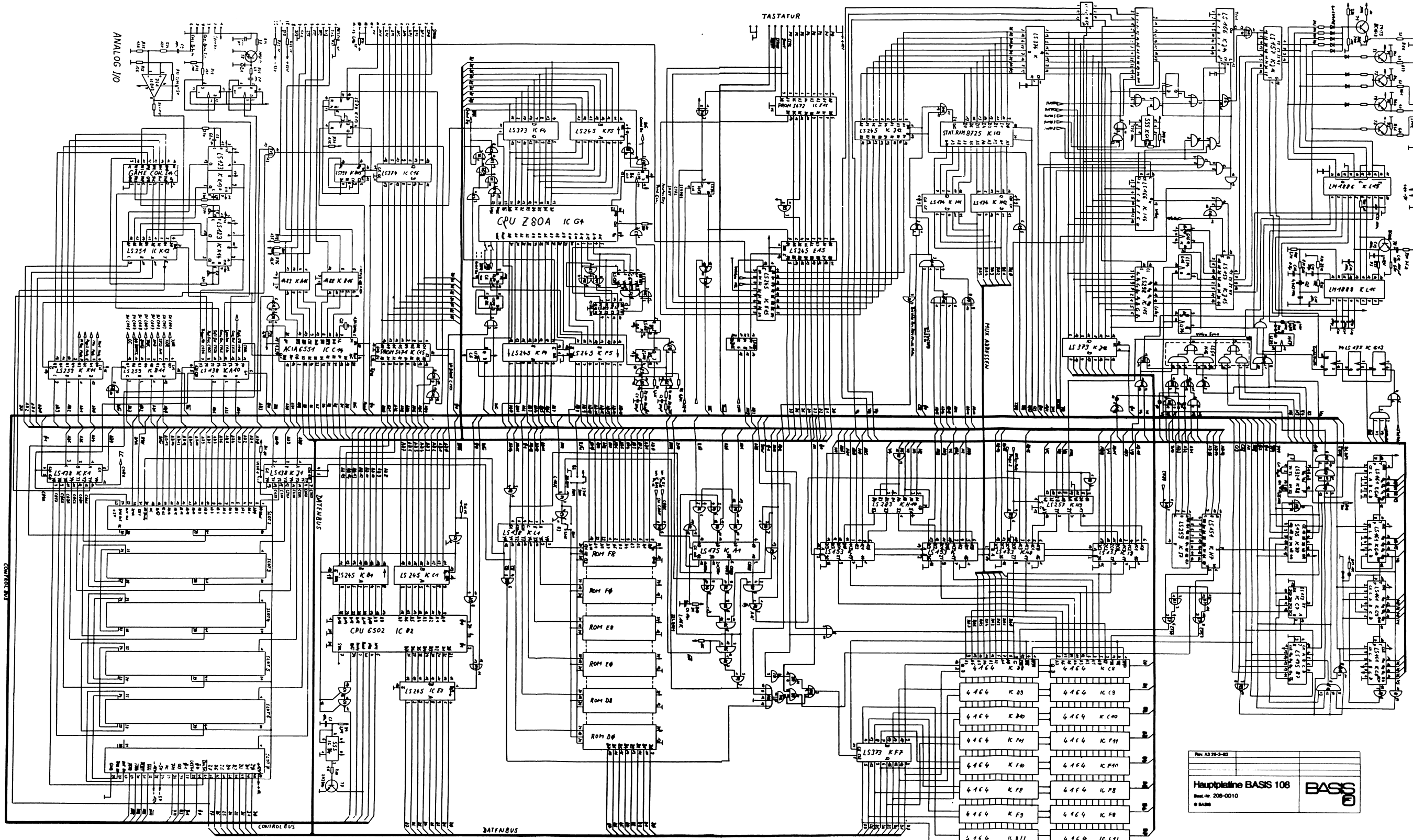


Figure K-2. Computerboard Schematic

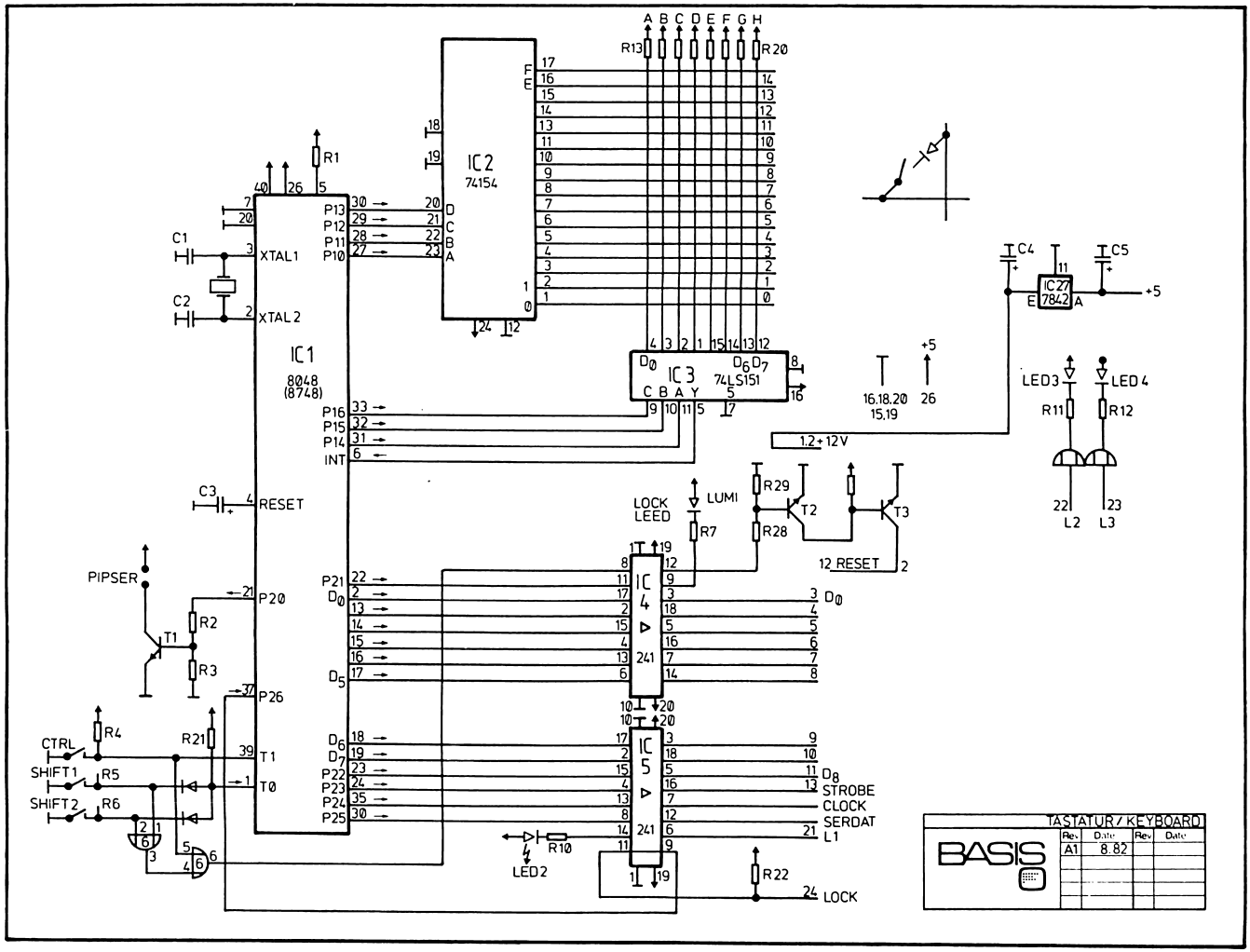


Figure K-3. Keyboard Schematic

APPENDIX L

NOTES AND ADDITIONAL INFORMATION

This space is for all those notes and good things you collect. If you keep them here you'll always be able to find them.

WARRANTY

BASIS 108 LIMITED WARRANTY

BASIS MICROCOMPUTER GMBH warrants this product against defects in material and workmanship for a period of 180 days from the date of purchase. This warranty applies only to products and components manufactured by **BASIS**, which can be identified by the **BASIS** trademark, tradename, or logo affixed to them. **BASIS** does not warrant any products not manufactured by **BASIS.**

During the warranty period **BASIS** will repair (or at its option replace), at no charge, this **BASIS** product or any **BASIS** manufactured components that prove defective, provided the **BASIS** product is returned to an authorized **BASIS** Service Center.

In order to obtain warranty performance, obtain the name and address of the nearest authorized **BASIS** Service Center. Attach to the **BASIS** product your name, address, telephone number, a description of the problem, and proof of date of retail purchase (such as the bill of sale or cancelled check). Return the **BASIS** product to the nearest authorized **BASIS** Service Center, transportation charges prepaid.

This warranty does not apply if the product has been damaged by accident, abuse, misuse or misapplication, or as a result of service or modification by other than an authorized **BASIS** Service Center.

BASIS IS NOT RESPONSIBLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING DAMAGE TO PROPERTY AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURY. THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES. ANY IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO 180 DAYS FROM THE DATE OF RETAIL PURCHASE OF THIS PRODUCT.

MICROCOMPUTER
GMBH

D-4400 Münster
Postfach 1603
Telex 892 643 basis d
BTX 244