SBC-300

SINGLE BOARD COMPUTER

OPERATIONS MANUAL

SD #7140170
REVISION C
AUGUST 25, 1983

SDSYSTEMS

A SYNTECH COMPANY

SBC-300

SINGLE BOARD COMPUTER

OPERATIONS MANUAL

SD #7140170
REVISION C
AUGUST 25, 1983

EMI NOTICE

This equipment has been designed and constructed to professional standards. However, the equipment must be utilized correctly by the user to obtain proper performance and to comply with applicable industry and governmental regulations.

Since the equipment is supplied as an unconfigured component and cannot be tested for electro-magnetic interference (EMI) in all possible configurations, the equipment is not subject to standards imposed by Subpart J of Part 15 of FCC Rules and Regulations.

Final system configuration will require compliance with applicable FCC regulations. SDSystems recommends the installation of shielded data cables for all external cabling. Electro-magnetic interference (EMI) levels are dependent upon final system hardware configuration and application.

Contact SDSystems Customer Service if additional configuration information is required.

TERMINATION NOTE

SDSystems recommends the use of active termination circuitry on the S-100 bus to avoid system problems.

MODIFICATION FOR SBC-300 REVISION A ARTWORK WITH MONITOR ROM 2.0

The Revision A artwork for the SBC-300 is missing pull up resistors on the EPROM for pins 1 and 27. To correctly function with a 2764 EPROM, these pins must be tied to +5V. To accomplish this, wire pins 1 and 27 to pin 28. This will enable the EPROM.

TABLE OF CONTENTS

Subsection Number	Title	Page No.
	SECTION I INTRODUCTION	
1.0	GENERAL	1-1
1.1	FEATURES	1-1
1.2	SCOPE	1-1
	SECTION II FUNCTIONAL DESCRIPTION	
2.0	GENERAL	2-1
2.1	CENTRAL PROCESSING UNIT	2-2
2.1.1	CPU	2-2
2.1.1.1	PROCESSOR OPERATION	2-2
2.1.1.2	Z-80 CPU	2-2
2.1.2	CLOCK RATE GENERATOR	2-3
2.1.3	CPU CLOCK SPEED SELECTION	2-4
2.2	MEMORY	2-5
2.2.1	RAM	2-5
2.2.1.1	REFRESH	2-6
2.2.1.1.1	PIN 1 REFRESH MEMORY DEVICES	2-6
2.2.1.1.2	AVOIDING LONG WAIT STATES	2-6
2.2.1.1.3	REFRESH DURING DMA OPERATIONS	2-7
2.2.1.2	REFRESH WAIT STATE GENERATOR	2-7
2.2.1.2.1	WAIT STATES	2-8
2.2.1.2.2	WAIT STATE SELECTION	2-8
2.2.1.2.3	PIN 1 REFRESH GENERATION	2-9
2.2.1.3	MEMORY TIMING	2-9
2.2.1.3.1	RAM ADDRESS DECODE	2-9
2.2.1.3.2	ROW AND COLUMN TIMING	2-10
2.2.2	ROM	2-10
2.2.3	MEMORY ADDRESS AND I/O PORT DECODE	2-10
2.2.3.1	MEMORY ADDRESS DECODE	2-11
2.2.3.2	I/O PORT DECODE	2-11
2.2.4	PARITY	2-11
2.2.5	MODE DEPENDENT MEMORY CHARACTERISTICS	2-11
2.2.5.1	BUS MASTER MODE MEMORY	2-11
2.2.5.2	SLAVE PROCESSOR MODE MEMORY	2-12
2.3	MEMORY MAPPER	2-12
2.3.1	MAPPER OPERATION	2-12
2.3.1.1	BOARD ADDRESS SELECT	2-13
2.3.1.2	MAPPER RAM	2-13
2.4	INTERRUPT CONTROLLER	2-14
2.4.1	INTERRUPT SELECT	2-14
2.5	BUS INTERFACE	2-15
2.5.1	BUS TIMING CONTROLLER	2-15
2.5.2	POWER ON CLEAR AND RESET	2-16
2.5.3	SLAVE CONTROL	2-16
2.6	MASTER/SLAVE CONTROL	2-18
2.6.1	MASTER/SLAVE SELECT	2-18

TABLE OF CONTENTS (Continued)

Subsection Number	Title	Page No.
	SECTION II--Continued FUNCTIONAL DESCRIPTION--Continued	
2.6.2	LINE TERMINATION	2-18
2.7	SASI/PARALLEL PORT	2-19
2.7.1	CIO	2-19
2.8	SERIAL I/O	2-19
2.8.1	ASCC	2-20
2.9	POWER REGULATION	2-20
2.9.1	5 VOLT BUS OPERATION	2-20
	SECTION III THEORY OF OPERATION	
3.0	GENERAL	3-1
3.1	MASTER/SLAVE MODES	3-1
3.1.1	MASTERS	3-1
3.1.1.1	PERMANENT BUS MASTER	3-1
3.1.1.2	TEMPORARY BUS MASTER	3-1
3.1.2	SLAVES	3-2
3.1.2.1	SLAVE PROCESSORS	3-2
3.1.2.2	SLAVE PROCESSORS AND MULTI-USER SYSTEMS	3-2
3.1.3	BUS CYCLE OPERATIONS	3-3
3.1.3.1	BUS CYCLES	3-4
3.1.3.2	TEMPORARY MASTER ACCESS CYCLES	3-4
3.2	SLAVE PRIORITY	3-5
3.3	INTERRUPTS	3-5
3.3.1	NON-MASKABLE INTERRUPTS	3-5
3.3.2	MASKABLE INTERRUPTS	3-5
3.3.3	INTERRUPT ENABLE	3-6
3.3.4	INTERRUPT CONTROL	3-6
3.3.4.1	OFF-BOARD INTERRUPT PRIORITIZATION	3-7
3.3.4.2	ON-BOARD INTERRUPT PRIORITIZATION	3-7
3.3.4.3	ON-BOARD INTERRUPT ARBITRATION	3-8
3.3.5	INTERRUPT ACKNOWLEDGE	3-9
3.4	MAPPER CONCEPTS	3-10
	SECTION IV OPERATIONS	
4.0	GENERAL	4-1
4.1	INSTALLATION	4-1
4.1.1	IEEE-696	4-1
4.1.2	SDSYSTEMS S-100 AND OTHERS	4-1
4.1.3	INITIAL CHECKOUT	4-1
4.2	SERIAL I/O	4-2

TABLE OF CONTENTS (Continued)

Subsection Number	Title	Page No.
	SECTION V MONITOR	
5.0	GENERAL	5-1
5.1	SD MONITOR START-UP	5-1
5.1.1	COMMAND SYNTAX	5-1
5.2	MEMORY COMMANDS	5-3
5.2.1	MEMORY MANAGEMENT - "A" COMMAND	5-3
5.2.2	MEMORY DISPLAY - "D" COMMAND	5-4
5.2.3	EXAMINE MEMORY - "E" COMMAND	5-5
5.2.4	FILL MEMORY - "F" COMMAND	5-5
5.2.5	EXTENDED MEMORY TEST - "J" COMMAND	5-6
5.2.6	LOCATE STRING - "L" COMMAND	5-6
5.2.7	MOVE MEMORY - "M" COMMAND	5-7
5.2.8	MEMORY TEST - "T" COMMAND	5-7
5.2.9	VERIFY MEMORY - "V" COMMAND	5-7
5.3	INPUT/OUTPUT COMMANDS	5-8
5.3.1	INPUT FROM PORT - "I" COMMAND	5-8
5.3.2	OUTPUT FROM PORT - "O" COMMAND	5-8
5.3.3	PORT EXAMINE - "P" COMMAND	5-9
5.4	PROGRAM CONTROL COMMANDS	5-9
5.4.1	BREAKPOINT - "B" COMMAND	5-10
5.4.2	"GO" TO PROGRAM - "G" COMMAND	5-11
5.4.3	SINGLE STEP - "S" COMMAND	5-11
5.4.4	REGISTER EXAMINE - "X" COMMAND	5-12
5.4.5	HEX ARITHMETIC - "H" COMMAND	5-12
5.4.6	COMMAND PROCESSING - "Y" COMMAND	5-13
5.5	DISK UTILITY COMMANDS	5-13
5.5.1	BOOT UP - "C" COMMAND	5-13
5.5.2	READ FROM DISK - "R" COMMAND	5-14
5.5.3	WRITE TO DISK - "W" COMMAND	5-15
5.5.4	FORMAT A DISK - "Z" COMMAND	5-15
5.6	CIO INITIALIZATION AND USE	5-15
5.6.1	ACCESSING THE CIO INTERNAL REGISTERS	5-15
5.6.2	SBC-300 MONITOR AND CIO INITIALIZATION	5-16
5.6.2.1	PORT B	5-16
5.6.2.2	PORT C	5-18
5.6.2.3	PORT A	5-19
5.6.2.4	COUNTER/TIMERS	5-19
5.7	ASCC INITIALIZATION	5-19
5.8	PARITY ERRORS	5-19
	SECTION VI ENVIRONMENTAL CONSIDERATIONS	
6.0	GENERAL	6-1
6.1	POWER REQUIREMENTS	6-1
6.2	PHYSICAL SPECIFICATIONS	6-1

TABLE OF CONTENTS (Continued)

Subsection Number	Title	Page No.
APPENDICES		
A	SELECTED IEEE-696 SPECIFICATION SHEETS	A-1
B	HEXADECIMAL TO DECIMAL CONVERSION	B-1
C	STANDARD ASCII CODES	C-1
D	EBCDIC TABLE	D-1
E	CIO SPECIFICATION	E-1
F	ASCC SPECIFICATION	F-1
G	PAL16L8 SPECIFICATION	G-1
H	PORT ASSIGNMENTS	H-1
I	PARALLEL INTERFACES: SASI/CENTRONICS	I-1
J	MONITOR LOCATIONS	J-1
K	SAMPLE I/O SUPPORT CODE	K-1
L	JUMPER CONNECTION QUICK REFERENCE TABLES	L-1
M	DISCLAIMER	M-1
N	LIMITED WARRANTY	N-1
O	PARTS LIST FOR SBC-300	O-1
P	PARTS PLACEMENT DIAGRAM	P-1
Q	SCHEMATIC	Q-1

ILLUSTRATIONS

Figure No.	Title	Page No.
2-1	BLOCK DIAGRAM	2-1
4-1	ADDRESS SWITCH	4-1
4-2	SIGN-ON MESSAGE	4-2
4-3	MEMORY ERROR MESSAGE	4-2

TABLES

Table No.	Title	Page No.
2-1	CPU CLOCK SPEED SELECTION	2-4
2-2	WAIT STATE SELECTION	2-9
2-3	ON-BOARD MEMORY LOCATION IN BUS SYSTEM SELECTION	2-12
2-4	JUMPER CONNECTIONS FOR BUS MASTER OPERATION	2-19
5-1	SD MONITOR COMMAND SUMMARY	5-2
5-2	SD MONITOR REGISTER MAP	5-10
5-3	DISK FORMATS	5-14

SECTION I
INTRODUCTION

1.0 GENERAL

The SBC-300 is a high performance, self-contained microcomputer system designed around the Z-80 family of microprocessors. Control (CPU), memory, and extensive input/output facilities are all contained on a single printed wiring board conforming both physically and electrically to every S-100 BUS requirement as specified by IEEE-696. Operation as either the IEEE-696 Permanent Bus Master or as one of any number of slave processors on the bus is supported. The advanced design of the system allows a wide variety of function and performance options to be selected and easily implemented by the end user in most cases by simple dip switch or jumper arrangement. In addition, the SBC-300 is available with the appropriate CPU, support, memory elements, and jumpers installed for operation at clock speeds of 4 or 6 mHz.

In most cases, options are selected by the connection or disconnection of one or more jumpers; a table will show the specific jumper connections to be made for each option desired. The SBC-300 will be configured at the factory in a manner which is likely to suit the greatest number of users. Those requiring different performance may change option selections as required.

1.1 FEATURES

- S-100 Bus Compatible (IEEE-696)
- Z-80 CPU: 4 or 6 mHz
- 64K Bytes of Internal RAM Plus Parity
- 2 to 16K Bytes of PROM Using Byte-Wide PROM
- System Memory Addressing to 16Mbytes
- Serial, Parallel, and Counter-Timer I/O
- SASI Port
- Fully Programmable Communications Options
- Dual Programmable Serial Full-Duplex Channels
- Dual Programmable Baud Rate Generators

1.2 SCOPE

This document describes the function, performance, electrical and physical characteristics of the SBC-300 single board microcomputer system and its interface to the IEEE-696 bus. For detailed description and analysis of individual integrated circuit components, the SASI port, and the IEEE-696 bus beyond what is presented here, consult the various manufacturer data books and standard specifications.

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION II
FUNCTIONAL DESCRIPTION

2.0 GENERAL

This section provides a functional description of the basic blocks that make up the SBC-300 single board microcomputer. A block diagram of the SBC-300 appears in Figure 2-1, and each block is discussed in this section. Refer to Section III for an in-depth theory of operation.

The SBC-300 communicates with peripherals through one SASI/parallel and two synchronous or asynchronous serial ports. When operated as the Permanent Bus Master in accordance with the IEEE-696 standard, the SBC-300 controls all aspects of bus operation and provides timing, control, and status signals for S-100 bus operation as well as its own operation.

In the slave processor mode, the SBC-300 responds to bus cycles initiated by the current bus master and/or processes data in accordance with instructions stored in the on-board ROM and RAM. It is able to communicate with the outside world via its serial and parallel ports. A slave appears to the bus as a 64K block of RAM. Data is transferred between the slave and the bus via the slave's on-board dual ported RAM. A slave device may request service from the bus master by generating an interrupt request.

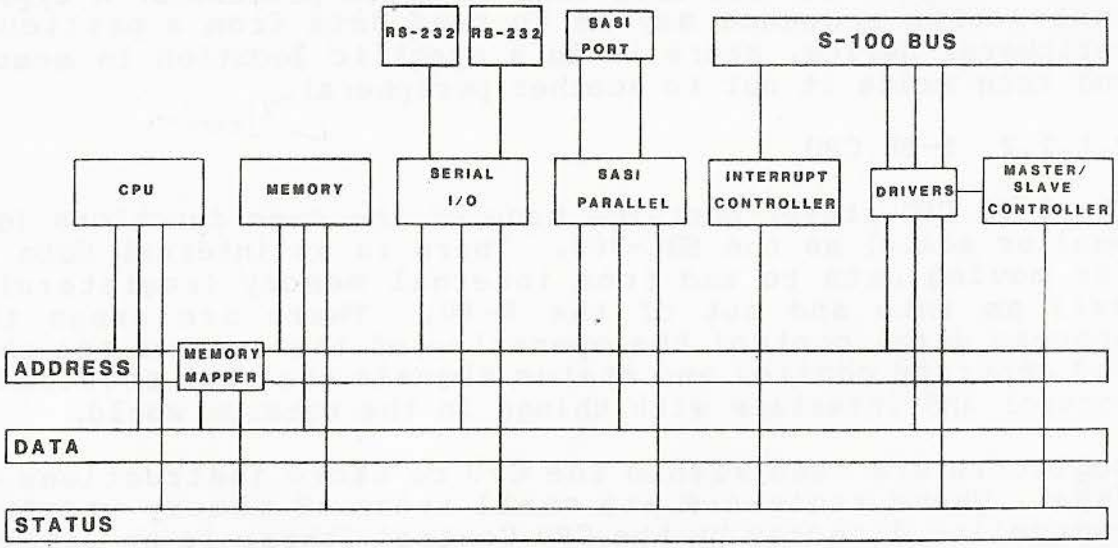


Figure 2-1. BLOCK DIAGRAM

2.1 CENTRAL PROCESSING UNIT

This block of the SBC-300 includes the CPU and the clock rate generator. The CPU of the SBC-300 is a member of the Z-80 family and can be run at clock speeds of 4 or 6 mHz as selected by the appropriate configuration of the clock speed jumpers.

2.1.1 CPU

During normal operation, the CPU directs all aspects of data manipulation, movement, storage, and retrieval within the SBC-300 microcomputer. The CPU controls communications with the outside world (serial and parallel I/O), other bus peripherals, and dynamic memory refresh during normal bus cycles.

2.1.1.1 Processor Operation

The function of the CPU is to get instructions from memory and perform the desired operations.

The CPU accesses a location in memory and retrieves a coded instruction. This instruction is decoded and then performed by the CPU. The address of the next instruction to be executed is applied to the address lines and the instruction (OP CODE) is fetched and executed. This process is continued indefinitely or until the processor is stopped for some reason.

The system memory is used to contain the instructions for the CPU to execute and the data that is to be processed. A typical instruction sequence may be to read data from a particular peripheral device, store it in a specific location in memory, and then write it out to another peripheral.

2.1.1.2 Z-80 CPU

The Z-80 CPU itself contains many of the same functions (on a smaller scale) as the SBC-300. There is an internal data bus for moving data to and from internal memory (registers) as well as into and out of the Z-80. There are areas that process data, control the operation of the rest of the chip, and generate control and status signals that are required to control and interface with things in the outside world.

Registers are used within the CPU to store instructions and data. These registers are small areas of memory which are controlled directly by the CPU Control (there is no internal address bus for accessing registers). The Z-80 contains 208 bits of R/W memory that are accessible to the programmer. This memory is arranged as eighteen 8 bit registers and four 16 bit registers. All Z-80 registers are constructed of static RAM and include 2 sets of 6 general purpose registers. These registers may be used individually as six 8 bit registers or in pairs as three 16 bit registers. There are also 2 sets of accumulator and flag registers and 6 special purpose registers.

The actual processing in the Z-80 takes place in the Arithmetic and Logic Unit (ALU) which communicates with the internal data registers as well as the external data bus by way of the internal data bus. The ALU can perform a number of logical and arithmetic operations on data contained in internal registers at the direction of the CPU Control. These operations include:

ADD	XOR	INCREMENT
SUBTRACT	COMPARE	DECREMENT
AND	SHIFT L/R	SET BIT
OR	TEST BIT	RESET BIT

One instruction at a time is retrieved from the system memory and loaded in the instruction register. The CPU Control section decodes the instruction and generates all control signals necessary to execute the instruction. These signals read and write data from and to the registers, control the ALU, and control all external items as required by the rest of the system.

Dynamic RAM must be refreshed by accessing each row in the memory matrix once every 2 or 4 milliseconds, depending on the RAM type. While the CPU is busy decoding an instruction that has been fetched, the internal memory refresh register sends address data out on the lower portion of the address bus along with a refresh control signal to access a row of memory. The register is incremented one and the new data sent out to access the next row of memory after every instruction fetch. As the register rolls over, the first row is again accessed. As long as the CPU is running (fetching and performing instructions), the memory is automatically refreshed by the memory refresh counter. This operation is completely transparent to the system.

2.1.2 Clock Rate Generator

The clock rate generator includes the crystal controlled clock oscillator, the division logic to derive the CPU clock (Ø) and the 2 mHz CLOCK signal for the bus from the crystal frequency, and the line driver for Ø. The divider supplies either a 6 mHz or a 4 mHz CPU clock and a 2 mHz clock. The 6 mHz clock is derived from a 12 mHz crystal, and the 4 mHz clock is derived from an 8 mHz crystal. These rates are selected by a combination of the appropriate chip installed in the board and the installation of a jumper. A 74LS93 must be installed for 4 mHz operation or a 74LS92 for 6 mHz operation.

This IC also applies the proper division ratio to the crystal frequency to arrive at an output of 2 mHz for either of the two possible crystal frequencies that might be installed. The bus line CLOCK must always carry a signal of 2 mHz regardless of the processor clock speed to comply with the IEEE-696 specification.

2.1.3 CPU Clock Speed Selection

The SBC-300 is configured at the factory for the processor speed specified, but may be reconfigured by the user for other CPU clock speeds. If the clock speed is to be increased, the speed of the CPU, peripherals, and memory chips must be increased as well. This requires the removal of the existing parts and installation of faster parts in these locations. If the clock speed is to be reduced for some reason, no change of these units is required.

*** * * C A U T I O N * * ***

MULTI-LAYER PRINTED WIRING BOARD

When removing parts from the SBC-300 board, ensure that the proper desoldering tools and techniques are used to prevent damage to the printed wiring board surface and inter-layer etches.

The actual CPU clock speed is determined by a combination of jumper connections and parts (crystal frequency and division ratio). One or both of these parts must be changed to effect a change of the CPU clock speed. Once these parts are in place and the appropriate speed units are installed for ICs U46 through U48 and the on-board RAM (Table 2-1), the SBC-300 is configured for one of two CPU clock speeds.

Table 2-1. CPU CLOCK SPEED SELECTION

Proc. Speed (mHz)	Xtal (Y1) (mHz)	U42 74LS-	U46 Peripherals	U47	U48 CPU	RAM Speed (ns)	Delay Line (ns)
6	12	'92	Z8536A	Z8531A	Z-80B	150	150
4	8	'93	Z8536	Z8531	Z-80A	200	200

If the fastest (most expensive) CPU, peripheral, and memory chips are installed, it is recommended that the CPU clock be run as fast as is consistent with the slowest chip on board. If the fast units are installed, there is little to be gained (slightly less power will be dissipated) by running the CPU clock at a lower than maximum speed. If the SBC-300 is to be run at a low CPU clock rate, populate the board with the slowest parts.

2.2 MEMORY

The SBC-300 supports full 24 bit memory addressing in accordance with the IEEE-696 specification and may be operated in either the PERMANENT BUS MASTER or the SLAVE mode. Appropriate memory facilities are provided on board to fully exploit the flexibility and utility of the standard bus.

A full 64K of dynamic RAM (64Kx1 parts) with parity is installed at the factory. Any byte-wide (up to 128K) PROM may be plugged into the 28 pin socket on the board. Upgrading the on-board memory system requires removal of the existing parts and installation of the replacement memory chips. Several modes of refresh are available to cope with any anticipated situation that might arise on the bus.

The on-board memory, as well as any memory that might be visible to the bus, can be mapped to reside in any location within the 16Mbyte range specified by the IEEE-696 S-100 standard. The 16 bit address capability of the Z-80 processor is augmented by a memory mapper which affords the Z-80 access to the entire 16Mbyte potential of the new S-100 bus standard.

The basic memory block of the SBC-300 can be divided into several subsystems: RAM, ROM, MEMORY ADDRESS AND I/O PORT DECODE, and PARITY. These subsystems are required to support the 2 internal memory systems (RAM and ROM) and to ensure compatibility with the bus standard and the various other blocks on the SBC-300.

2.2.1 RAM

The RAM devices that are installed prior to shipment allow pin 1 refresh and are selected so that their speed is consistent with the CPU clock rate.

Upgrading the on-board RAM requires replacement of the set of memory IC chips and the possible connection of a few jumpers to select the proper wait states. The only constraints are that the RAM speed be compatible with the CPU clock rate and that the refresh requirements are met (acceptable device numbers for replacement appear in the Parts List Appendix). CPU clock rate considerations are discussed above in the CPU subsection and the refresh restrictions are covered below.

The RAM subsystem includes a latch for data out and line drivers for interface to the bus as well as several small blocks and functions which work together to allow the RAM to perform properly and interface with the rest of the system. These items are: refresh, refresh wait state generator, and memory timing.

2.2.1.1 Refresh

Any memory device equivalent or similar to the items noted in the Parts List Appendix may be installed in place of the existing RAM devices to increase capacity, replace failed parts (all chips in the array should be of the same type), or satisfy any other particular need of the user. There is, however, one refresh restriction which must be observed:

* * * C A U T I O N * * *

REFRESH IS NOT SUPPORTED DURING LONG BUS WAIT STATES IF NON-PIN 1 REFRESH PARTS ARE INSTALLED

There are 2 easy solutions to the situation. One is to use pin 1 refresh memory devices in the on-board memory system, and the other is to make sure that items which might cause a long wait state for the processor are never attached to the bus.

2.2.1.1.1 Pin 1 Refresh Memory Devices

Industry standard 64K dynamic RAM devices are refreshed by addressing each row of the storage matrix within 2 milliseconds. This is called Row Address Strobe or RAS refresh. Pin 1 is not connected to the IC die (it is a non-connect) and the only method of refresh is through the refresh address access cycle generated elsewhere.

The Z-80 is rather unique among microprocessors in that it has an internal 7 bit address counter which generates memory refresh cycle addresses after every OP CODE fetch. If the processor is stopped (in a wait state) for an extended period of time, there will not be an OP CODE fetch in time to refresh the memory. If this happens, the data in the memory devices becomes uncertain and cannot be used.

Memory devices with pin 1 refresh, however, allow refresh to be performed by an additional procedure. There is logic on the SBC-300 that detects long wait states. This logic then generates a refresh cycle which is applied to pin 1 of the appropriately equipped memory devices. The pin 1 refresh cycle is repeated at the proper interval until the processor is running again and pin 1 refresh is no longer needed to maintain the memory. The only time this cycle is implemented is during wait states where the CPU is stopped for excessive amounts of time.

2.2.1.1.2 Avoiding Long Wait States

The other method of preventing loss of memory data during long wait states is to never install any device on the bus that might generate a long wait state. Sources of long wait states include items such as polled I/O disk controllers. These devices use wait states to transfer data into the system memory.

If, for example, the processor directs the disk controller to read a sector into memory and the sector requested has just passed the heads, the controller must then wait (which means that the CPU must wait) until that sector comes around again to retrieve the data. This can take as long as 150 ms, which is 148 ms too long if the memory content is to remain valid. The Versafloppy II disk controller is an example of a polled I/O device that uses a wait state to transfer data to and from the memory. DMA devices such as the Versafloppy III disk controller do not cause long wait states for data transfer and so are safe to use without pin 1 refresh memory devices.

2.2.1.1.3 Refresh During DMA Operations

During a DMA operation, where the SBC-300 is the permanent bus master but has relinquished control of the bus to a DMA device, the SBC-300 generates standard memory refresh cycles from bus activity. As a result, even if there is a DMA controller transferring data into another memory board on the bus, SBC-300 memory is preserved due to its self-generation of refresh cycles.

Depending on the design of other memory boards that might be installed on the bus, this self-generated refresh cycle might take care of memory on the bus as well as on the SBC-300 board. Older designs (such as the EXPANDORAM II and III) can be refreshed in this manner, while newer designs (EXPANDORAM IV) support this procedure as well as several others during DMA transfers.

These boards, which might be used with other processors (no other processors generate memory refresh cycles), take care of the refresh for on-board memory devices themselves and do not care if the bus master processor is stopped or not. With that type of design, when refresh is not obtained from the bus master processor, the memory board generates cycles for itself. The bus is held off by the memory board until it is finished so it will not be busy with a refresh cycle at the moment its services are required by the bus. Refresh occurs independently of all other activity on the bus, and memory is thus immune to long wait states. There is usually a jumper selected option on these boards, however, which allows refresh to be forced at certain times.

2.2.1.2 Refresh Wait State Generator

The refresh wait state generator performs 2 functions. It simultaneously generates the pin 1 refresh signal and holds the wait line low. The wait line is held low only during the refresh cycle; so if the long wait state which activated the pin 1 refresh cycle goes away, the pin 1 refresh cycle will not be interrupted until the refresh is complete.

2.2.1.2.1 Wait States

Wait states are ordered by devices attached to the S-100 bus and by the SBC-300 itself. Generally wait states are called to hold up the processor because the CPU runs faster than some peripheral devices are capable of responding. To ensure that valid data read from memory has settled on the data lines before the processor tries to use it, or to prevent the processor from removing the address and data from those lines before a write cycle is complete, the processor is stopped for a period of time until it is safe to proceed.

The wait state is implemented by pulling down the WAIT* line to the processor. The CPU checks this line on the falling edge of the clock during clock cycle T2 and, if it finds that the WAIT* line is active, it suspends all processing that might have occurred during the next clock cycle. This creates a wait state. The CPU will continue to create wait states as long as its WAIT* line is held low.

This line (WAIT*) is controlled by the wait state generator. It can be triggered by the wait state generator (jumper W3; see Appendix called Jumper Connection Quick Reference Tables) when lines indicating an OP CODE or memory operation become active. This memory operation trigger produces a synchronous wait state of exactly 1 clock period. A device on the bus can trigger a wait state of any duration beginning at the next falling edge of the clock by pulling and holding down either of 2 lines on the bus (RDY and XRDY). And finally, the refresh wait state generator can cause a wait state to hold off the processor while it applies a refresh cycle to pin 1 of the on-board RAM. The RAM cannot be written to or read from during refresh, so the processor and bus must be prevented from trying to use the RAM until refresh is complete.

2.2.1.2.2 Wait State Selection

There are 3 modes of memory operation wait state generation possible as selected by jumper W3 (Table 2-2). If the M1 mode is selected, a wait state is generated each time the processor fetches an OP CODE from memory. A wait state will not be generated during a normal memory access because the memory is fast enough to respond to the processor between clock T states. During OP CODE fetches, however, the CPU must refresh the on-board RAM after the read, so a wait state is generated to allow this refresh to be completed before things continue.

If the MREQ* option is selected, a wait state is generated during both OP CODE fetches and normal memory accesses. This allows slower memory devices to be used with a faster processor; the processor is held up during a memory read from the slow memory but operates at full speed otherwise.

If no memory option wait state is selected, then no wait states are generated during memory operations.

Table 2-2. WAIT STATE SELECTION

Wait Selection	Wait State Jumper	Function
M1*	W3-2 to W3-3	1 wait state during OP CODE fetches
MREQ*	W3-1 to W3-2	1 wait state during memory accesses
None	None	No waits inserted

2.2.1.2.3 Pin 1 Refresh Generation

During long wait states, the CPU is stopped. This means that there will be no RAS refresh cycles from the processor to maintain the memory. If the processor is stopped for more than approximately 2 msec, the content of the memory will no longer be valid. To prevent such loss of data, the refresh wait state generator provides pin 1 refresh cycles to the RAM whenever the CPU is in a wait state. This causes a wait state to be generated in order to prevent the CPU from accessing the RAM during refresh.

2.2.1.3 Memory Timing

Memory timing is basically concerned with restructuring the address information that appears on the internal address bus into a form that is compatible with the memory devices and then applying that data to the RAM chips. These 2 functions are implemented by the RAM address decode and the row and column timing areas.

2.2.1.3.1 RAM Address Decode

The RAM address decode is very simple. It serves to interface the RAM chip address lines to the SBC-300 internal 16 bit address bus. This is required because in order to fit into a

standard 16 pin package, large capacity memory devices have only 8 address lines.

The 16 bit address on the bus is broken into two 8 bit bytes and applied first one and then the other to the 8 address lines of the memory chip array. The two halves of the address are referred to as the row and the column address. Each pair of row and column addresses defines a unique storage location within the storage matrix of each memory device. This is the same addressing scheme that was developed for magnetic core memories.

First the row address portion of the information on the address bus is applied to the memory array address lines and then the row enable line (RAS strobe) to the memory device array is activated. Next the column address portion of the information on the address bus lines is applied to the same memory array address lines and then the column enable line (CAS strobe) is activated. This procedure loads the 16 bit address of the desired location into the memory device where it is internally decoded to specify a unique location in the storage array.

There are specific intervals of time which must elapse between each segment of the addressing operation (enables, settling time, change between row and column address, etc.). The row and column timing area is responsible for generation of these intervals.

2.2.1.3.2 Row And Column Timing

The memory devices require a time interval between the different parts of the addressing operation. A delay line with multiple taps provides these intervals. A pulse applied to the input appears at each successive tap a specific time after it appears at the previous tap (or the input, if the first tap is being considered). Devices and lines connected to these taps are therefore activated sequentially with specific time intervals inbetween.

This area also contains logic which, when activated by the signals from the delay line, controls all aspects of RAM reads and writes.

2.2.2 ROM

A 28 pin socket on the SBC-300 board allows any of several byte-wide (2Kx8, 4Kx8, etc.) ROMs to be plugged in. The Parts List Appendix shows compatible device type numbers. The ROM installed before shipment includes the Monitor firmware and the system BIOS. There is empty space in the ROM that can be used for any purpose the user desires. However, the user is responsible for programming the ROM with additional code.

The ROM portion of the on-board memory occupies the first 64K segment of the total 128K block even though a relatively small device might be installed in the socket. Because the ROM is invisible to the bus, the SBC-300 appears to other items on the bus as an ordinary 64K block of RAM when in the slave mode.

2.2.3 Memory Address And I/O Port Decode

The on-board address and port decoding is handled by a combination of the Zilog 8536 Counter Input/Output chip (CIO), a PAL[1] (U28) and a decoder chip. The CIO generates a signal (BOOT) which is used by the PAL in conjunction with other timing signals to decide whether the processor is talking to ROM, RAM or off-board memory. When an I/O address is being generated, the PAL inhibits the enables to memory devices, and the port decoder (U57) enables the particular I/0 device being addressed.

2.2.3.1 Memory Address Decode

The memory decoder generates 4 signals which are used to determine what the SBC-300 is addressing. These 4 signals are created within a PAL (U28) from a combination of standard Z-80 timing signals and on-board hardware/software switches.

The 4 signals are PROM ENB*, RAM ENB*, RDOFF* and GORAM*. PROM ENB* is the on-board ROM enable signal. It is created when the BOOT signal is active. RAM ENB* is the on-board RAM enable signal for the ROM data latch. RDOFF* is the signal which determines that the present memory cycle is off board on another 696 board. The 4th signal is GORAM* and is used to generate the RAS* and CAS* RAM timing signals.

2.2.3.2 I/O Port Decode

The I/O port decoder is an address decoder which enables a particular I/O device whenever its preselected address is found on the address bus. The I/O decoder generates the enable signals for the ASCC, CIO, memory mapper, SASI Port and the ATTEN interrupt bit.

2.2.4 Parity

The SBC-300 generates and checks odd parity in its on-board RAM. The parity checker generates 2 parity error signals for use by the on-board processor and, if in the slave mode, the bus master. Parity error detection is enabled by jumper W14-4. The on-board parity error signal is also sent to CIO port C bit 1.

2.2.5 Mode Dependent Memory Characteristics

Some memory characteristics exhibited by the SBC-300 change according to the SBC-300 mode selected. The SBC-300 may be operated as either the Permanent Bus Master (PBM) or as a slave processor in accordance with IEEE-696 specifications.

2.2.5.1 Bus Master Mode Memory

In the PBM mode, the SBC-300 occupies 128K of the system memory space. The 1st 64K segment is PROM (even though there may be only an 8K PROM installed in the socket), and the 2nd

[1]PAL - Programmable Array Logic MMI$_R$

64K segment is the RAM. The entire on-board memory can be mapped into any 128K location in the 16Mbyte bus memory system desired by the appropriate setting of dip switch SW-1 elements as detailed in Table 2-3.

Table 2-3. ON-BOARD MEMORY LOCATION IN BUS SYSTEM SELECTION

SW-1 (X=Off; O=On)							Always Off When Master	SBC-300 On-Board Memory Begins At (In Hex)--
8	7	6	5	4	3	2	1	
O	O	O	O	O	O	O	X	010000
O	O	O	O	O	O	X	X	030000
O	O	O	O	O	X	O	X	050000
O	O	O	O	O	X	X	X	070000
			|					|
			|					|
X	X	X	X	X	X	O	X	FD0000
X	X	X	X	X	X	X	X	FF0000

2.2.5.2 Slave Processor Mode Memory

In the slave mode, only the 64K of on-board RAM is accessable by the bus. The ROM is completely invisible to the bus, even though the on-board processor might be operating out of it. The RAM portion of the on-board memory is dual ported--it can be accessed by the on-board processor and the bus as well. In the slave mode, the on-board RAM is the only means through which the PBM and all other items on the bus can communicate with the slave processor.

2.3 MEMORY MAPPER

Since the Z-80 microprocessor contains 16 address lines, it can address 64K of memory directly. In order to meet the IEEE-696 specification and allow the addressing of the full 16Mbyte of memory, an additional 8 address lines must be generated. A memory mapper allows the Z-80 to control indirectly, through the mapper, all 24 address lines of the S-100 bus. The memory mapper consists of bipolar RAM, a multiplexer, gating logic, and a switch and comparator for selecting which 64K block of system memory the board occupies.

2.3.1 Mapper Operation

The upper 4 bits of the Z-80 physical address lines (A12-A15) are applied to the address inputs of the memory mapper RAM to determine which location within the mapper will be accessed and output to the extended address bus (A12-A23). The mapper is accessed as I/O port 71H for purposes of updating the stored address. The memory mapper is bypassed during I/O operations and during accesses to the on-board EPROM. During these periods the upper 4 address bits are passed directly to the logical address bus.

2.3.1.1 Board Address Select

The board address select consists of a dip switch and a comparator containing internal 20K pullup resistors. One set of inputs to the comparator is tied to address bus lines A16 to A23 on the S-100 bus side of the on-board address line drivers. The other set of inputs to the comparator is connected to the board address select dip switch SW1.

Once the starting address of the board is decided, SW1 is set to reflect the board starting address. The 8 elements of SW1 are capable of defining 256 different 64K boundaries anywhere in the 16Mbyte range (2^8). The boundaries can be thought of as boundary 0 (beginning at address 0) through boundary 255 (beginning at address 16,711,680). The user selects the binary equivalent for the decimal number of the desired boundary by means of the 8 elements of SW1.

The comparator constantly checks the state of each of the address lines to which it is connected against the setting of the address select switch SW1. When a match is found, the comparator indicates the match by activating its output line. This output is used elsewhere on the board to indicate that the on-board 64K of RAM is being addressed.

Care must be taken to ensure that no 2 boards attached to the bus have memory that is set to the same starting address.

2.3.1.2 Mapper RAM

The memory mapper RAM is composed of a 16x12 bit bipolar high-speed static RAM matrix (three 16x4 devices). The upper 4 bit address lines (A12-A15) from the Z-80, when applied to the mapper RAM address input lines, select one of sixteen 12 bit data words. This 12 bit word is applied to the upper 12 bits of the S-100 address bus (A12-A23). These 12 address lines from the mapper, along with the 12 lower address lines from the CPU, create 24 real address lines that are driven out onto the bus, allowing access to the entire 16Mbyte address range by the Z-80.

The 12 bits of data to be loaded into the mapper RAM appear on the 8 data lines (D0-D7) of the data bus and 4 lines (A8-A11) of the address bus of the SBC-300 board. Just after power up the monitor firmware loads the mapper RAM with initial data by applying the appropriate data to the input lines for each address and selecting each of the 16 locations sequentially with A12-A15 of the on-board address bus.

Once the operating system is in control, it can change the content of the mapper RAM at any time as required. Application programs can, in certain circumstances, change the mapper RAM content dynamically as the program executes, allowing immediate access to the entire system memory.

2.4 INTERRUPT CONTROLLER

The SBC-300 provides service for the Non-Maskable Interrupt and the 8 vectored interrupt lines from the S-100 bus as well as the interrupts from the 2 serial communication ports. The interrupt controller prioritizes on-board interrupt requests and the vectored interrupt lines from the bus, generates the least significant 8 bits of the interrupt vector, and starts the interrupt cycle. An interrupt can also be generated in response to the status of several other lines on the bus (jumper selectable by the user) to allow special features to be implemented.

2.4.1 Interrupt Select

An area of the SBC-300 board is reserved for the user to install jumpers to connect any of several lines from other places on the board as well as the S-100 bus to the NMI and vectored interrupt lines. If the SBC-300 is configured to be the bus master, there are 2 lines from the bus that are considered inputs (PWRFAIL and ERROR). As a slave processor, there are 7 jumpers that might be installed in order to put the on-board parity error and attention lines out onto the bus as vectored interrupts.

As a master, the power failure line might be jumpered to the NMI and the parity error line jumpered to one of the vectored interrupt lines so that if devices on the bus detected these conditions, the bus master processor would be immediately notified and could proceed accordingly.

As a slave, the on-board attention line could be jumpered to any of 6 vectored interrupt lines on the S-100 bus. This would allow the interrupt controller on the bus master to easily determine which slave was requesting the interrupt and then generate the appropriate service routine vector. This procedure limits the number of slaves on the bus to 6.

In situations where there are more than 6 slaves on the bus, all slave attention lines could be jumpered to one or a few interrupt lines on the bus. Then the bus master CPU can poll the slaves to determine which one requested the interrupt as part of the interrupt service routine. All slave parity error lines are usually jumpered to a single interrupt line (VI1), and the bus master CPU can poll the slaves to determine the location of the error.

2.5 BUS INTERFACE

The IEEE-696 S-100 bus standard specification describes a universal bus that can support a wide range of processors rather than just a particular processor or family of processors. Signals required to be on the bus were deemed to be the best compromise between ease of implementation and utility. The bus is configured to allow the widest range of processors the most flexibility possible without being completely incompatible with existing systems.

Because the bus is a universal design, it is not expected that all status and control signals from a particular processor will be directly applicable to the bus. Nor will all signals on the bus be directly compatible with the requirements of a particular CPU.

All processor manufacturers have slightly different ways of accomplishing the same task with their machines. Signal names, polarity, phase, and timing might vary radically from one CPU to another even though the bottom line of what gets accomplished is identical. As a result of all this, some of the signals from the Z-80 CPU on the SBC-300 must be processed a little before they can be applied to the bus, and some signals on the bus must be processed a little before the Z-80 can understand them. This is the task of the bus interface.

The bus interface block consists of several smaller blocks, each one handling a portion of the job. As bus master, the SBC-300 supplies all timing and control signals on the bus. The bus timing controller uses signals from the CPU and other areas of the board to synthesize the signals required to run the bus. The Power On Clear (POC) and RESET area simply handles power clears and resets when the SBC-300 is in either mode.

When the SBC-300 is operated in the slave mode, the slave control area translates the bus signals into a form that can be understood by the Z-80 so that the SBC-300 can be properly synchronized to the activity on the bus.

2.5.1 Bus Timing Controller

As bus master, the SBC-300 must supply all timing and control signals for the bus. The IEEE-696 specification calls for a bus that runs almost everything off the rising edge of the clock. The Z-80 runs almost everything off the falling edge of the clock. The solution is to invert the clock and signals out of the CPU and to delay some signals to get the proper timing relationships for the bus. Other signals that the Z-80 does not provide directly are derived from the signals that it does generate.

The status and control signals that must be applied to the bus when the SBC-300 is operated as bus master are derived from the status and control signals from the Z-80 CPU. Through combinational logic (gates), bus standard equivalent signals are constructed from those available from the CPU and applied to the bus. Clocked logic allows the timing relationship of signals to be adjusted to meet the bus specification.

Signals that need to be delayed or have the timing adjusted are derived from a chain of flip-flops. As the input propagates through the chain, different signals are picked off at the appropriate points and sent out to the bus. These signals have a constant and predictable relationship to each other in time and match the signal function and timing relationships required by the IEEE-696 bus specification.

The address bus is derived from two sources. The low order bits (A0-A11) are taken directly from the Z-80, while the high bits (A12-A23) are derived from the memory mapper. The low order address bits are driven directly by the Z-80 CPU and will not hold on the bus as long as the high bits. Therefore, care should be taken to insure that no conflict is created by the short hold time on the low order address bits.

In the slave mode, the SBC-300 must not apply any of the on-board status and control signals to the bus. The bus timing controller causes the output of the gated line drivers for these signals to become a high impedance when the master mode is selected, thereby preventing problems on the bus.

2.5.2 Power On Clear (POC) And RESET

This area performs a few simple tasks. In the master mode, it holds down the RESET line on the bus until the power has come up and stabilized. This makes sure that all items on the bus (including the SBC-300) get a RESET signal **after** the power lines are up far enough for the devices to fully reset. Otherwise, there may be random data in memory cells and registers that could cause trouble if they are not cleared out before normal operation.

The RESET signal is applied to the POC and SLVCLEAR lines on the S-100 bus line only when enabled as a master. The RESET line is also usually connected to some sort of mechanical momentary switch which is accessible to the operator for manual resets.

As a slave processor, the SLVCLEAR and RESET lines are monitored and acted upon instead of driven. When the bus master sends out a SLVCLEAR signal on the bus, the slave processors on the bus decode this signal as an on-board RESET and reset.

2.5.3 Slave Control

The same bus signals which were derived, inverted, and/or time shifted from the Z-80 before they were applied to the bus by the SBC-300, acting as bus master, must be decoded by the SBC-300, acting as a slave processor on the bus, before they can be understood by the SBC-300. The slave control area takes care of this task.

Combinational logic elements (gates) and a Programmed Array of Logic (PAL) device are used as a decoder to condense many of the status and control signals on the bus down to a few signals that are meaningful to the SBC-300 and the on-board Z-80. This requires that some signals be inverted and/or shifted in time.

All the timing signals that the on-board CPU would normally generate for the on-board memory come from the bus during commurication with the slave. Normally the slave is running free with its CPU controlling all aspects of its operation. But when it has to communicate with the bus (through the memory), the bus must control the memory so that everything on the bus will be compatible and synchronized to the bus universal timing. The slave control logic accomplishes this.

Because slave control logic is required to interface the slave to the bus during memory transfers, the slave's memory does not appear (respond) exactly as just another board of **fast** memory. There is an appreciable propagation delay through the several layers of decoding in the slave control area before the appropriate signals are applied to the on-board memory.

The slave control area contains a wait state generator that sends an adjustable number of waits to the bus master CPU to hold things up until the slave decoding logic and memory systems have had a chance to settle and respond. The wait state generator is simply a counter that holds down the RDY line on the bus until it has counted a specific number of cycles from the bus clock. The number of waits sent to the bus master CPU is selected by jumper.

A jumper (W13) selects from 0 to 5 waits to be sent to the master during the slave memory cycle. If there is no jumper installed, there will be no wait states sent during a memory cycle to the slave. The number of waits that should be sent is dependent on the processor clock speed of the bus master and how fast the rest of the memory in the system runs. The bus master inserts waits into an ordinary memory access, the number depending on the speed of the memory installed. The number of waits needed to be inserted by the slave is in addition to the waits generated by the master during an ordinary memory access.

In the master mode, the entire slave control area is disabled because as the master the SBC-300 is supplying the status, timing and control signals to the bus instead of receiving them.

2.6 MASTER/SLAVE CONTROL

The IEEE-696 S-100 specification allows for the connection of 3 main types of devices to the bus. There must be exactly 1 Permanent Bus Master (PBM) on the the bus at all times. Up to 16 Temporary Bus Masters (TBM) and any number of bus slaves are allowed on the bus (short of the 22 maximum slots called for in the specification).

The SBC-300 may be operated as either the PBM or a slave processor. There is a significant difference between one mode and the other in the way the SBC-300 operates and interfaces to the bus. These differences are effected by logic on the SBC-300 board responding to the mode selection jumpers and switch.

2.6.1 Master/Slave Select

The main difference in the activities of an SBC-300 operating in the master mode and the slave mode is that in the master mode, the SBC-300 puts status, timing, and control signal on the bus, while in the slave mode, it is controlled by those signals on the bus. In order to accomplish the necessary changes, a single hardware jumper (W4) is used to tell the CPU whether the board is a master or slave.

In the master mode, the board ignores the status, timing, and control signals on the bus and arranges to drive these lines on the bus instead. In the slave mode, the drivers are disabled and the decoders are enabled so that the signals on the bus control the slave.

2.6.2 Line Termination

The IEEE-696 bus standard provides that pull up resistors for certain lines on the bus shall reside either on the bus (mother board) itself or on the PBM. Since the SBC-300 can be operated either as a PBM or as a slave processor, there is a means of providing the pull up resistors for the bus lines that require them when in the master mode (if the bus does not have them). A group of jumpers must be installed if the SBC-300 is to provide the pull up resistors for the bus. If the SBC-300 is not to supply this pull up function or is operating as a slave processor, these jumpers are to be removed. Table 2-4 shows the line termination jumpers.

2.7 SASI/PARALLEL PORT

The SASI/Parallel port is an 8 bit parallel data port along with 8 status lines. The data port is a bi-directional 8 bit port located at I/O address 70H. The 8 SASI control lines are controlled by the CIO port A, which is addressed as I/O port 7AH. The port is available to the user through connector J4 and can be configured to be any parallel device required. See the Appendix called Parallel Interfaces: SASI/Centronics for a detailed pin out.

2.7.1 CIO

The Zilog 8536 Counter Input/Output chip (CIO) is a parallel I/O device containing two 8 bit parallel ports and one 4 bit port. Internally, the device has three 16 bit counter/timers which are user configurable. The SBC-300 uses port A of the CIO as the status lines for the parallel port, port B as the vectored interrupt controller and port C as general purpose control lines. See the Appendix called CIO Specification for a brief description of the CIO.

Table 2-4. JUMPER CONNECTIONS FOR BUS MASTER OPERATION

(See the Appendix called Jumper Connection Quick Reference Tables for detailed list)

Jumper #	Master	Slave	Notes
W4	OUT	IN	Master/Slave select
W11	IN	OUT	MWRT select
W15	IN	OUT	Pull ups
W19			Slave clear input
1-2	OUT	(manual reset for slave)	Manual slave clear when in position 1-2
2-3	OUT	IN	Standard position for slave
W21	IN	OUT	RDY and XRDY enable

2.8 SERIAL I/O

The SBC-300 contains 2 full duplex serial ports. Port A (J3) is a general purpose port for use with serial printers or any user selectable device. Port B (J2) can be used for a terminal when a video board is not used or as a general purpose port. When a video board is not present, the Monitor checks for a terminal on port B and waits for a response from the port. The ports can be used as DTE or DCE, depending upon the way the configuration jumpers (J5, J6) are wired.

2.8.1 ASCC

The heart of the serial ports is the Zilog 8531 Asynchronous Serial Communications Controller chip (ASCC). This chip contains 2 full duplex asynchronous serial ports along with dual programmable baud rate generators. See the Appendix called ASCC Specification for the programming model.

2.9 POWER REGULATION

The SBC-300 is equipped with an on-board 5 volt linear regulator for use on 8 volt power busses in accordance with the IEEE-696 S-100 standard. The regulator provides a tightly controlled 5 volts for use by the various devices on the board and is immune to minor fluctuations of the 8 volt power bus.

If 5 volt operation is required (instead of the normal 8 volt operation) and an adequate source of well regulated 5 volts is available, a jumper connection will configure the SBC-300 for 5 volt operation.

A separate area of the power supply section provides +/- 12 volts DC from the +/- 16 volts DC power lines on the bus. The +/- 12 volts is used on the SBC-300 board only by the Asynchronous Serial Communications Controller (ASCC) for the standard RS-232 communication protocol. Very little power is dissipated in this area, and so a zener stabilized supply is more than adequate for the purpose.

2.9.1 5 Volt Bus Operation

If the SBC-300 is to be installed on a bus using a **regulated 5 volt** power supply to operate the 8 volt S-100 lines (this deviates from the IEEE-696 specification), the on-board regulator may be bypassed by the installation of a jumper between the input and the output pins of the regulator device. If all items to be attached to the system bus are able to operate with a 5 volt power bus instead of the 8 volts called for in the specification and if a suitable high performance 5 volt regulated power supply is available, it may be advantageous to operate the system on 5 volts.

Operation of the SBC-300 on a 5 volt line eliminates the heat generated by the on-board regulator, allowing a system to be safely operated without a fan. A high performance regulator (such as a switching regulator) for the bus dissipates little or no heat itself and provides generally superior regulation.

***** C A U T I O N *****

The on-board regulator must remain in place when the SBC-300 is operated on an 8 volt power bus, or destruction of the logic devices may result.

SECTION III
THEORY OF OPERATION

3.0 GENERAL

This section covers the general theory behind the subdivisions of the SBC-300 and the basic requirements of the IEEE-696 bus specification. For detailed information about the IEEE-696 specification, the user needs to obtain a copy of the specification from IEEE as it cannot be reproduced in full in this manual.

3.1 MASTER/SLAVE MODES

3.1.1 Masters

There are 2 types of bus masters, permanent and temporary. Each serves a different purpose in the system from the other. The Permanent Bus Master (PBM) is a processor type device that controls the bus and the system, while the Temporary Bus Master (TBM) is usually a DMA device such as a disk or tape controller that obtains control of the bus from the PBM from time to time for the purpose of transferring data to or from the PBM or any slave on the bus.

The acting bus master (permanent or temporary) may initiate bus cycles and, if it is the PBM, may grant control of the bus to a TBM on the bus.

3.1.1.1 Permanent Bus Master (PBM)

As the PBM, the SBC-300 directs all bus activity within the system, driving all data, address, status, and control lines of the bus in accordance with the IEEE-696 bus standard. In this mode, the SBC-300 may utilize any resource that might be attached to the bus including system memory, any I/O device such as a disk controller or a multiple port I/O card, keyboard and video boards, and any slave processor on the bus.

Control of several lines on the bus is retained by the PBM whether it has control of the bus or not. These signals include both clocks, the Power On Clear (POC), and the memory write line and are always controlled by the PBM to avoid timing and synchronization problems with the other items on the bus.

3.1.1.2 Temporary Bus Master (TBM)

A TBM is a device on the bus that may request control of the bus from the PBM to conduct some business that requires the use of the bus. When a TBM gains control of the bus, it becomes the acting bus master until it returns control to the PBM. The TBM does not generate all possible bus cycles nor does it generate all signals required to drive the bus. It may not grant control of the bus to another TBM nor accept an interrupt request.

TBMs acquire control of the bus, conduct as much business as required, and return control of the bus to the PBM. Disk and tape controllers are examples of devices that are considered TBMs.

3.1.2 Slaves

The SBC-300 may be configured as a slave processor and interfaced to the bus as a bus slave. However, not all bus slaves are slave processors.

A slave on the bus responds to bus cycles initiated by the acting bus master. The slave constantly monitors the bus and, if addressed during a particular bus cycle, accepts the data it finds on the bus data lines or applies data to the lines as directed by the control signals present on the bus. At the completion of the bus cycle, the slave resumes its regular activity if it is a slave processor or some similar device, or waits for the next bus cycle if it is a memory board or other unintelligent device.

The slave appears to the bus as simply another block of memory. Each slave only takes up a slot on the bus and a block of memory in the map. The location (address) of the slave in the memory map is selected on the slave board by means of a switch. Slaves can be added to the system until there are no more available slots on the bus or blocks in the memory map.

3.1.2.1 Slave Processors

In the slave processor mode, the SBC-300 operates exclusively out of on-board memory and may utilize any resource that it can access through any means except the S-100 bus (serial ports and the SASI interface). Unless addressed during a bus cycle, the slave processor operates independently of most lines on the bus.

The acting bus master may communicate with the slave processor only through the slave processor RAM which is dual ported. As a slave processor, the SBC-300 may not initiate a bus cycle, and the only way it can access the bus master is if it pulls down an interrupt, wait, or error line to get the attention of the bus master in response to some condition on the board.

3.1.2.2 Slave Processors And Multi-User Systems

Slave processors on the bus spread the intelligence and processing power around, allowing high performance multi-user and/or multi-tasking systems to be easily configured on the S-100 bus. Each slave processor runs at full speed regardless of activity on the bus or in other processors attached to the bus. The only time bus activity impacts a particular slave processor is when it must interrupt the PBM to access some shared resource (hard disk, tape controller, etc.).

This approach offers an excellent compromise between a large, shared processing system with its high cost, inherent response time problems, and all or nobody operation and a locally networked, fully independent array of work stations with its high cost due to duplication of hardware and high speed network communication problems.

In this configuration, each user has essentially a full-featured system operating independently of the bus. This is a vast improvement over previous small multi-user systems where a single processor served every user and attended to system housekeeping as well. That type of system spreads a single 8 bit microprocessor too thinly and causes overall system performance to deteriorate dramatically with each added user.

Previously, the solution to this problem was to run multi-user systems on minicomputers using powerful (and expensive) 16 and 32 bit processors, operating systems, and applications programs. A single user on the system seldom required the power of a large processor--which was good--because the power of the single large processor in a shared system was diluted by the addition of each user. Beyond a certain number, the addition of more users would often cause the performance of the entire system to deteriorate to the point that it was no longer acceptable and a still larger system was required.

More recently, arrays of small, completely independent systems have been linked via one form or another of a high speed serial communication network. However, there has not yet been a lot of satisfaction with the performance of these systems, and the cost is quite high owing to the duplication of hardware (each user has a power supply, floppies, maybe a hard disk, etc.).

A multi-user system consisting of 1 PBM to manage the shared resources attached to the bus and a slave processor for each user offers performance almost independent of the number of users on the system. In addition, this approach allows the use of the enormous amount of familiar and inexpensive application software available to run on the Z-80 and other microprocessors.

3.1.3 Bus Cycle Operations

There are 2 kinds of bus cycles that might occur on the bus, generalized bus cycles and Temporary Master Access (TMA) cycles. All effective communication between 2 devices attached to the bus takes place during a generalized bus cycle (or simply, bus cycle). The process of a TBM gaining control of the bus from the PBM, conducting its business, and then returning control of the bus to the PBM is called a TMA.

3.1.3.1 Bus Cycles

A bus cycle is initiated by a device acting as the bus master and is responded to by a device acting as a bus slave. Communication must occur between the bus master and the addressed bus slave. The device acting as the bus master may be either the PBM or a TBM that has acquired control of the bus from the PBM, and the bus slave may be any addressable slave on the bus.

For example, if the PBM requires some data stored in a memory board somewhere out on the bus, it initiates a bus cycle by applying the address of the required data to the address lines of the bus. The slave (memory board), recognizing its address on the bus address lines, then responds to the control signals put on the bus by the PBM by retrieving the data from the specified address, putting it on the bus data lines, and informing the PBM that the operation is complete. The PBM deselects the slave, ending the bus cycle, and continues its processing.

3.1.3.2 Temporary Master Access (TMA) Cycles

The TMA bus cycle is initiated by the TBM requiring the use of the bus. The PBM turns over control of the bus to the highest priority TBM requesting the bus. Once a TBM gains control of the bus, it may initiate generalized bus cycles as often as its task requires before returning control of the bus to the PBM.

While the TBM has control of the bus, however, another TBM may not gain control of the bus from the TBM acting as bus master. Only the PBM may assign control of the bus to a requesting TBM, and it can only do so it when it has control of the bus. If, for example, the PBM required data from a disk, it would direct the disk controller (a TBM) to read a sector from the disk and load the data into memory beginning at a specific address. The starting address may indicate memory on the PBM board or any memory on the bus.

When the disk controller has read the data and checked for errors, it initiates a TMA cycle by requesting control of the bus. The PBM then releases the bus to the disk controller. The disk controller takes control of the bus and initiates a bus cycle to load the data from the disk into memory, beginning at the starting address requested by the PBM. When the transfer is complete, control of the bus is returned to the PBM, thus completing the TMA cycle.

While the disk controller TBM is the acting bus master, any requests for the bus from other TBMs are ignored. When the current TMA is complete and the bus has been returned to the PBM, control of the bus may be granted to the highest ranking requesting TBM.

3.2 SLAVE PRINTER

Slave priority is determined by jumpering the ATTEN line (W14) to the vectored interrupt bus. Priority is given to slaves as determined by the jumper setup. The bus master then arbitrates and gives priority to the correct slave.

3.3 INTERRUPTS

The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner, and begin a service routine for the peripheral. This service routine usually involves the exchange of data or status and control information between the CPU and the peripheral requesting the interrupt. When the interrupt service routine is complete, the CPU resumes operation exactly where it left off when it received the interrupt request.

There are 2 types of interrupts recognized by the Z-80, a software maskable interrupt (INT) and a Non-Maskable Interrupt (NMI). A control line for each goes into the CPU. The NMI cannot be ignored by the CPU. When this interrupt line goes active, the CPU must accept it. The maskable interrupt function can be enabled or disabled by instructions in the program that the CPU is executing.

3.3.1 Non-Maskable Interrupts (NMI)

An NMI will be accepted by the CPU whenever it is requested. Because of this characteristic, the NMI is usually reserved for indicating very important conditions such as power failure or error detection. The SBC-300 allows the user to select (by jumper) any of several lines from the S-100 bus for connection to the NMI line into the Z-80.

When an NMI is accepted, the CPU ignores the next instruction that it fetches and does a restart to location 0066 HEX instead. This location is usually in ROM and should contain the starting address out in RAM of the service routine for the NMI. What the CPU finds in 66 and the rest of the service routine (if any) is completely at the discretion of the user.

3.3.2 Maskable Interrupts

The maskable interrupt function can be enabled or disabled by the program that the CPU is executing. If there are areas of a program where timing or other constraints would make an untimely interrupt inconvenient or catastrophic, the user has the option of disabling the maskable interrupts by means of an instruction until the sensitive parts of the program are completed. The interrupt line can then be enabled.

There are 3 possible modes of INT response by the Z-80, each selectable by the user. The most powerful and therefore the most popular is the MODE 2 response. When the Z-80 is operated in this mode, the program maintains a table of 8 bit starting addresses for every interrupt service routine in memory. The CPU I register is loaded with the 8 high order address bits of the table by the program. When an interrupt is accepted by the CPU, the low order bits are supplied by the interrupt controller. The high and low order bits form a pointer which points to one of the service routine starting addresses in the table according to the low order address bits.

The CPU then obtains the starting address from the pointed to location in the table, jumps to that address, and executes the service routine that begins at that location. The table can be changed at any time by the program if it is stored in RAM, allowing different service routines for the same peripheral.

3.3.3 Interrupt Enable

There are 2 enable flip-flops (IFF1 and IFF2) in the Z-80 to take care of the status of the maskable interrupt. The state of IFF1 is used to enable and disable the INT line of the Z-80 while IFF2 acts as a storage latch for the state of IFF1 under certain circumstances. When IFF1 is set, interrupts are enabled. When IFF1 is reset, interrupts are inhibited. If the CPU is RESET, both flip-flops are reset and interrupts are ignored. The instruction EI (Enable Interrupts) changes the state of both flip-flops and interrupts are enabled. DI resets both flip-flops and then interrupts are disabled.

When an EI instruction is encountered by the CPU, any pending interrupts will not be accepted until after the instruction following EI has been executed. This prevents any trouble should the next instruction be a RETURN which must be executed before any interrupts are allowed. Both IFF1 and IFF2 are reset when the CPU accepts an interrupt, inhibiting further interrupts until the CPU encounters another EI instruction.

When dealing with maskable interrupts, the state of IFF1 and IFF2 is always equal. The function of IFF2 is to save the status of IFF1 when an NMI is accepted by the CPU. This resets IFF1 only, preventing further interrupts until the EI instruction is executed. After the NMI is serviced, executed instructions may cause the content of IFF2 to be examined, tested, or copied into IFF1 in accordance with executed instructions so that the complete state of the CPU before the NMI occurred can be restored.

3.3.4 Interrupt Control

Interrupt control is handled by the Counter/Timer and Parallel I/O unit (CIO). The CIO is a multi-function device containing 2 general purpose 8 bit bidirectional parallel ports, 1 special purpose 4 bit port, 3 counter/timers, pattern recognition logic and several assorted registers.

The CIO prioritizes the interrupt requests from items on the S-100 bus and from its internal parts that might be programmed to cause an interrupt in response to some condition (counter/timers, for example). The Asynchronous Serial Communications Controller (ASCC) looks after the 2 serial ports and may need to interrupt the CPU from time to time in response to some condition on the serial channels. Conflicts between interrupt requests from these 2 devices are resolved through a daisy chain arbitration procedure.

3.3.4.1 Off-Board Interrupt Prioritization

Service for the 8 vectored interrupt lines from the S-100 bus is implemented through the general purpose 8 bit parallel port B of the CIO. The 8 lines from the bus are applied to the parallel port for prioritization. When a request for interrupt service is received from a device on the bus, the pattern recognition area of the CIO determines which interrupt line was activated and causes the CIO to generate a unique vector for each of the 8 interrupt lines from the S-100 bus and place the vector on the lower 8 lines of the address bus when the CPU accepts the interrupt.

At initialization, the CIO is loaded with a vector starting location. Each interrupt line on the bus, if activated, causes a data word with a unique offset from this starting point to be generated by the internal CIO logic and applied to the low order lines of the address bus upon acceptance of the interrupt request by the CPU. These 8 bits, together with the high order 8 bits supplied from the Z-80 I register, form a 16 bit pointer. This pointer indicates a unique service routine starting address in a table. The CPU reads the starting address from the table and jumps to that location to begin execution of the interrupt service routine specific to the interrupt line activated on the bus.

3.3.4.2 On-Board Interrupt Prioritization

The CIO can generate interrupts based on the condition of the other 8 bit parallel port (A) or any of the 3 on-board counter/timers. Each of these items has a register for holding the starting value for vector generation. When activated, internal logic generates a unique offset from this starting value for each origin of the interrupt request and applies this data to the lower 8 lines of the address bus. Data in other registers and additional logic determine the priority of each of these items.

These items and functions are accessible to the user (programmer) and, through implementation, allow the overall function of the CIO to be radically altered to fit almost any special application. All the functions, modes of operation, and features of the CIO are selected, programmed, and exploited via the programming of the CIO. This is normally done by the monitor firmware at power up initialization, but the CIO can be dynamically reconfigured easily by the Operating System (OS) and applications program as required during the execution of programs.

3.3.4.3 On-Board Interrupt Arbitration

There is an arbitration procedure to decide which on-board device will be serviced by the CPU when requests for interrupt service are received from more than one peripheral at the same moment. The CIO controls and prioritizes interrupts requested by items on the bus (8 or maybe more) and may be configured to generate interrupts based on other criteria as well (counter/timer results, etc.). The ASCC controls communication with the outside world through the 2 serial ports. These are the only sources of interrupts to the CPU that are generated in response to events occurring off the board.

Each device capable of pulling the CPU interrupt line down in response to a request for an interrupt from a peripheral (on the bus or external to the system) is provided with priority arbitration logic. There are 2 pins on each of these devices that allow priorities to be set and disputes settled. One is the Interrupt Enable Input (IEI) and the other is the Interrupt Enable Output (IEO).

The design of the system determines the priority of each interrupt control device on the board, and the board is etched accordingly. The IEI (input) line of the highest priority device is tied directly to the positive power bus (permanently logic high). The IEO (output) line of the highest priority device is connected to the IEI line of the next highest priority device whose IEO line is connected to the IEI line of the next highest priority device and so on. This forms the daisy chain.

The Interrupt Acknowledge (INTACK*) line is connected to all devices on the arbitration daisy chain and signals that the CPU has accepted the interrupt and is ready for the daisy chain to determine which device has priority. This signal also causes the winner of the arbitration to apply the vector to the address lines. When the current interrupt service is complete, the interrupt acknowledge line goes inactive and the CPU continues with the task suspended by the interrupt.

There are several internal registers in the CIO--Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE)--that the CPU can set or reset to enable or disable the interrupt initiating ability of the device. There is also a Master Interrupt Enable (MIE) register that enables or disables the entire device with a single bit. The contents of these registers reflect the condition of the device.

When a device needs to pull down the CPU INT line to initiate an interrupt service routine, it must first examine the status of its internal control registers. If it finds that it has an interrupt pending (IP bit high), that the device is enabled (MIE and IE bits high), and that there is not another interrupt under service (IUS bit low), then the IEI line is examined.

If the device contains several possible sources of interrupts (as does the CIO), an internal daisy chain determines interrupt priority. The internal daisy chain is the same sort of arrangement as the external one. The IEI line must be high for a device (or part of a device) to assume it has the highest priority. Once the internal prioritization is determined, the device inspects the IEI line to the device.

If the IEI line is high, it assumes it has highest priority of all devices needing an interrupt at that moment; and, if all other conditions are met, it initiates an interrupt by pulling down the INT line to the CPU. If the INT function of the CPU is enabled, the CPU will accept the interrupt and return an INTACK* signal.

The INTACK* signal is applied to all devices in the daisy chain, causing every device that meets the conditions required to request an interrupt (enabled, IP set, etc.) to send a low out its IEO line to disable all devices of lower priority. Each IEO is the IEI to the device having the next highest priority in the daisy chain. Each device thus passes this low to the next device on the chain and so on down to the end of the chain.

When the chain settles, only 1 device will have its IP bit set high (indicating need to initiate an interrupt) and its IEI line high (indicating it is the highest item in chain requesting interrupt). This device wins the arbitration, applies its vector to the bus, and has its interrupt serviced by the CPU.

The highest priority device will always have a logic high on its IEI (it is wired to logic high); so when it is enabled and its IP bit is set, it will always win the arbitration. The highest priority device on the SBC-300 is the CIO and the next (last) item in the chain is the ASCC. This ranking is determined by the board etch layout and is fixed.

3.3.5 Interrupt Acknowledge (INTACK*)

The INTACK* signal is derived from the M1* and MREQ* lines from the CPU. The Z-80 examines the INT line at the rising edge of the last clock cycle at the end of each instruction. If the CPU finds the INT line active and it has been enabled by the software, it generates a special M1* cycle consisting of the activation of the M1* line and then the IOREQ* line (instead of the MREQ*).

This combination, or rather the combination of the M1* signal and the **absence** of the MREQ*, is decoded by the logic of the interrupt acknowledge area to produce the INTACK* signal. During normal processing, this combination will never occur.

INTACK* is applied to the on-board interrupt controllers (CIO and ASCC) to indicate that priority arbitration should begin, the winner applying the low order vector to the address lines, and to lock out any additional interrupt requests from those locations. The CPU releases the INTACK* line after it has completed the interrupt service routine and is ready to continue processing or accept another interrupt.

3.4 MAPPER CONCEPTS

The 16 address lines of the Z-80 allow it to address or "see" only a 64K page of memory. This 64K page can be thought of as a window 64K high that the Z-80 can look through. The Z-80 cannot see out above or below this window--only through it--and considers the address of the lowest location visible through the window to be 0 and the address of the highest location visible through the window to be 64K (65,535). To access locations that are visible through the window, the Z-80 will generate an address accordingly (0 through 65,535).

The memory mapper allows the repositioning (moving up or down) of this 64K window in 4K increments so that the Z-80 can see into any area of the installed memory. This means that the 64K of memory that the Z-80 can see and address through the window can physically be anywhere in the installed memory beginning at a 4K boundary. When the 64K page is moved about, it always comes to rest with the beginning address that the Z-80 can see (0) at some multiple of 4K (4K, 8K, 256K, 1M, etc.) in the system memory and the highest address that the Z-80 can see, 64K higher.

If, for example, a 1Mbyte (1,048,576 bytes) memory board is installed on the bus, the SBC-300 on-board 64K of RAM might be assigned to address 0 through 65,535 and the 1Mbyte board to address 65,536 through 1,114,111 (1Mbyte+64K). In this situation, there are now 1,114,110 bytes of contiguous logical and physical memory extending from 0 to 1,114,111. But as far as the Z-80 is concerned, there is only 64K out there--it can only see a 64K block through its window.

The mapper, under direction of the monitor firmware or the OS, can move the 64K window that the CPU looks through to any area of the installed memory. The lowest location that the Z-80 can see and address through the window will be located at a 4K boundary and is considered by the Z-80 to be location 0. So, in the example, if the mapper moves the window up into the memory board area, to begin at 81920 (64K+16K), a 4K boundary, the Z-80 can directly address all physical memory between 81,920 and 147,455.

Even though this area begins far above the addressing capabilities of an unaided Z-80, it is still only a 64K range which is exactly defined by the 16 bit physical addressing of the Z-80 (0 to 65,535). It is very unlikely that a greater range of memory will need to be addressed at any one time. However, the versatility of the mapper is such that if a greater range is required, the window can be moved to different locations between memory accesses, thereby simulating a 16Mbyte range. The implementation and efficiency of this operation is dependent on the characteristics of the monitor firmware, OS, and application program.

The end result of using the memory mapper is similar to that obtained by a page select. Both allow different areas of a large memory to be accessed by the limited addressing capability of an 8 bit CPU. The main difference is in the implementation and flexibility of the 2 methods. The page select brings pages of memory into the range of the CPU by changing the addresses of the selected block to coincide with the addressing range of the CPU. As a new block is paged in, the block that was previously in the addressing range of the CPU is paged out.

Memory mapper operation simulates the addressing abilities of many 16 bit processors. The mapper creates a real 24 bit address on the S-100 address lines corresponding to the location of the desired area of memory instead of just moving a block of memory into and out of the addressing range of the CPU. Using a mapper is somewhat faster than page selecting, and the physical and logical addresses of all memory elements on the bus are never manipulated by the mapper and remain constant.

When the mapper is being used to generate an address beyond the 64K capability of the Z-80, its operation is completely transparent to the CPU on the SBC-300 board and all other items on the bus.

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION IV
OPERATIONS

4.0 GENERAL

The SBC-300 can be configured to operate in several different bus modes. It can be configured to operate in accordance with the IEEE-696 bus specification or, through the use of jumpers, can be used with older systems which do not conform to the IEEE-696 specification.

4.1 INSTALLATION

4.1.1 IEEE-696

The SBC-300 can be configured to operate as either the bus master or a bus slave as specified by the IEEE-696 specification. Refer to the Appendix called Jumper Connection Quick Reference Tables for the standard jumper configurations.

4.1.2 SDSystems S-100 And Others

To configure the SBC-300 to run with older SDSystems boards (EXPANDORAM III, VERSAFLOPPY II, VDB-8024), first jumper the board as a bus master (see the Appendix called Jumper Connection Quick Reference Tables). Then change jumper W6 to be on pins 2 and 3 (lower position). This connection jumpers the system clock (Ø1) to bus pin 25 (pSTVAL*) and allows proper operation.

4.1.3 Initial Checkout

After setting the jumpers and configuring the serial port, the address switch must be set. The address switch selects which 128K block the master resides in (64K block for a slave). The 8 position dip switch corresponds to the 8 most significant bits of the 24 bit extended address bus. Up is a logic 0 and down is a logic 1 when reading the switches. The leftmost switch (S1) is the LSB (A16).

	A16							A23	
Set to address		*	*	*	*	*	*	*	ON
01XXXXH	*								OFF
(H=Hex;									
X=Don't care)	LSB							MSB	

Figure 4-1. ADDRESS SWITCH

Next, insert the SBC-300 into the bus and either connect a serial terminal to J2 or insert an SDSystems VDB-8024 board into the bus (the Monitor will talk to either one). Apply power to the system and observe the sign-on message.

```
SDSYSTEMS SBC-300 MONITOR Vx.x
```

Figure 4-2. SIGN-ON MESSAGE

After the sign-on message is displayed, the Monitor performs a nondestructive RAM test of all the on-board RAM and either returns the Monitor prompt (.) or an error message giving the bad memory location.

```
MEMORY ERROR    aaaa bb cc
```

aaaa = memory address, bb = data written, cc = data read

Figure 4-3. MEMORY ERROR MESSAGE

4.2 SERIAL I/O

The serial ports (J5 and J6) can be configured to appear as Data Terminal Equipment (DTE) or Data Communications Equipment (DCE) through the installation of a 16 pin dip header in the proper location.

During initialization the SBC-300 monitor checks serial port B (I/O connector J2, jumper J5) for a terminal or the bus for a VDB-8024 board. The SBC-300 responds to whichever type of terminal communicates first.

To talk to a standard RS-232 terminal, J5 must have a header installed with a minimum of pins 1-16 and 2-15 connected. This connects transmit and receive data together and will allow most terminals to communicate with the SBC-300. The SBC-300 does not require any of the handshake signals (RTS, CTS, DSR, DTR) to be connected to function. However, some terminals may require certain of these lines to be active so that these signals are available at the header for the user to jumper as desired.

SECTION V
MONITOR

5.0 GENERAL

The SDSystems Monitor for the SBC-300 must be ordered to support the specific disk controller used. Contact SDSystems for available Monitors.

The SD Monitor program was developed by SDSystems to assist the SBC-300 user in software and hardware debug. Commands are included to provide control and testing of memory and input/output ports. Software debug is aided with breakpoint, single step, and register examine commands. Additionally, when using the VERSAFLOPPY II or VFW-III disk controller board, commands are provided to support reading and writing to disk from memory and booting up a floppy disk based operating system.

5.1 SD MONITOR START-UP

The SD Monitor start-up is as follows:

1. Power on or depress system reset.
2. Response:

   SDSYSTEMS SBC-300 MONITOR V2.0

   (3-4 second pause while RAM memory test executes. The address of any memory location which fails the test is printed followed by the test byte and memory byte.)
3. Reply:

   The operator may proceed to use any or all of the SD Monitor commands as described in this document.

NOTE: On start-up the Monitor program will accept input from either an SDSystems VDB-8024 or a 9600 baud serial device connected to the SBC-300 serial port B (i.e. connector J2 at top left center of board). The source from which the first input is received will be recognized as the console device until the next system reset occurs.

5.1.1 Command Syntax

Each SD Monitor command is initiated by entering an alpha letter (see Table 5-1) followed by optional hexadecimal operands. The number of operands varies and when there are more than one, each must be separated by a space or a comma. Note, however, that a space is automatically inserted when the command letter is keyed in and the user must not enter a space before the first operand.

Table 5-1. SD MONITOR COMMAND SUMMARY

Command	Description
A nbbb	Map the bbbth 4K memory block in 16Mbyte address space to Z-80's nth 4K memory block
B aaaa	Set breakpoint at aaaa
C n	Boots up operating system from disk unit n
D aaaa bbbb	Display memory from aaaa to bbbb
E aaaa	Examine memory at aaaa
F aaaa bbbb dd	Fill memory from aaaa to bbbb with dd
G aaaa	Go to program at aaaa
H aaaa bbbb	Hex sum and difference
I pp nn	Input from port pp, nn times
J aaaa bbbb n	Test RAM pages aaaa through bbbb in the 16Mbyte address space of the 16Mbyte memory using the Z-80s nth 4K memory block
L aaaa bbbb cc_0 cc1 ccn	Locate string in memory from aaaa to bbbb
M aaaa bbbb cccc	Move memory block from aaaa to bbbb to block starting at cccc
O pp dd nn	Output dd to port pp, nn times
P pp	Examine port pp
R aaaa fd tt ss nn	Read from disk to memory at aaaa
S aaaa nn	Single step at aaaa, nn times
T aaaa bbbb	Test RAM from aaaa to bbbb
V aaaa bbbb cccc	Verify (Compare) memory blocks
W aaaa fd tt ss nn	Write data to disk from memory
X a b	Examine CPU registers (map)
Y aaaa bbbb	Yank a hex file into memory
Z fd n	Format a disk (f=format, d=unit, & n=skew factor)

When entering an operand, only hexadecimal characters (0-9, A-F) are accepted. If a non-hex character is entered, a question mark (?) is printed and the command is aborted. If, while entering an operand, a wrong number is entered, the user may start over, as only the **last 4 digits** of an operand are used. **Leading zeroes** are assumed if less than 4 digits are entered. Alpha characters may be upper or lower case.

Examples:

```
.B 6A defaults to 006A (Hex).
.B 123456 defaults to 3456 (Hex).
```

All command lines are terminated with a carriage return (<cr>) which causes the command to execute. To abort a command before the carriage return (<cr>), enter a period (.).

5.2 MEMORY COMMANDS

The SD Monitor contains nine (9) system memory commands providing: memory management, display, modification, block fill, extended memory test, string location, block move, Random Access Memory (RAM) test and block comparison.

5.2.1 Memory Management - "A" Command

SYNTAX .A naaa mbbb ... qeee

Where each of n, m, ...q is the most significant digit of a Z-80 4 digit hex address, and each of aaa, bbb, ...eee is the most significant half of a 16Mbyte 6 digit hex address.

The purpose of the "A" command is to manage memory in blocks of 4K bytes.

Although the Z-80 can address only 64K bytes (i.e. 16 4K blocks) of memory, the SBC-300's memory management capability allows access to 16Mbytes. In order to make this 16Mbytes of memory accessible to the Z-80 microprocessor, sections of the 16Mbyte address space are mapped into the Z-80 64K address space in blocks of 4096 bytes.

The first 64K of the 16Mbyte space (000 000 thru 00f fff) is reserved for the SBC-300 on board ROM. The second 64K (010 000 through 01f fff) addresses the SBC-300 on board RAM.

A maximum of 10 operands can be entered on a single command line.

Note that the 4K blocks need not be continuous or unique.

Example:

```
.A 1040 305f
```

After executing the above Monitor command, a memory reference by the Z-80 to location 1nnn will actually access location 040nnn in the 16Mbyte memory space; and, similarly, the Z-80 will see 05fnnn in the 16Mbyte as its memory address 3nnn.

NOTE: Extended memory addressing is effective only to the extent that it is recognized by a memory device. That is, given a memory device with 16 bit addressing capability, the above Monitor command would make memory location f001 of the memory device accessible to the Z-80 as address 3001 and, similarly, 0fff as 1fff.

In addition, alternate memory management techniques used by other memory devices are not directly affected by the SBC-300's memory management capability. The SDSystems Expandoram III memory board, for example, is a memory device which is limited to a 16 bit address, but which has 256K bytes of RAM, the 256K bytes of RAM being accessible by a bank select via an I/O port. Whichever memory section of the Expandoram III is currently selected is the memory that the SBC-300's memory management will access. Since the Expandoram III is a 16 bit addressable device, any 24 bit address will access it unless no bank is selected (i.e. a bank which does not exist is selected). Note, however, that a 24 bit memory address will access only the Expandoram III if the address is not valid for any other memory devices on the bus.

5.2.2 Memory Display - "D" Command

SYNTAX .D aaaa bbbb <cr>

Where aaaa is the hexadecimal value of the memory address to start displaying from, and bbbb is the optional end address. If bbbb is omitted, 256 bytes are displayed starting at aaaa.

When the specified memory is displayed, the Monitor waits for the user to enter a "space" to display the next 256 bytes or a period (.) to return to the SD Monitor command mode.

The memory is displayed 16 bytes per line, hexadecimal and ASCII form, with the starting address (hex) at the beginning of each line. The ASCII output displays periods (.) for all non-printable characters.

The space bar can be used to toggle a pause in execution.

If the user wishes to terminate a display before completion, a period (.) is entered to return to the SD Monitor command mode.

5.2.3 Examine Memory - "E" Command

SYNTAX - .E aaaa <cr>

Where aaaa is the hexadecimal memory address at which the user wishes to begin examination and/or modification.

The purpose of the "E" command is to print the memory address and data for user examination. The user may then enter new data and a carriage return (<cr>) to change the data in that memory location. The Monitor will then advance to the next memory location and display the contents. A period is used at any time to return to the SD Monitor command mode.

The "up caret" (^) is used in lieu of the carriage return after data entry to re-examine the same address allowing verification of the change. If the "up caret" (^) is used without data entry, the previous memory address is examined.

Note that the memory is not altered unless an operand (hex data) is entered before the carriage return or "up caret" (^). If no hex data is entered before the carriage return, the next location is examined. If no data is entered before the "up caret" (^), the previous location is examined.

Note also that if a period (.) is entered after a hex data operand, memory **is not** altered before returning to the SD Monitor command mode.

5.2.4 Fill Memory - "F" Command

SYNTAX - .F aaaa bbbb cc <cr>

Where aaaa is the starting address, bbbb is the ending address and cc is the fill data.

The purpose of the "F" command is to cause the hex data (cc) to write to every Random Access Memory (RAM) location from aaaa through bbbb.

5.2.5 Extended Memory Test - "J" Command

SYNTAX - .J aaaa bbbb c <cr>

Where aaaa and bbbb are the 4 most significant hex digits of a 24 bit memory address, and c is the most significant hex digit of a 16 bit address.

The purpose of the "J" command is to perform a modified incrementing memory test on a range of 256 byte pages anywhere in the SBC-300's 16Mbyte address space. aaaa is the address of the start page, and bbbb is the address of the end page. If not entered, bbbb defaults to aaaa.

In order to test a section of the 16Mbyte memory, the Z-80 needs to map the section into its address space. The c parameter specifies the 4K memory region for the Z-80 to use. c defaults to 1 (i.e. the Z-80 addresses 1000 through 1ffff will be used to map in sections of the 16Mbyte memory for testing. If specifying a value for c, do not use any of the blocks occupied by the Monitor program (i.e. blocks e and f if using a 4K Monitor, blocks d, e, and f if using an 8K Monitor).

All errors are reported to the console with the 6 digit hex memory address, data written, and data read back being displayed for all bad locations.

At the end of each pass through the range specified, a "P" is displayed on the console.

At any time during the test, the space bar can be used to toggle a pause in execution.

A period (.) must be entered to terminate the test.

NOTE: After execution of a "J" command, the mapped state of the Z-80 4K address space used for the test is unknown.

5.2.6 Locate String - "L" Command

SYNTAX - .L aaaa bbbb cc_0 cc_1 cc_2....cc_n <cr>

Where aaaa and bbbb are hex memory addresses and cc_0 through cc_n are single byte hex data values.

The purpose of the "L" command is to search memory starting at aaaa and ending at bbbb for the string (up to 6 bytes) specified by the hex values cc_0 through cc_n.

Each time the string is located, the address of the first byte is printed on the console, followed by the first 16 bytes of data found at that address in hexadecimal and ASCII.

The space bar can be used to toggle a pause in execution.

The search is terminated at any time by entering a period (.) on the console.

5.2.7 Move Memory - "M" Command

SYNTAX - .M aaaa bbbb cccc

Where aaaa, bbbb and cccc are hexadecimal memory addresses.

The purpose of the "M" command is to cause the memory block starting at address aaaa and ending at address bbbb to copy to the memory block starting at cccc. bbbb must be greater than aaaa.

5.2.8 Memory Test - "T" Command

SYNTAX - .T aaaa bbbb <cr>

Where aaaa and bbbb are hexadecimal memory addresses.

The purpose of the "T" command is to perform a modified incrementing memory address test on the Random Access Memory (RAM) starting at aaaa and ending at bbbb.

All errors are reported on the console with the memory address, data written and data read back being displayed for all bad locations.

At the end of each complete pass on the specified memory block, a "P" is displayed on the console.

At any time during the test, the space bar can be used to toggle a pause in execution.

A period (.) must be entered to terminate the memory test.

5.2.9 Verify Memory - "V" Command

SYNTAX - .V aaaa bbbb cccc <cr>

Where aaaa, bbbb and cccc are valid hexadecimal memory addresses and bbbb is greater than aaaa.

The purpose of the "V" command is to compare the memory block starting at aaaa and ending at bbbb with the memory block starting at cccc.

Any differences in the memory blocks are reported on the console with the first block address with data followed by the second block address with data.

The space bar can be used to toggle a pause in execution.

The verification is terminated by entering a period (.) at any time.

5.3 INPUT/OUTPUT COMMANDS

The SD Monitor contains 3 commands for reading and writing to input and output ports: "I" command for input from ports, "O" command for output to ports and "P" command for port examination.

5.3.1 Input From Port - "I" Command

SYNTAX - .I pp nn <cr>

Where pp is the input port address and nn is the number of times the port is read.

The purpose of the "I" command is to read the input port "pp" and display the data nn times. If "nn" is omitted, the default is one (1) read. If "nn"=0, the port is continuously read and displayed until a period (.) is entered to terminate the operation. The space bar can be used to toggle a pause in execution.

5.3.2 Output From Port - "O" Command

SYNTAX - .O pp dd nn <cr>

Where pp is the port address, dd is the data to output, nn is the number of times to output the data.

The purpose of the "O" command is to write the data defined by dd to port pp. The data is written the number of times defined by nn. If nn is omitted, the default is one (1). If "nn"=0, the data is continuously written until a period (.) is entered to terminate the operation. The space bar can be used to toggle a pause in execution.

5.3.3 Port Examine - "P" Command

SYNTAX - .P pp <cr>

Where pp is the hexadecimal port number (address) to begin examining and/or modification.

After the command is entered, the specified input port address and data are printed. The user may then enter new data and a carriage return (<cr>) to change the data in that input/output port. The monitor will then advance to the next input/output port number and display the contents. A period can be entered at any time to return to the SD Monitor command mode.

The "up caret" (^) may be used in lieu of the carriage return after data entry to re-examine the same input/output port. If the "up caret" is used without data entry, the previous input/output port is examined.

Note that the port is not altered unless an operand (hex data) is entered before the carriage return or "up caret" (^). If no hex data is entered before the carriage return, the next input/output port is examined. If no hex data is entered before the "up caret" (^), the previous input/output port is examined.

Note also that if a period (.) is entered after a hex data operand, the input/output port is not altered before returning to the SD Monitor command mode.

5.4 PROGRAM CONTROL COMMANDS

The SD Monitor provides several commands designed to facilitate rapid Z-80 software development.

The user has total control of all the Z80 Central Processing Unit (CPU) registers via a register map in memory. This map is loaded into the Z-80's registers each time a program is executed from the monitor. Conversely, each time a breakpoint is encountered, the Z-80 CPU registers are saved in the register map and displayed on the console.

During single step operation, the registers are loaded and saved in between each instruction step.

The user may also set up the register map using the "E" command. Table 5-2 contains the memory addresses of each of the Z-80 registers in the map.

All of the register map is preserved through system resets except for the Stack Pointer (FFE6-FFE7), the Interrupt Enable Flag (FFFA) and the Program Counter (FFFE-FFFF).

Table 5-2. SD MONITOR REGISTER MAP

Memory Address (Hex)	Register	Description
FFE6	SP (LSB)	Stack Pointer Lower Half
FFE7	SP (MSB)	Stack Pointer Upper Half
FFE8	IY (LSB)	Index Reg IY Lower Half
FFE9	IY (MSB)	Index Reg IY Upper Half
FFEA	IX (LSB)	Index Reg IX Lower Half
FFEB	IX (MSB)	Index Reg IX Upper Half
FFEC	L'	Alternate L
FFED	H'	Alternate H
FFEE	E'	Alternate E
FFEF	D'	Alternate D
FFF0	C'	Alternate C
FFF1	B'	Alternate B
FFF2	F'	Alternate Flags
FFF3	A'	Alternate A
FFF4	L	L Register
FFF5	H	H Register
FFF6	E	E Register
FFF7	D	D Register
FFF8	C	C Register
FFF9	B	B Register
FFFA	IF	Interrupt Enable Flag (04=enabled)
FFFB	I	I Register
FFFC	F	Flags
FFFD	A	A Register
FFFE	PC (LSB)	Program Counter Lower Half
FFFF	PC (MSB)	Program Counter Upper Half

5.4.1 Breakpoint - "B" Command

SYNTAX - .B aaaa <cr>

Where aaaa is the address at which to insert the breakpoint.

The purpose of the "B" command is to insert a software breakpoint in the user's RAM based program. The breakpoint consists of a 3 byte jump instruction replacing the user's code at the specified address. If "aaaa" is omitted, any existing breakpoint is removed.

When the breakpoint is encountered, the user's code is restored and the CPU registers are displayed on the console. The monitor then enters the single step mode and the user may proceed as discussed under Single Step, Subsection 5.4.3, or may enter a period (.) to return to the monitor command mode.

The monitor only tracks one breakpoint at a time. If a breakpoint is inserted while another is still in memory, the previous one is automatically removed before inserting the new one. Note that this results in single step execution also removing any existing breakpoint.

If the user inserts a breakpoint and a system reset occurs without the breakpoint having been executed or removed, the breakpoint will still be imbedded in the user's code. In this case, the monitor will not remove the breakpoint or restore the user's code unless the breakpoint is flagged by writing a 01 to location ffc0 (the "E" command may be used to do this). The breakpoint will then be removed by either executing it or by using the monitor "B" or "S" commands. Alternatively to flagging the breakpoint, the user may, of course, restore his code manually either by using the "E" command or by moving it back to the breakpoint address from locations ffc1-ffc3.

5.4.2 "GO" To Program - "G" Command

SYNTAX - .G aaaa <cr>

Where aaaa is the hexadecimal address at which the user wishes to begin execution.

The purpose of the "G" command is to begin execution of a program anywhere in memory (aaaa).

When the "G" command is entered, the Random Access Memory (RAM) register map is loaded into the Z-80's registers before executing the address aaaa.

A program is resumed after a breakpoint (or single step) by omitting the aaaa. This causes execution to begin at the address defined by "PC" in the register map (as displayed by "X" command).

5.4.3 Single Step - "S" Command

SYNTAX - .S aaaa nn <cr>

Where aaaa is the address to begin stepping and nn is the number of steps to trace.

The purpose of the "S" command is to single step through Random Access Memory (RAM) based programs, displaying all the CPU registers after each step. If the nn is omitted, the default value is 1. If the aaaa is omitted, the default value is the previous "PC" loaded into the Random Access Memory (RAM) register map either by a breakpoint, the memory examine "E" command, or a previous single stepping sequence.

When the "S" command is entered, the Z-80 registers are displayed with the registers labeled over each column. The user may then step 1 instruction by entering a carriage return. If the user enters a "space," the Monitor steps 11 steps along with a heading and register display. The user must enter a period (.) to return to the SD Monitor command mode.

The space bar can be used to toggle a pause in execution.

Once back in the SD Monitor command mode, all commands may be used. Single stepping is resumed at the address left off by entering "S <cr>" with no operands.

5.4.4 Register Examine - "X" Command

SYNTAX - .X a b <cr>

Where "a" is the heading print option and "b" is the length option.

The purpose of the "X" command is to examine the Random Access Memory (RAM) register map at any time.

If "a" is omitted, the registers are displayed without a label heading.

If "a" is non-zero, a label heading is printed above the registers.

If "b" is entered, it sets the register display length.

If b=0, it sets the display to short mode; if b=1, it sets the display to the long mode (short mode only displays "PC" and "AF"). This effects breakpoint and single step in addition to the "X" command.

The display mode remains set until the "X" command is used to change it.

Note that if "b" is omitted, the display mode is not altered.

The user must remember also that the Random Access Memory (RAM) register map is only altered by a breakpoint, single step or manual changes via the "E" command.

5.4.5 Hex Arithmetic - "H" Command

SYNTAX - .H aaaa bbbb <cr>

Where aaaa and bbbb are any 2 hexadecimal numbers.

The purpose of the "H" command is to display the hexadecimal sum and difference of aaaa and bbbb. The sum aaaa + bbbb is displayed first preceeded by a "+." The difference (aaaa - bbbb) is displayed second, preceeded by a "-.".

5.4.6 Command Processing - "Y" Command

SYNTAX - .Y aaaa bbbb

Where aaaa is the optional override load address and bbbb is the optional byte count.

The purpose of the "Y" command is to provide a means of loading intel hex data from dart port a into memory starting at the address specified by the incoming data stream and (optionally) loading a specified number of bytes.

5.5 DISK UTILITY COMMANDS

The SD Monitor provides several commands that are useful in a disk based system. These commands are only operative when a VERSAFLOPPY II or VFW-III disk controller board is present in the SDSystems microcomputer.

These commands operate on both mini and standard drives. All parameters relating to drive type are contained in the SBC-300 PROM.

5.5.1 Boot Up - "C" Command

SYNTAX - .C a <cr>

Where a is the drive number 0 through 3.

The purpose of the "C" command is to boot up a floppy disk operating system from the specified drive. If the drive number is not specified in the SD Monitor .C command, it defaults to 0.

NOTE: A compatible disk operating system diskette must be in the defined drive before entering this command.

Once the boot up is complete, the SD Monitor is exited and the disk operating system is activated causing the associated prompt to display on the console.

If nonrecoverable errors are encountered, the message COLD BOOT ERROR is displayed and the user is returned to the Monitor command mode.

5.5.2 Read From Disk - "R" Command

SYNTAX - .R aaaa fd tt ss nn

Where aaaa is a valid RAM memory address, f is the format type (see Table 5-3), d is a valid drive number, tt is a valid track number (base 0), ss is a valid sector number (base 1) and nn is the number of sectors to read into memory (up to FF). All parameters are required.

The purpose of the "R" command is to provide a means of reading data starting at the specified drive, track and sector into the RAM buffer (starting at aaaa). The number of bytes is determined by nn times the sector size.

Table 5-3. DISK FORMATS

	Bit #							
	f				d			
Format Code	7	6	5	4	3	2	1	0
Unit No.							X	X
Not Used					X	X		
No. of Sides				X				
Size (8" or 5")			X					
Sector Size	X	X						

Unit No.	No. of Sides	Size	Sector Size
00 = 0	0 = Single sided	0 = 8"	00 = 128 bytes
01 = 1	1 = Double sided	1 = 5"	01 = 256 bytes
10 = 2			10 = 512 bytes
11 = 3			11 = 1024 bytes

EXAMPLES:

Code 10 (0001 0000) = 8" double sided single density unit 0
Code 23 (0010 0011) = 5" single sided single density unit 3
Code 32 (0011 0010) = 5" double sided single density unit 2
Code 41 (0100 0001) = 8" single sided double density unit 1
Code 53 (0101 0011) = 8" double sided double density unit 3

5.5.3 Write To Disk - "W" Command

SYNTAX - .W aaaa fd tt ss nn

Where aaaa is a valid memory address, f is the format type (see Table 5-3), d is a valid drive number (0 to 3), tt is a valid track number (base 0), ss is a valid sector number (base 1), and nn is the number of sectors to be written from memory (up to FF). All parameters are required.

The purpose of the "W" command is to provide a means of writing data starting at the specified drive, track and sector from the memory starting at aaaa. The number of bytes written is determined by nn times the sector size.

5.5.4 Format A Disk - "Z" Command

SYNTAX - .Z fd n

Where f is the format type (see Table 5-3), d is a valid drive number (0 to 3), and n is the skew factor.

The purpose of the "Z" command is to provide a means of formatting a disk according to the formats in Table 5-3. The skew factor n, if not specified, defaults to a value of 1.

5.6 CIO INITIALIZATION AND USE

A detailed discussion of programming the CIO chip is beyond the scope of this manual and therefore only a brief discussion of its use is given. A summary of the CIO registers can be found in the Appendix called CIO Specification. The specific initialization code used by version 1.2 of the SBC-300 Monitor can be found in the Appendix called Monitor Locations.

For more detailed information on the 8536, see the Zilog publication "Z8036 Z-CIO/Z8536 CIO Counter/Timer and Parallel I/O Unit Technical Manual."

5.6.1 Accessing The CIO Internal Registers

Access to the CIO internal registers is via an internal pointer register. Access to the internal pointer register is via the CIO internal state machine logic. The state machine can determine whether accesses to the CIO are via the data ports (SBC-300 ports 78H-7AH) or via the control port (SBC-300 port 7BH). Reads and writes to the data ports do not effect the state of the CIO; those to the control port do.

There are three states possible in the CIO: the reset state (which should only be active during a SBC-300 reset), state 0, and state 1. The latter two are the normal operating mode for the CIO.

When in state 0, reads to SBC-300 port 7BH access the CIO register currently addressed by the CIO pointer register and leave the CIO in state 0; writes to port 7BH update the internal CIO pointer register and leave the CIO in state 1. When in state 1, reads or writes to port 7BH access the CIO register currently indicated by the internal register pointer and return the CIO to state 0. Since any read to port 7BH, whether in state 0 or in state 1, always leaves the CIO in state 0, a read can be used to force the CIO to state 0. State 1 suspends many internal CIO operations, especially those related to interrupt generation and control; therefore, the CIO should not be left in state 1.

Note that because the CIO operates as a state machine, it is important that code which accesses the CIO internal registers not be interrupted if the interrupt itself would result in access to the CIO's internal registers. If such an interrupt is possible, the user should disable interrupts at the Z80 during CIO internal register access.

5.6.2 SBC-300 Monitor And CIO Initialization

Version 2.0 of the SBC-300 Monitor program performs minimal configuration on the Zilog 8536 CIO chip. Port A and the counter/timers are left in a post-reset condition, port B is configured as an interrupt controller for the S-100 vectored interrupt lines, and port C is configured in accordance with necessary hardware requirements.

5.6.2.1 Port B

For use as an interrupt controller, port B is configured as a bit port using Or-Priority Encoded Vector pattern recognition. The correspondence of vectored interrupts to port B's data lines follows the CIO's internal prioritization under Or-PEV mode. As data bit D7 is highest priority, it is tied to VI0. Data bit D0 is tied to VI7, D1 to VI6, etc. All port B data lines are set as non-inverting inputs with 1's catchers. Enabling of particular vectored interrupts is controlled by the pattern definition registers, i.e. the Pattern Polarity, Pattern Transition and Pattern Mask registers. For each interrupt line, the settings of its respective bit in these registers determine whether the line can generate an interrupt to the CPU and, if so, under what conditions. See Figure 5 in the Appendix called CIO Specification for available configurations.

As is, all the bits in the Pattern Polarity register have been set to 1 by the Monitor and those of the Pattern Transition register to 0. VI1 has been enabled by writing a 1 to D6 of the Pattern Mask register, while all the other vectored interrupts have been disabled by writing 0's to their respective bits. VI1 is assumed to be the parity line (PBERR) off the SBC-300 or Expandoram IV. This configuration of the pattern definition registers recognizes a high level on port B data bit D6 as an active interrupt (note that jumper W14-4 must be in place for the SBC-300's on board parity to be seen by the CIO).

Writing a 1 to its respective bit in the Pattern Mask register would enable interrupts for any other particular vectored interrupt line, with a high signal recognized as active. For example, the following Monitor commands would enable VI0:

```
    .I 7B      - force CIO to state 0 with read to 7B
    7B 40      - prints contents of currently addressed
                   register
    .O 7B 1    - set pointer register to address register 1

    .I 7B      - read CIO register 1
    7B 90      - prints contents of register 1
                   (ports B and C are enabled)
    .O 7B 1    - set pointer register to write register 1

    .O 7B 10   - write 10H to register 1 : disable port B by
                   resetting D7 in register 1 (MCC)
    .O 7B 2F   - set pointer to address register 2F

    .I 7B      - read register 2F
    7B 40      - prints contents of register 2F
                   (VI1 is already enabled)
    .O 7B 2F   - set to write register 2f

    .O 7B C0   - write C0 to register 1 : enable VI0 and VI1

    .O 7B 1    - set pointer register to address register 1

    .O 7B 90   - renable port B by setting D7 in MCC register
                   (and also leave port C enabled)
```

or coded:

```
    dovi0:                  ; assume CIO is already in state 0
            di              ; disable interrupts
            mvi  c,7bh      ; 7b is the CIO control port
            mvi  b,1        ; 1 is the master configuration
                            ; control
            outp b          ; set access up to register 1
            inp  a          ; read register 1
            ani  7fh        ; disable port B by resetting bit D7
            outp b          ; set write to register 1
            outp a          ; send the new info
            mvi  b,2fh      ; 2f is the pattern mask register
```

```
outp b       ; set pointer register to read it
inp  a       ; read it
ori  80h     ; set D7 to enable VI0
outp b       ; set up the write to 2f
outp a       ; write out the new info
mvi  b,1     ; now go back and reenable port B
outp b       ; set the access
inp  a       ; read register 1
ori  80h     ; reenable port B by setting bit D7
outp b       ; set the write access
outp a       ; write register 1
ei           ; reenable interrupts
```

When an interrupt is received by the CIO on a vectored interrupt line which has been enabled, the CIO will request an interrupt at the Z80. When the interrupt is acknowledged by the Z80, the CIO will imbed into bits D3-D1 of the port B interrupt vector the number of the highest priority port B data bit which exhibits a pattern match at the time of the interrupt acknowledge, and will provide this to the Z80 in response to the mode 2 interrupt acknowledge. In conjunction with the Z80 I register, the port B interrupt vector points to the service routine addresses for vectored interrupts at memory locations ff10-ff1f. The vectored interrupts are set up with routines which will print a message noting which vectored interrupt occurred, reset the CIO interrupt logic and the respective port B data line 1's catcher, display the state of the Z80 registers at the time of the interrupt acknowledge, and return the Monitor to command mode. VI1 has a special routine to handle parity. See Subsection 5.8 for more information on how VI1 is handled.

An interrupt can be handled differently by placing the address of the desired service routine into the appropriate word of the port B vector table at ff10. The interrupt service routine will have to reset the interrupt logic by code similar to that in the Resetting Vectored Interrupts section of the Appendix called Sample I/O Support Code.

5.6.2.2 Port C

Port C is enabled as a bit port with non-inverting data bits: bits D3, D2, D0 output and bit D1 input. D1 must be an input to avoid erratic signals on the parity error line. D3 must not go low or the SBC-300 PROM will be forced active with subsequent loss of control. D2 is held low during Monitor initialization and then set high to enable S-100 bus slaves. D1 is for use when port A is configured as a SASI port.

Note that because of D3, port C must always remain enabled. Thus, the CIO must never be reset via D0 in CIO register 0 (Master Interrupt Control), and port C must never be disabled via D4 in CIO register 1 (Master Configuration Control).

5.6.2.3 Port A

Though not done by the Monitor, port A can be configured in conjunction with the SBC-300 parallel I/O data port 70H as either a SASI port or a parallel printer port. See the Monitor Initialization Code section of the Appendix called Sample I/O Support Code.

5.6.2.4 Counter/Timers

The Monitor does not initialize the CIO counter/timers, but space is reserved at ff08-ff0f for a vector table to service the CIO counter/timer interrupts. See the Monitor Initialization Code section of the Appendix called Sample I/O Support Code.

5.7 ASCC INITIALIZATION

Version 1.2 of the SBC-300 Monitor has configured both ASCC serial ports for 9600 baud with 8 data bits, 1 stop bit, and DTR and RTS on. The x16 clock is used, with the baud rate generator enabled from the system PCLK and output to TX and RX clocks. See the Appendix called ASCC Specification for a summary of ASCC registers. Refer to the Zilog publication "Z8030/Z8530 SCC Serial Communications Controller Technical Manual" for more detailed information on programming the ASCC.

5.8 PARITY ERRORS

Version 1.2 of the SBC-300 Monitor handles parity errors through bit D6 of the data port of CIO port B. Under most circumstances, when a parity error is acknowledged, the parity error service routine prints the message "PARITY ERROR" on the console, resets the interrupt logic in the CIO and at the data port of CIO port B, displays the contents of the Z80 CPU registers at the time of the interrupt acknowledge, and returns the monitor to the command mode.

There are two conditions under which a parity error is handled differently: during execution of the Monitor "T" test and during execution of the "J" test. If a parity error is acknowledged at the time of the read memory instruction during the verify loop of the test, the message "PARITY ERROR" is displayed at the console followed by the memory address of the location being read at the time of the interrupt. If the error occurs during the "T" test, this memory address is the Z80 four digit hex address of the location being tested. If the error occurs during the "J" test, this memory address is the six digit hex address within the 16Mbyte memory space of the location being tested. If possible, execution of the test then resumes.

Note that removing jumper W14-4 on the SBC-300 will disconnect the SBC-300's on-board parity error line.

THIS PAGE INTENTIONALLY LEFT BLANK

SECTION VI
ENVIRONMENTAL CONSIDERATIONS

6.0 GENERAL

The maximum recommended ambient operating temperature is 50°C (122°F).

6.1 POWER REQUIREMENTS

	8V	+16V	-16V
4 mHz	1.9A	50mA	50mA
6 mHz	2.1A	50mA	50mA

6.2 PHYSICAL SPECIFICATIONS

The SBC-300 is contained on a multi-layer printed circuit board which conforms to the IEEE-696 specification. The board dimensions are 5.125" x 10.0". No components extend more than 0.50" from the component side of the board or more than 0.125" from the solder side of the board.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A
SELECTED IEEE-696 SPECIFICATION SHEETS

NOTE: For additional information, see the complete document "IEEE Standard 696 Interface Devices."

IEEE-696 Bus Pin List

Pin No.	Signal & Type	Active Level		Description
1	+8 V (B)			Instantaneous minimum greater than 7 V, instantaneous maximum less than 25 V, average maximum less than 11 V.
2	+16 V (B)			Instantaneous minimum greater than 14.5 V, instantaneous maximum less than 35 V, average maximum less than 21.5 V.
3	XRDY (S)	H		One of two ready inputs to the current bus master. The bus is ready when both these ready inputs are true. See pin 72.
4	VIO* (S)	L	OC	Vectored interrupt line 0.
5	VII* (S)	L	OC	Vectored interrupt line 1.
6	VI2* (S)	L	OC	Vectored interrupt line 2.
7	VI3* (S)	L	OC	Vectored interrupt line 3.
8	VI4* (S)	L	OC	Vectored interrupt line 4.
9	VI5* (S)	L	OC	Vectored interrupt line 5.
10	VI6* (S)	L	OC	Vectored interrupt line 6.
11	VI7* (S)	L	OC	Vectored interrupt line 7.
12	NMI* (S)	L	OC	Nonmaskable interrupt.
13	PWRFAIL* (B)	L		Power fail bus signal.
14	TMA3* (M)	L	OC	Temporary master priority bit 3.
15	A18 (M)	H		Extended address bit 18.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
16	A16 (M)	H	Extended address bit 16.
17	A17 (M)	H	Extended address bit 17.
18	SDSB* (M)	L OC	The symbol to disable the 8 status signals.
19	CDSB* (M)	L OC	The signal to disable the 5 control output signals.
20	0 V (B)		Common with pin 100.
21	NDEF		Not to be defined. Manufacturer must specify any use in detail.
22	ADSB* (M)	L OC	The signal to disable the address signals.
23	DODSB* (M)	L OC	The control signal to disable the data output signals. (DO7-0 for 8 bit transfers, ED7-0 and OD7-0 for 16 bit transfers.)
24	ϕ (B)	A	The master timing signal for the bus.
25	pSTVAL* (M)	L	Status valid strobe.
26	pHLDA (M)	H	A control signal used in conjunction with HOLD* to coordinate bus master transfer operations.
27	RFU		Reserved for future use.
28	RFU		Reserved for future use.
29	A5 (M)	H	Address bit 5.
30	A4 (M)	H	Address bit 4.
31	A3 (M)	H	Address bit 3.
32	A15 (M)	H	Address bit 15 (most significant for nonextended addressing).

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
33	A12 (M)	H	Address bit 12.
34	A9 (M)	H	Address bit 9.
35	DO1 (M)/ED1 (M/S)	H	Data out bit 1, bidirectional data bit 1.
36	DO0 (M)/ED0 (M/S)	H	Data out bit 0, bidirectional data bit 0.
37	A10 (M)	H	Address bit 10.
38	DO4 (M)/ED4 (M/S)	H	Data out bit 4, bidirectional data bit 4.
39	DO5 (M)/ED5 (M/S)	H	Data out bit 5, bidirectional data bit 5.
40	DO6 (M)/ED6 (M/S)	H	Data out bit 6, bidirectional data bit 6.
41	DI2 (S)/OD2 (M/S)	H	Data in bit 2, bidirectional data bit 2.
42	DI3 (S)/OD3 (M/S)	H	Data in bit 3, bidirectional data bit 3.
43	DI7 (S)/OD7 (M/S)	H	Data in bit 7, bidirectional data bit 7.
44	sM1 (M)	H	The status signal which indicates that the current cycle is an op-code fetch.
45	sOUT	H	The status signal identifying the data transfer bus cycle to an output device.
46	sINP (M)	H	The status signal identifying the data transfer bus cycle from an input device.
47	sMEMR (M)	H	The status signal identifying bus cycles which transfer data from memory to a bus master, which are not interrupt acknowledge instruction fetch cycle(s).

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level		Description
48	sHLTA (M)	H		The status signal which acknowledges that a HLT instruction has been executed.
49	CLOCK (B)	A		2 MHz (+0.5%) 40-60% duty cycle. Not required to be synchronous with any other bus signal.
50	0 V (B)			Common with pin 100.
51	+8 V (B)			Common with pin 1.
52	-16 V (B)			Instantaneous maximum less than -14.5 V, instantaneous minimum greater than -35 V, average minimum greater than -21.5 V.
53	0 V (B)			Common with pin 100.
54	SLAVE CLR* (B)	L	OC	A reset signal to reset bus slaves. Must be active with POC* and may also be generated by externl means.
55	TMAO* (M)	L	OC	Temporary master priority bit 0.
56	TMA1* (M)	L	OC	Temporary master priority bit 1.
57	DMA2* (M)	L	OC	Temporary master priority bit 2.
58	sXTRQ* (M)	L		The status signal which requests 16-bit slaves to assert SIXTN*.
59	A19 (M)	H		Extended address bit 19.
60	SIXTN*	L	OC	The signal generated by 16-bit slaves in response to the 16 bit request signal sXTRQ*.
61	A20 (M)	H		Extended address bit 20.
62	A21 (M)	H		Extended address bit 21.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level		Description
63	A22 (M)	H		Extended address bit 22.
64	A23 (M)	H		Extended address bit 23.
65	NDEF			Not to be defined signal.
66	NDEF			Not to be defined signal.
67	PHANTOM* (M/S)	L	OC	A bus signal which disables normal slave devices and enables phantom slaves--primarily used for bootstrapping systems without hardware front panels.
68	MWRT (B)	H		pWR*-sOUT (logic equation). This signal must follow pWR* by not more than 30 ns.
69	RFU			Reserved for future use.
70	0 V (B)			Common with pin 100.
71	RFU			Reserved for future use.
72	RDY (S)	H	OC	See comments for pin 3.
73	INT* (S)	L	OC	The primary interrupt request bus signal.
74	HOLD* (S)	L	OC	The control signal used in conjunction with pHLDA to coordinate bus master transfer operations.
75	RESET* (B)	L	OC	The reset signal to reset bus master devices. This signal must be active with POC* and may also be generated by external means.
76	pSYNC (M)	H		The control signal identifying BS_1.
77	pWR* (M)	L		The control signal signifying the presence of valid data on DO bus or data bus.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
78	pDBIN (M)	H	The control signal that requests data on the DI bus or data bus from the currently addressed slave.
79	A0 (M)	H	Address bit 0 (least significant).
80	A1 (M)	H	Address bit 1.
81	A2 (M)	H	Address bit 2.
82	A6 (M)	H	Address bit 6.
83	A7 (M)	H	Address bit 7.
84	A8 (M)	H	Address bit 8.
85	A13 (M)	H	Address bit 13.
86	A14 (M)	H	Address bit 14.
87	A11 (M)	H	Address bit 11.
88	DO2 (M)/ED2 (M/S)	H	Data out bit 2, bidirectional data bit 2.
89	DO3 (M)/ED3 (M/S)	H	Data out bit 3, bidirectional data bit 3.
90	DO7 (M)/ED7 (M/S)	H	Data out bit 7, bidirectional data bit 7.
91	DI4 (S)/OD4 (M/S)	H	Data in bit 4 and bidirectional data bit 12.
92	DI5 (S)/OD5 (M/S)	H	Data in bit 5 and bidirectional data bit 13.
93	DI6 (S)/OD6 (M/S)	H	Data in bit 6 and bidirectional data bit 14.
94	DI1 (S)/OD1 (M/S)	H	Data in bit 1 and bidirectional data bit 9.
95	DI0 (S)/OD0 (M/S)	H	Data in bit 0 (least significant for 8 bit data) and bidirectional data bit 8.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level		Description
96	sINTA (M)	H		The status signal identifying the bus input cycle(s) that may follow an accepted interrupt request presented on INT*.
97	sWO* (M)	L		The status signal identifying a bus cycle which transfers data from a bus master to a slave.
98	ERROR* (S)	L	OC	The bus status signal signifying an error condition during present bus cycle.
99	POC* (B)	L		The power-on clear signal for all bus devices; when this signal goes low, it must stay low for at least 10 microseconds.
100	0 V (B)			System ground.

IEEE-696 Bus Layout--Quick Reference

pin 1	+8 V (B)		pin 51	+8 V (B)	
pin 2	+16 V (B)		pin 52	-16 V (B)	
pin 3	XRDY (S)	H	pin 53	0 V	
pin 4	VI0* (S)	L	pin 54	SLAVE CLR* (B)	L
pin 5	VI1* (S)	L	pin 55	TMA0* (M)	L
pin 6	VI2* (S)	L	pin 56	TMA1* (M)	L
pin 7	VI3* (S)	L	pin 57	TMA2* (M)	L
pin 8	VI4* (S)	L	pin 58	sXTRQ* (M)	L
pin 9	VI5* (S)	L	pin 59	A19	H
pin 10	VI6* (S)	L	pin 60	SIXTN* (S)	L
pin 11	VI7* (S)	L	pin 61	A20 (M)	H
pin 12	NMI* (S)	L	pin 62	A21 (M)	H
pin 13	PWRFAIL* (B)	L	pin 63	A22 (M)	H
pin 14	TMA3* (M)	L	pin 64	A23 (M)	H
pin 15	A18 (M)	H	pin 65	NDEF	
pin 16	A16 (M)	H	pin 66	NDEF	
pin 17	A17 (M)	H	pin 67	PHANTOM* (M/S)	L
pin 18	SDSB* (M)	L	pin 68	MWRT (B)	H
pin 19	CDSB* (M)	L	pin 69	RFU	
pin 20	0 V		pin 70	0 V	
pin 21	NDEF		pin 71	RFU	
pin 22	ADSB* (M)	L	pin 72	RDY (S)	H
pin 23	DODSB* (M)	L	pin 73	INT* (S)	L
pin 24	ø (B)	H	pin 74	HOLD* (M)	L
pin 25	pSTVAL* (M)	L	pin 75	RESET* (B)	L
pin 26	pHLDA (M)	H	pin 76	pSYNC	H
pin 27	RFU		pin 77	pWR* (M)	L
pin 28	RFU		pin 78	pDBIN (M)	H
pin 29	A5 (M)	H	pin 79	A0 (M)	H
pin 30	A4 (M)	H	pin 80	A1 (M)	H
pin 31	A3 (M)	H	pin 81	A2 (M)	H
pin 32	A15 (M)	H	pin 82	A6 (M)	H
pin 33	A12 (M)	H	pin 83	A7 (M)	H
pin 34	A9 (M)	H	pin 84	A8 (M)	H
pin 35	DO1 (M)/ED1 (M/S)	H	pin 85	A13 (M)	H
pin 36	DO0 (M)/ED0 (M/S)	H	pin 86	A14 (M)	H
pin 37	A10 (M)	H	pin 87	A11 (M)	H
pin 38	DO4 (M)/ED4 (M/S)	H	pin 88	DO2 (M)/ED2 (M/S)	H
pin 39	DO5 (M)/ED5 (M/S)	H	pin 89	DO3 (M)/ED3 (M/S)	H
pin 40	DO6 (M)/ED6 (M/S)	H	pin 90	DO7 (M)/ED7 (M/S)	H
pin 41	DI2 (S)/OD2 (M/S)	H	pin 91	DI4 (S)/OD4 (M/S)	H
pin 42	DI3 (S)/OD3 (M/S)	H	pin 92	DI5 (S)/OD5 (M/S)	H
pin 43	DI7 (S)/OD7 (M/S)	H	pin 93	DI6 (S)/OD6 (M/S)	H
pin 44	sM1 (M)	H	pin 94	DI1 (S)/OD1 (M/S)	H
pin 45	sOUT (M)	H	pin 95	DI0 (S)/OD0 (M/S)	H
pin 46	sINP	H	pin 96	sINTA (M)	H
pin 47	sMEMR	H	pin 97	sWO* (M)	L
pin 48	sHLTA (M)	H	pin 98	ERROR* (S)	L
pin 49	CLOCK (B)		pin 99	POC* (B)	
pin 50	0V		pin 100	0 V	

APPENDIX B
HEXADECIMAL TO DECIMAL CONVERSION

HEXADECIMAL COLUMNS											
6		5		4		3		2		1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,152	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1,024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1,280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1,536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1,792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2,408	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2,304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2,560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2,816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3,072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3,328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3,584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3,840	F	240	F	15
0 1 2 3		4 5 6 7		0 1 2 3		4 5 6 7		0 1 2 3		4 5 6 7	
BYTE				BYTE				BYTE			

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C
STANDARD ASCII CODES

BIT POSITION

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

NUL	Null or all zeros	DC1	Device control 1
SOH	Start of heading	DC2	Device control 2
STX	Start of text	DC3	Device control 3
ETX	End of text	DC4	Device control 4
EOT	End of transmission	NAK	Negative acknowledge
ENQ	Enquiry	SYN	Synchronous idle
ACK	Acknowledge	ETB	End of transmission block
BEL	Bell, or alarm	CAN	Cancel
BS	Backspace	EM	End of medium
HT	Horizontal tabulation	SUB	Substitute
LF	Line feed	ESC	Escape
VT	Vertical tabulation	FS	File separator
FF	Form feed	GS	Group separator
CR	Carriage return	RS	Record separator
SO	Shift out	US	Unit separator
SI	Shift in	SP	Space
DLE	Data link escape	DEL	Delete

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D
EBCDIC TABLE

		Bit Positions 0, 1, 2, 3															
Bit Positions 4, 5, 6, 7		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	DLE	DS		SP	&	-						{	}	\	0
0001	1	SOH	DC1	SOS						a	j	~		A	J		1
0010	2	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	3	ETX	DC3							c	l	t		C	L	T	3
0100	4	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	5	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	6	LC	BS	EOB/ETB	UC					f	o	w		F	O	W	6
0111	7	DEL	IL	PRE/ESC	EOT					g	p	x		G	P	X	7
1000	8		CAN							h	q	y		H	Q	Y	8
1001	9	RLF	EM						\	i	r	z		I	R	Z	9
1010	A	SMM	CC	SM		¢	!	¦	:								
1011	B	VT				.	$	,	#								
1100	C	FF	IFS		DC4	<	*	%	@								
1101	D	CR	IGS	ENQ	NAK	(	)	_	'								
1110	E	SO	IRS	ACK		+	;	>	=								
1111	F	SI	IUS	BEL	SUB	|	¬	?	"								

▨ Duplicate Assignment

EBCDIC Character Assignments (Part 1 of 2)

Character	Hex
A	C1
B	C2
C	C3
D	C4
E	C5
F	C6
G	C7
H	C8
I	C9
J	D1
K	D2
L	D3
M	D4
N	D5
O	D6
P	D7
Q	D8
R	D9
S	E2
T	E3
U	E4
V	E5
W	E6
X	E7
Y	E8
Z	E9
a	81
b	82
c	83
d	84
e	85
f	86
g	87
h	88
i	89
j	91
k	92
l	93
m	94
n	95
o	96
p	97
q	98
r	99
s	A2
t	A3
u	A4
v	A5
w	A6
x	A7
y	A8
z	A9
0	F0
1	F1
2	F3
3	F3
4	F4
5	F5
6	F6
7	F7
8	F8
9	F9
&	50
-	60
/	61
$	5B

Character	Hex
¢	4A
!	5A
:	7A
#	7B
,	6B
.	4B
<	4C
*	5C
%	6C
@	7C
(	4D
)	5D
_	6D
'	7D
+	4E
;	5E
>	6E
=	7E
|	4F
¬	5F
?	6F
"	7F
{	C0
}	D0
\	E0
~	A1
`	79
¦	6A
BEL	2F
BS	16
BYP	24
CAN	18
CC	1A
CR	0D
DC1	11
DC2	12
DC3	13
DC4	3C
DEL	07
DLE	10
DS	20
EM	19
ENQ	2D
*EOB	26
EOT	37
*ESC	27
*ETB	26
ETX	03
FF	0C
FS	22
HT	05
IFS	1C
IGS	1D
IL	17
IRS	1E
IUS	1F
LC	06
LF	25
NAK	3D
NL	15
NUL	00
PF	04
PN	34
*PRE	27
RES	14
RLF	09

Character	Hex
RS	35
SI	0F
SM	2A
SMM	0A
SO	0E
SOH	01
SOS	21
Space	40
STX	02
SUB	3F
SYN	32
UC	36
VT	0B

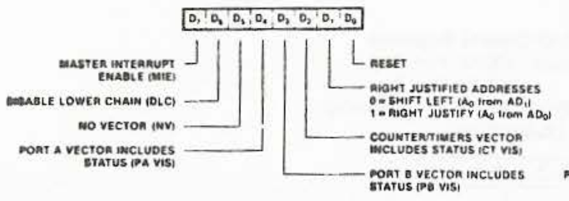
*ETB and EOB have the same hex assignment. PRE and ESC have the same hex assignment.

EBCDIC Character Assignments (Part 2 of 2)

APPENDIX E
CIO SPECIFICATION

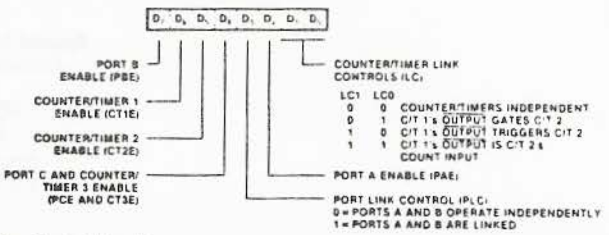
Registers

Master Interrupt Control Register
Address: 000000
(Read/Write)



D7 D6 D5 D4 D3 D2 D1 D0
MASTER INTERRUPT ENABLE (MIE)
DISABLE LOWER CHAIN (DLC)
NO VECTOR (NV)
PORT A VECTOR INCLUDES STATUS (PA VIS)
RESET
RIGHT JUSTIFIED ADDRESSES
0 = SHIFT LEFT (A0 from AD1)
1 = RIGHT JUSTIFY (A0 from AD0)
COUNTER/TIMERS VECTOR INCLUDES STATUS (CT VIS)
PORT B VECTOR INCLUDES STATUS (PB VIS)

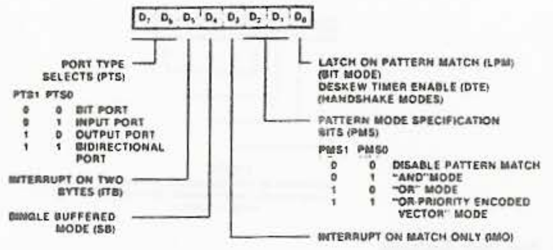
Master Configuration Control Register
Address: 000001
(Read/Write)



D7 D6 D5 D4 D3 D2 D1 D0
PORT B ENABLE (PBE)
COUNTER/TIMER 1 ENABLE (CT1E)
COUNTER/TIMER 2 ENABLE (CT2E)
PORT C AND COUNTER/TIMER 3 ENABLE (PCE AND CT3E)
COUNTER/TIMER LINK CONTROLS (LC)
LC1 LC0
0 0 COUNTER/TIMERS INDEPENDENT
0 1 C/T 1's OUTPUT GATES C/T 2
1 0 C/T 1's OUTPUT TRIGGERS C/T 2
1 1 C/T 1's OUTPUT IS C/T 2's COUNT INPUT
PORT A ENABLE (PAE)
PORT LINK CONTROL (PLC)
0 = PORTS A AND B OPERATE INDEPENDENTLY
1 = PORTS A AND B ARE LINKED

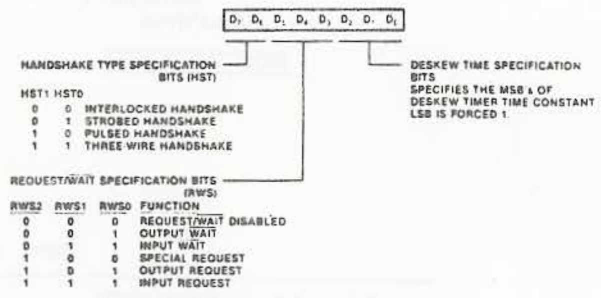
Figure 1. Master Control Registers

Port Mode Specification Registers
Addresses: 100000 Port A
101000 Port B
(Read/Write)

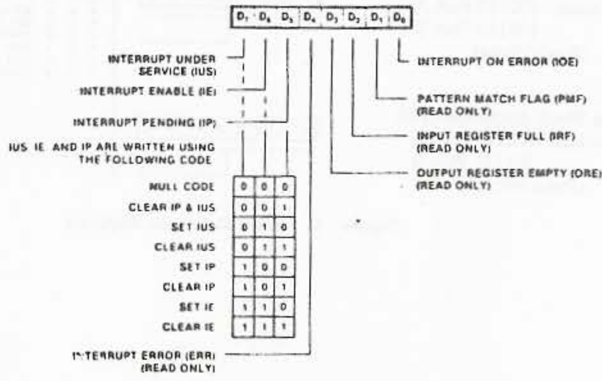


D7 D6 D5 D4 D3 D2 D1 D0
PORT TYPE SELECTS (PTS)
PTS1 PTS0
0 0 BIT PORT
0 1 INPUT PORT
1 0 OUTPUT PORT
1 1 BIDIRECTIONAL PORT
INTERRUPT ON TWO BYTES (ITB)
SINGLE BUFFERED MODE (SB)
LATCH ON PATTERN MATCH (LPM) (BIT MODE)
DESKEW TIMER ENABLE (DTE) (HANDSHAKE MODES)
PATTERN MODE SPECIFICATION BITS (PMS)
PMS1 PMS0
0 0 DISABLE PATTERN MATCH
0 1 "AND" MODE
1 0 "OR" MODE
1 1 "OR-PRIORITY ENCODED VECTOR" MODE
INTERRUPT ON MATCH ONLY (IMO)

Port Handshake Specification Registers
Addresses: 100001 Port A
101001 Port B
(Read/Write)



Port Command and Status Registers
Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)

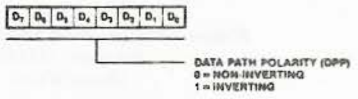


D7 D6 D5 D4 D3 D2 D1 D0
INTERRUPT UNDER SERVICE (IUS)
INTERRUPT ENABLE (IE)
INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING CODE
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
INTERRUPT ERROR (ERR) (READ ONLY)
INTERRUPT ON ERROR (IOE)
PATTERN MATCH FLAG (PMF) (READ ONLY)
INPUT REGISTER FULL (IRF) (READ ONLY)
OUTPUT REGISTER EMPTY (ORE) (READ ONLY)

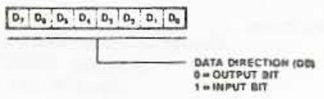
Figure 2. Port Specification Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

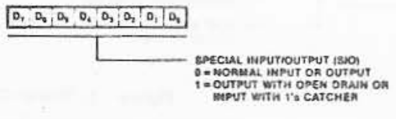
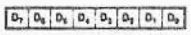

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
SPECIAL INPUT/OUTPUT (SIO)
0 = NORMAL INPUT OR OUTPUT
1 = OUTPUT WITH OPEN DRAIN OR INPUT WITH 1's CATCHER

Figure 3. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A
001110 Port B
(Read/Write)



Port C Data Register
Address: 001111
(Read/Write)

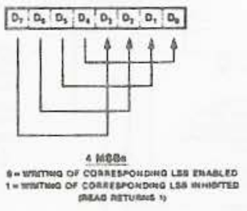
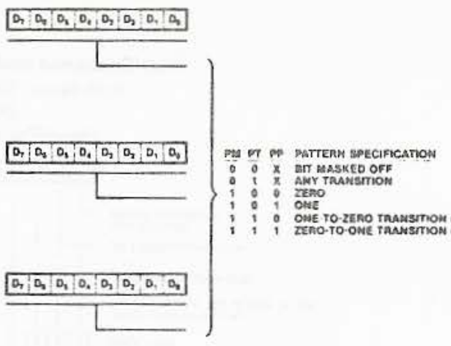

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
4 MSBs
0 = WRITING OF CORRESPONDING LSB ENABLED
1 = WRITING OF CORRESPONDING LSB INHIBITED (READ RETURNS 1)

Figure 4. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)

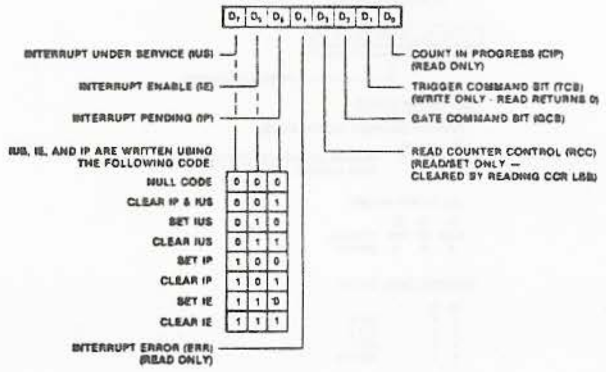

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE-TO-ZERO TRANSITION (\)
1	1	1	ZERO-TO-ONE TRANSITION (/)

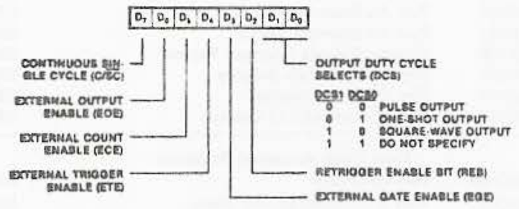
Figure 5. Pattern Definition Registers

Registers
(Continued)

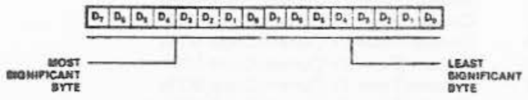
Counter/Timer Command and Status Registers
Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)


D7 D6 D5 D4 D3 D2 D1 D0
INTERRUPT UNDER SERVICE (IUS)
INTERRUPT ENABLE (IE)
INTERRUPT PENDING (IP)
IUS, IE, AND IP ARE WRITTEN USING THE FOLLOWING CODE:
NULL CODE 0 0 0
CLEAR IP & IUS 0 0 1
SET IUS 0 1 0
CLEAR IUS 0 1 1
SET IP 1 0 0
CLEAR IP 1 0 1
SET IE 1 1 0
CLEAR IE 1 1 1
INTERRUPT ERROR (ERR) (READ ONLY)
COUNT IN PROGRESS (CIP) (READ ONLY)
TRIGGER COMMAND BIT (TCB) (WRITE ONLY - READ RETURNS 0)
GATE COMMAND BIT (GCB)
READ COUNTER CONTROL (RCC) (READ/SET ONLY — CLEARED BY READING CCR LSB)

Counter/Timer Mode Specification Registers
Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)


D7 D6 D5 D4 D3 D2 D1 D0
CONTINUOUS SINGLE CYCLE (C/SC)
EXTERNAL OUTPUT ENABLE (EOE)
EXTERNAL COUNT ENABLE (ECE)
EXTERNAL TRIGGER ENABLE (ETE)
OUTPUT DUTY CYCLE SELECTS (DCS)
DCS1 DCS0
0 0 PULSE OUTPUT
0 1 ONE-SHOT OUTPUT
1 0 SQUARE-WAVE OUTPUT
1 1 DO NOT SPECIFY
RETRIGGER ENABLE BIT (REB)
EXTERNAL GATE ENABLE (EGE)

Counter/Timer Current Count Registers
Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers
Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

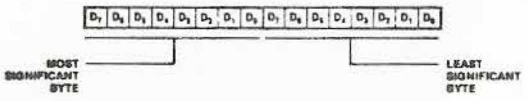
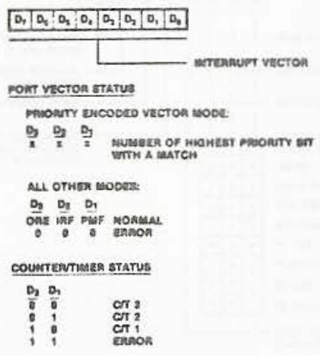


Figure 6. Counter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



Current Vector Register
Address: 011111
(Read Only)

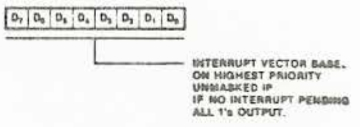


Figure 7. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address	Register Name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Most Often Accessed Registers

Address	Register Name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data
001110	Port B's Data
001111	Port C's Data

Counter/Timer Related Registers

Address	Register Name
010000	Counter/Timer 1's Current Count-MSBs
010001	Counter/Timer 1's Current Count-LSBs
010010	Counter/Timer 2's Current Count-MSBs
010011	Counter/Timer 2's Current Count-LSBs
010100	Counter/Timer 3's Current Count-MSBs
010101	Counter/Timer 3's Current Count-LSBs
010110	Counter/Timer 1's Time Constant-MSBs
010111	Counter/Timer 1's Time Constant-LSBs
011000	Counter/Timer 2's Time Constant-MSBs
011001	Counter/Timer 2's Time Constant-LSBs
011010	Counter/Timer 3's Time Constant-MSBs
011011	Counter/Timer 3's Time Constant-LSBs
011100	Counter/Timer 1's Mode Specification
011101	Counter/Timer 2's Mode Specification
011110	Counter/Timer 3's Mode Specification
011111	Current Vector

Port A Specification Registers

Address	Register Name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Port B Specification Registers

Address	Register Name
101000	Port B's Mode Specification
101001	Port B's Handshake Specification
101010	Port B's Data Path Polarity
101011	Port B's Data Direction
101100	Port B's Special I/O Control
101101	Port B's Pattern Polarity
101110	Port B's Pattern Transition
101111	Port B's Pattern Mask

APPENDIX F
ASCC SPECIFICATION

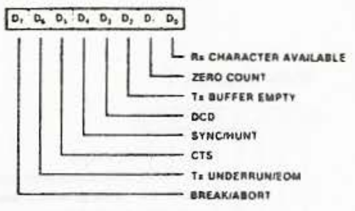
The Zilog 8531 ASCC chip contains 13 write registers in each channel that are programmed separately to configure each of the channels and 2 write registers shared by both channels.

Register addressing is direct only for the data registers (ports 7DH and 7FH). In all other cases programming the write registers requires 2 write operations, and reading the read registers requires 1 write and 1 read. The first write is to WR0 and contains 3 bits which point to the selected register. The second write is to the actual register selected or, if a read, the selected register is accessed and read. The pointer bits are automatically cleared after a read or write so that the pointer register (WR0 or RR0) is accessed again.

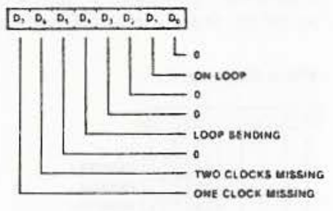
The ASCC contains 8 read registers which contain various status signals.

READ REGISTER FUNCTION		WRITE REGISTER FUNCTION	
RR0	Transmit/Receive buffer status, and External status	WR0	Command Register, (Register Pointers, Z8530 only), CRC initialization, resets for various modes
RR1	Special Receive Condition status, residue codes, error conditions	WR1	Interrupt conditions, Wait/DMA request control
RR2	Modified (Channel B only) interrupt vector and Unmodified interrupt vector (Channel A only)	WR2	Interrupt vector (access through either channel)
RR3	Interrupt Pending bits (Channel A only)	WR3	Receive/Control parameters, number of bits per character, Rx CRC enable
		WR4	Transmit/Receive miscellaneous parameters and modes, clock rate, number of sync characters, stop bits, parity
		WR5	Transmit parameters and controls, number of Tx bits per character, Tx CRC enable
		WR6	Sync character or SDLC address field (1st byte)
		WR7	Sync character or SDLC flag (2nd byte)
RR8	Receive buffer	WR8	Transmit buffer
		WR9	Master interrupt control and reset (accessed through either channel), reset bits, control interrupt daisy chain
RR10	Miscellaneous XMTR, RCVR status parameters	WR10	Miscellaneous transmitter/receiver control bits, NR2I, NR2, FM encoding, CRC reset
		WR11	Clock mode control, source of Rx and Tx clocks
RR12	Lower byte of baud rate generator time constant	WR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant	WR13	Upper byte of baud rate generator time constant
		WR14	Miscellaneous control bits: baud rate generator, Phase-Locked Loop control, auto echo, local loopback
RR15	External/Status interrupt control information	WR15	External/Status interrupt control information-control external conditions causing interrupts

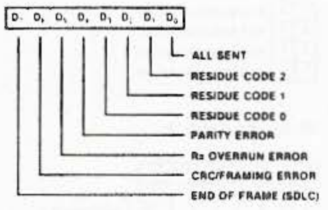
Read Register 0



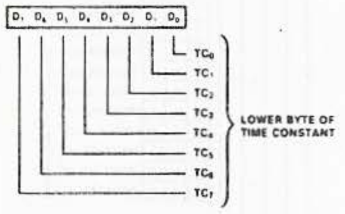
Read Register 10



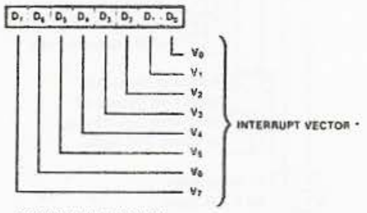
Read Register 1



Read Register 12

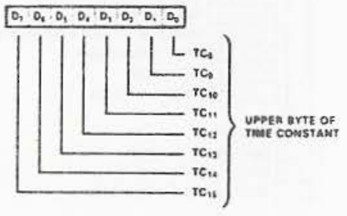


Read Register 2

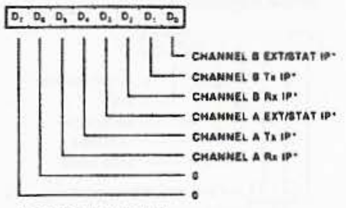


*MODIFIED IN B CHANNEL

Read Register 13

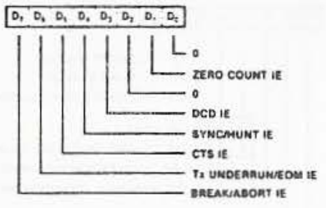


Read Register 3



*ALWAYS 0 IN B CHANNEL

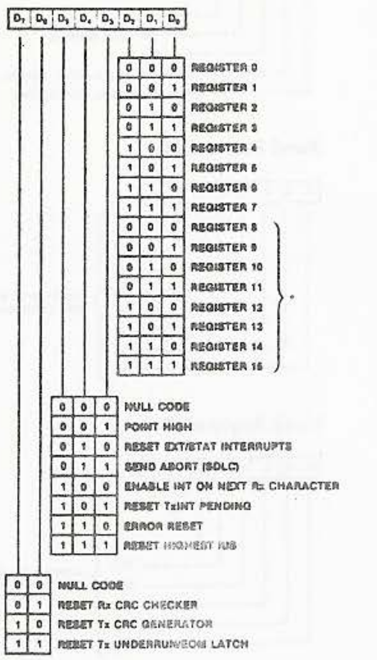
Read Register 15



Read Register Bit Functions

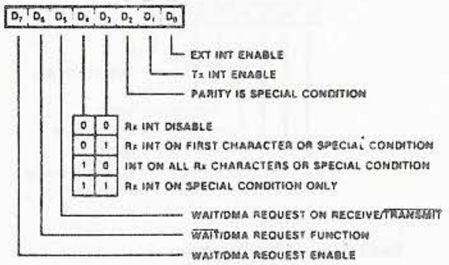
Write Registers. The SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits.

Write Register 0

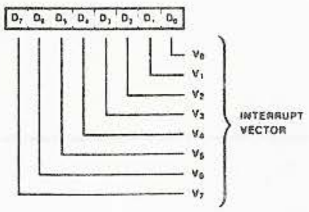


*WITH POINT HIGH COMMAND

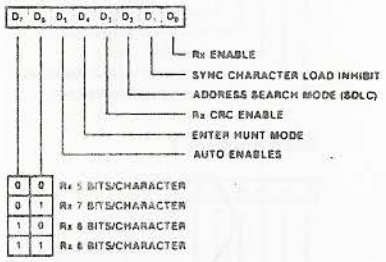
Write Register 1


D7 D6 D5 D4 D3 D2 D1 D0
EXT INT ENABLE
Tx INT ENABLE
PARITY IS SPECIAL CONDITION
0 0 Rx INT DISABLE
0 1 Rx INT ON FIRST CHARACTER OR SPECIAL CONDITION
1 0 INT ON ALL Rx CHARACTERS OR SPECIAL CONDITION
1 1 Rx INT ON SPECIAL CONDITION ONLY
WAIT/DMA REQUEST ON RECEIVE/TRANSMIT
WAIT/DMA REQUEST FUNCTION
WAIT/DMA REQUEST ENABLE

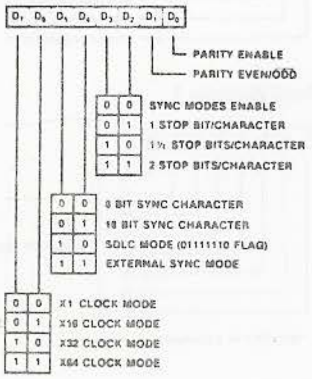
Write Register 2



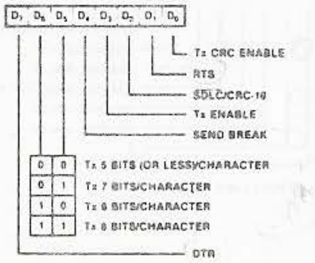
Write Register 3



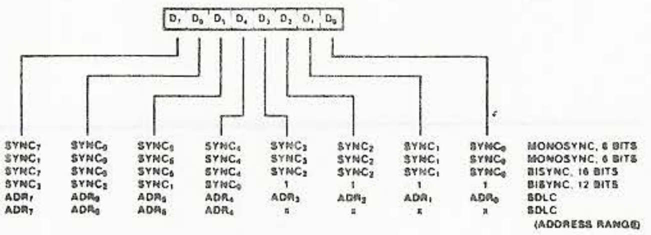
Write Register 4



Write Register 5

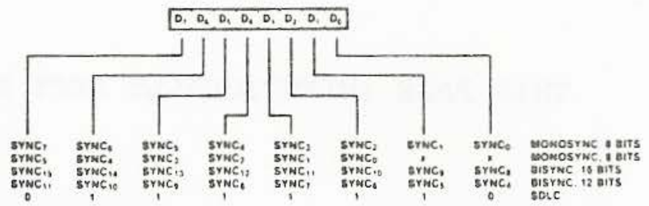


Write Register 6

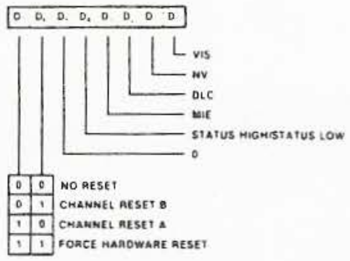

D7 D6 D5 D4 D3 D2 D1 D0
SYNC7 SYNC6 SYNC5 SYNC4 SYNC3 SYNC2 SYNC1 SYNC0 MONOSYNC, 8 BITS
SYNC1 SYNC0 SYNC5 SYNC4 SYNC3 SYNC2 SYNC1 SYNC0 MONOSYNC, 6 BITS
SYNC7 SYNC6 SYNC5 SYNC4 SYNC3 SYNC2 SYNC1 SYNC0 BISYNC, 16 BITS
SYNC3 SYNC2 SYNC1 SYNC0 1 1 1 1 BISYNC, 12 BITS
ADR7 ADR6 ADR5 ADR4 ADR3 ADR2 ADR1 ADR0 SDLC
ADR7 ADR6 ADR5 ADR4 x x x x SDLC (ADDRESS RANGE)

Write Register Bit Functions

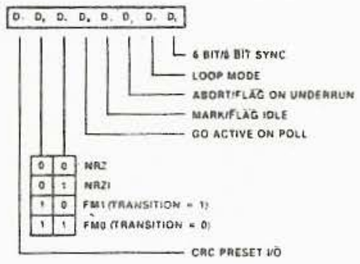
Write Register 7


D7 D6 D5 D4 D3 D2 D1 D0
SYNC7 SYNC1 SYNC15 SYNC11 0
SYNC6 SYNC0 SYNC14 SYNC10 1
SYNC5 SYNC5 SYNC13 SYNC9 1
SYNC4 SYNC7 SYNC12 SYNC8 1
SYNC3 SYNC1 SYNC11 SYNC7 1
SYNC2 SYNC0 SYNC10 SYNC6 1
SYNC1 x SYNC9 SYNC5 1
SYNC0 x SYNC8 SYNC4 0
MONOSYNC 8 BITS
MONOSYNC, 6 BITS
BISYNC 16 BITS
BISYNC, 12 BITS
SDLC

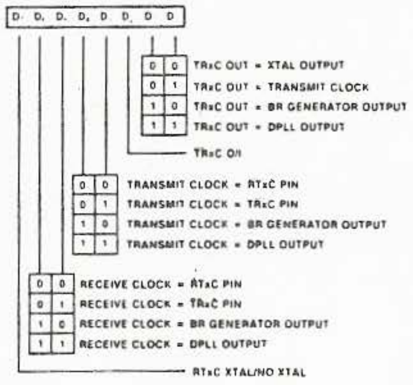
Write Register 9



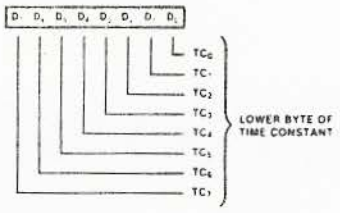
Write Register 10



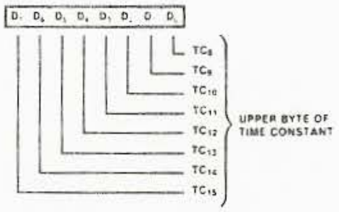
Write Register 11


D7 D6 D5 D4 D3 D2 D1 D0
0 0 TRxC OUT = XTAL OUTPUT
0 1 TRxC OUT = TRANSMIT CLOCK
1 0 TRxC OUT = BR GENERATOR OUTPUT
1 1 TRxC OUT = DPLL OUTPUT
TRxC O/I
0 0 TRANSMIT CLOCK = RTxC PIN
0 1 TRANSMIT CLOCK = TRxC PIN
1 0 TRANSMIT CLOCK = BR GENERATOR OUTPUT
1 1 TRANSMIT CLOCK = DPLL OUTPUT
0 0 RECEIVE CLOCK = RTxC PIN
0 1 RECEIVE CLOCK = TRxC PIN
1 0 RECEIVE CLOCK = BR GENERATOR OUTPUT
1 1 RECEIVE CLOCK = DPLL OUTPUT
RTxC XTAL/NO XTAL

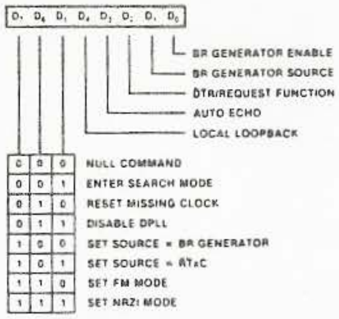
Write Register 12



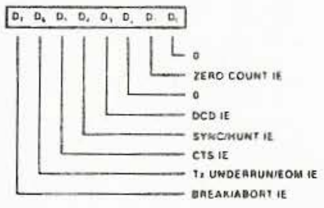
Write Register 13



Write Register 14



Write Register 15



Write Register Bit Functions (Continued)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G
PAL16L8 SPECIFICATION

U28 (DECODE):

```
/MREQ /IORQ /RD /WR BUSAK IEO BOOT /RFSH MASTER GND
/INTA /GORAM PA16 /IO /BRFSH /MATCH /RAMENB /PROMENB /RDOFF VCC

IF(VCC)/RAMENB = /BOOT*MASTER*MATCH*PA16+/BOOT*/MASTER*PA16+
          /BOOT*/MASTER*BUSAK*MATCH

IF(VCC)/PROMENB = BOOT+MASTER*/BUSAK*MATCH*/PA16+
           /MASTER*/BUSAK*/PA16

IF(VCC)/RDOFF    = /BOOT*MASTER*MREQ*/MATCH*RD+
           /BOOT*MASTER*IORQ*RD*/IO
          +/BOOT*IEO*INTA*IORQ*MASTER

IF(VCC)/GORAM    = RAMENB*MREQ*RD+RAMENB*MREQ*WR+
           RFSH*MREQ+BRFSH
```

U59 (SLAVE):

```
/ADSB /HOLD MATCH /DODSB /CEN MEMR SLVRQ DBIN BUSAK GND
MEMW /BUSRQ /RDOFF MASTER /BUSENB /DIEN /DOEN /AEN /BRFSH VCC

IF(VCC)/BRFSH = MASTER*/MATCH*MEMR*DBIN*BUSAK

IF(VCC)/BUSENB = /CEN*MASTER+MATCH*/MASTER*AEN

IF(VCC)/BUSRQ = MASTER*HOLD+MASTER*BUSAK*ADSB+/MASTER*SLVRQ

IF(VCC)/AEN = MASTER*/ADSB+BUSAK

IF(VCC)/DOEN = MASTER*/DODSB*/BUSAK+BUSAK*MATCH*MEMW

IF(VCC)/DIEN = MASTER*RDOFF*/BUSAK+BUSAK*MATCH*MEMR*DBIN
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H
PORT ASSIGNMENTS

Port (Hex)	Signal Name	Function
70	SASI*	SASI/Parallel port--data
71	MAPPER*	Memory mapper
72	ATTEN*	Attention bit for slave processor (used to generate an interrupt to master)
73-77	Not Connected	Not used
78-7B	CIOEN*	CIO (8536) enable
78	Port C	
79	Port B	
7A	Port A	
7B	Control	
7C-7F	SCCEN	Serial chip enable
	Channel	**Function**
7C	B	Control
7D	B	Data
7E	A	Control
7F	A	Data

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I
PARALLEL INTERFACES: SASI/CENTRONICS

SINGLE-ENDED SASI/SCSI COMPATIBLE INTERFACE CONNECTOR J4

NOTES:

All odd pins shall be connected to ground
True on SASI = Signal Assertion = 0.0 VDC to 0.5 VDC
False on SASI = 2.0 VDC to 5.25 VDC
Parity not used
To enable the data port for output (port 70), port 7A bit 5 must be high

Signal	J4 Pin Number	Port Number	Bit Number
DB0	2	70	0
DB1	4	70	1
DB2	6	70	2
DB3	8	70	3
DB4	10	70	4
DB5	12	70	5
DB6	14	70	6
DB7	16	70	7
BSY	36	7A	1(1)
ACK	38	(2)	
RST	40	7A	0(3)
MSG	42	7A	2
SEL	44	7A	7
C/D*	46	7A	3
REQ	48	7A	4
I/O*	50	7A	5

(1) Port 7A bit 6 is an enable for this line and must be high
(2) An I/O instruction to port 70 pulses this line
(3) Used in conjunction with port 78 bit 0 (see schematic sheet 4)

SASI TO CENTRONICS INTERCONNECT

NOTE: This table shows the pinouts for connecting a Centronics parallel port to the SBC-300 SASI port (J4). The pinouts are from J4 (50 pin) to a 25 pin 'D' connector. A second cable contains a mating 25 pin 'D' connector and the 34 pin Centronics connector. This cable is designed to be a 26 conductor flat ribbon to minimize the construction.

SASI Signal	J4 Pin Number	25 Pin 'D' Connector	Centronics (34 Pin) Signal	Pin
DB0	2	2	DATA 1	2
DB1	4	3	DATA 2	3
DB2	6	4	DATA 3	4
DB3	8	5	DATA 4	5
DB4	10<-------->	6<----------->	DATA 5	6
DB5	12	7	DATA 6	7
DB6	14	8	DATA 7	8
DB7	16	9	DATA 8	9
MSG	42	11	BUSY	11
SEL	44	1	STROBE*	1
GROUND	Any Odd Pin	14	GROUND	19

APPENDIX J
MONITOR LOCATIONS

THIS PAGE INTENTIONALLY LEFT BLANK

```
               ;-----------------------------------------------------------------
                      title   'SBC-300 Version of Monitor v2.0-VFW 9/30/83'
                ;-----------------------------------------------------------------
                ;
                ;###################################################################
                ;                                                               #
                ; 1/26/83   initial version                        dan hammond  #
                ; 5/30/83   initial release                        efj   v1.0   #
                ; 6/15/83   fix sp, misc optimization              efj   v1.1   #
                ; 7/13/83   fix parity, vi's, step, & cio port c   efj   v1.2   #
                ;                                                               #
                ; 8/03/83   first version for vfw board            efj   v0.0   #
                ; 8/19/83   added modes for format routines        dkb   v0.1   #
                ; 9/09/83   fix for double density disks boot      rlb   v0.3   #
                ; 9/11/83   more fixes for dual-density booting    dkb   v0.4   #
                ; 9/21/83   added mini-boot, winnie boot           wbd   v0.5   #
                ; 9/24/83     added vf2 disk routines              wbd          v0.6   #
                ;                                                               #
                ; 9/30/83   Fixed boot                             efj          #
                ; 9/30/83     Released as 2.0                            efj    v2.0   #
                ;                                                               #
                ;  ---- don't forget to update title, signon and version date  ---- #
                ;###################################################################
                ;
                ;
                ;
                ;
                ;###########################################################
                ;    runa:                                                 #
                ;            present restrictions are:                    #
                ;            runa must be a page boundary                 #
                ;            page 0ffh is reserved for tables, parms      #
                ;###########################################################
                ;
 DF00 =         runa  equ   0df00h      ; address to org monitor
                ;
                ;
```

```
                 ;#################################################################
                 ;                                                                #
                 ;                              system equates                    #
                 ;                                                                #
                 ;#################################################################
                 ;
                 ;
                 ;##############################################
                 ;                                             #
                 ;       miscellaneous equates                 #
                 ;                                             #
                 ;##############################################
                 ;
                 ;
0000 =           tor    equ      0000h          ; top of ram
0000 =           rbase equ       0000h          ; start of ram
DF00 =           usrsp equ       runa           ; default user stack below monitor
0080 =           sctrsz          equ     128    ; bytes per sector
                 ;
                 ;
8000 =           vdb$conf        equ     1000$0000$0000$0000b
4000 =           ser$conf        equ     0100$0000$0000$0000b
                 ;
                 ;
0040 =           active$vctrs  equ       0100$0000b
                 ;                                  bit mask for which vectored interrputs
                 ;                                  are enabled. ( vi0 -> b7, vi1 -> b6, etc. )
                 ;                                  only pberr (parity) enabled
                 ;
                       if mhz4
0000 =           mhz$conf        equ     0000$0000$0000$0000b
                                                ; bits 1,0 of conf$vctr denote cpu clock speed
8000 =           initdlay        equ     08000h ; delay for system to stabilize after cio init
0400 =           codlay                  equ    400h    ; delay for init console service
                       else
                        if mhz6
                 mhz$conf        equ     0000$0000$0000$0001b
                                                ; (bit 1 reserved for 8 mhz)
                 initdlay        equ     0c000h
                 codlay                  equ    600h
                        endif
                       endif
                 ;
                 ;
4000 =           conf$init       equ     ser$conf or mhz$conf
                                                ; initial vector is ser con and whatever clock.
                 ;
                 ;
```

```
               ;*********************************************
               ;                                            *
               ;       page 000h equates                    *
               ;                                            *
               ;*********************************************
               ;
0080 =         cld$stk      equ     rbase+80h      ; cold boot stack
0080 =         cold  equ    rbase+80h      ; cold start address
               ;
               ;
               ;*********************************************
               ;                                            *
               ;       page 0ffh equates                    *
               ;                                            *
               ;*********************************************
               ;
               ;
FF00 =         syspag       equ     0ff00h          ; page 0ffh reserved for monitor and os's
               ;
FF00 =         sys$info     equ     syspag + 00h    ; 8 bytes for system information
FF00 =         mon$adr              equ     sys$info + 0
FF02 =         conf$vctr    equ     sys$info + 2
FF02 =         conf$lo              equ     conf$vctr + 0
FF03 =         conf$hi              equ     conf$vctr + 1
FF04 =         mon$ver              equ     sys$info + 4
               ;                              bytes sysinfo(5 thru 7) are rfu
               ;
FF00 =         ivt   equ    syspag          ; base page ram location of interrupt vectors
FF08 =         ivtcioc      equ     ivt + 08h       ; offset to cio counter/timers isrv's   - rfu
FF10 =         ivtciob      equ     ivt + 10h       ; offset to cio port b isrv's
FF20 =         ivtcioa      equ     ivt + 20h       ; offset to cio port a isrv's           - rfu
FF30 =         ivtscc       equ     ivt + 30h       ; offset to scc isrv's                  - rfu
               ;
FF10 =         visr7 equ    ivtciob + 00h
FF12 =         visr6 equ    ivtciob + 02h
FF14 =         visr5 equ    ivtciob + 04h
FF16 =         visr4 equ    ivtciob + 06h
FF18 =         visr3 equ    ivtciob + 08h
FF1A =         visr2 equ    ivtciob + 0ah
FF1C =         visr1 equ    ivtciob + 0ch
FF1E =         visr0 equ    ivtciob + 0eh
               ;
               ;
FF40 =         var$spc      equ     syspag + 40h    ; space for variables
               ;
               ;     - twelve bytes -         - next used by disk routines, loaders
FF40 =         taddr equ    var$spc+0       ; transfer address
FF42 =         unit  equ    taddr  +2       ; new unit byte
FF44 =         trk   equ    unit   +2       ; track
FF46 =         sctr  equ    trk    +2       ; sector
FF48 =         nrec  equ    sctr   +2       ; # of sectors
FF4A =         nohds equ    nrec   +2       ; for winch                             - rfu
               ;
               ;
FF4C =         temps equ    nohds  +2       ; rest of varibles are short terms
```

```
                ;
                ;      - eight bytes -          - next four used by ycmd
FF4C =          scr1  equ     temps  +0
FF4E =          scr3  equ     scr1   +2
FF50 =          first equ     scr3   +2
FF52 =          last  equ     first  +2
                ;
                ;                               - miscellaneous disk temporaries
                ;      - seventeen bytes -        they can share with ycmd vars
                ;               only erstat used by VFWIII
FF4C =          ermask        equ     temps  +0       ; error mask
FF4D =          erstat        equ     ermask +1       ; error status
FF4E =          idsv  equ     erstat +1       ; 4 bytes
FF4F =          side  equ     idsv   +1       ; id side byte
FF52 =          cmdsv equ     idsv   +4       ; command save
FF53 =          spsv  equ     cmdsv  +1       ; sp save (2 bytes)
FF55 =          temp1 equ     spsv   +2       ; 1 byte temporary reg
FF56 =          temp2 equ     temp1  +1       ; 2 byte temporary reg
FF58 =          ixsav equ     temp2  +2       ; ix store buffer
FF5A =          unitck        equ     ixsav  +2       ; old unit byte
FF5B =          rseek equ     unitck +1       ; nbr of reseeks
FF5C =          rtry  equ     rseek  +1       ; nbr of rtrys
                ;-------------------------------------------------------------
                ;
                ;      - 2 bytes -              - next two used for parity err, interrupts
FF5D =          ivn   equ     rtry   +1       ; location for vectored interrupt flag
FF5E =          par$loc       equ     ivn    +1       ; flag to note if parity err location is known
                ;
FF5F =          nxt$loc       equ     par$loc+1       ; next available location
                ;                                 but watch out for intrrupt stack
                ;
FF7E =          sav$isp       equ     0ff7eh          ; save sp during interrupt handling
FF7E =          int$stk       equ     sav$isp         ; stack for interrupts
                ;
                ;
FFC0 =          spad  equ     tor-40h
                ;
FFC0 =          mossp1        equ     spad            ; op sys stack pointer
                ;
FFC0 =          sbpk  equ     spad            ; software bp flag
FFC1 =          bpcod equ     spad+1          ; software bp user code
FFC4 =          bploc equ     spad+4          ; software bp location
FFC6 =          long  equ     spad+6          ; print format flag
FFC7 =          opcnt equ     spad+7
FFC8 =          mxchr equ     spad+8
FFC9 =          count equ     spad+9
000A =          flag  equ     10              ; dcmd flag spad+10
FFD0 =          oprs  equ     spad+10h
FFD0 =          opr1  equ     oprs
FFD2 =          opr2  equ     oprs+2
FFD4 =          opr3  equ     oprs+4
FFD6 =          opr4  equ     oprs+6
FFD8 =          opr5  equ     oprs+8
FFDA =          opr6  equ     oprs+10
FFDC =          opr7  equ     oprs+12
FFDE =          opr8  equ     oprs+14
```

```
FFE0 =          opr9  equ       oprs+16
FFE2 =          opr10 equ       oprs+18
                ;
0000 =          mossp equ       tor                     ; op sys sp
FFFE =          upc   equ       mossp-2                 ; user register save area
FFFC =          uaf   equ       mossp-4
FFFA =          uir   equ       mossp-6
FFF8 =          ubc   equ       mossp-8
FFF6 =          ude   equ       mossp-10
FFF4 =          uhl   equ       mossp-12
FFF2 =          uafp  equ       mossp-14
FFF0 =          ubcp  equ       mossp-16
FFEE =          udep  equ       mossp-18
FFEC =          uhlp  equ       mossp-20
FFEA =          uix   equ       mossp-22
FFE8 =          uiy   equ       mossp-24
FFE6 =          usp   equ       mossp-26
FFE4 =          tusp  equ       mossp-28
                ;
FFFD =          ua    equ       uaf+1
FFFC =          uf    equ       uaf
FFFB =          ui    equ       uir+1
FFFA =          uif   equ       uir
FFF9 =          ub    equ       ubc+1
FFF8 =          uc    equ       ubc
FFF7 =          ud    equ       ude+1
FFF6 =          ue    equ       ude
FFF5 =          uh    equ       uhl+1
FFF4 =          ul    equ       uhl
FFF3 =          uap   equ       uafp+1
FFF2 =          ufp   equ       uafp
FFF1 =          ubp   equ       ubcp+1
FFF0 =          ucp   equ       ubcp
FFEF =          udp   equ       udep+1
FFEE =          uep   equ       udep
FFED =          uhp   equ       uhlp+1
FFEC =          ulp   equ       uhlp
                ;
                ;
                ;##################################################
                ;                                                 #
                ;       i/o port definitions, commands, & masks   #
                ;                                                 #
                ;##################################################
                ;
0001 =          vdbd  equ       1                       ; vdb data port
0000 =          vdbs  equ       0                       ; vdb status port
0070 =          sasid equ       70h                     ; sasi data port
0071 =          mapper          equ     71h             ; memory mapper port
0072 =          atten equ       72h                     ; host attention port
0078 =          cioc  equ       78h                     ; cio data port c
0079 =          ciob  equ       79h                     ; cio data port b
007A =          cioa  equ       7ah                     ; cio data port a
007B =          cioctl          equ     7bh             ; cio control port
007C =          sccbc equ       7ch                     ; scc channel b control
007D =          sccbd equ       7dh                     ; scc channel b data
```

```
007E =          sccac equ       7eh                     ; scc channel a control
007F =          sccad equ       7fh                     ; scc channel a data
                ;
0002 =          vrxmsk          equ     2               ; vdb rx mask
0004 =          vtxmsk          equ     4               ; vdb tx mask
0001 =          srxmsk          equ     1               ; scc rx mask
0004 =          stxmsk          equ     4               ; scc tx mask
                ;
0009 =          csr$b           equ     09h     ; cmd/status cio port b
0020 =          rst$csr$b       equ     0010$0000b
00A0 =          ius$ip$mask     equ     1010$0000b
                ;
                ;
                ;               VERSAFLOPPY II
                ;       ports used by disk controller
                ;
0060 =          dports          equ     60h
0060 =          rset  equ       dports+0        ; controller reset addr
0063 =          select          equ     dports+3        ; drive select port
0064 =          cmd   equ       dports+4        ; command port
0064 =          status2         equ     dports+4        ; status port
0065 =          track equ       dports+5        ; track port
0066 =          sector2         equ     dports+6        ; sector port
0067 =          data  equ       dports+7        ; data port
                ;
                ;
                ;       disk controller command codes
                ;
00C0 =          rdacmd          equ     0c0h            ; read address cmd
0088 =          rd2cmd          equ     88h             ; read sector cmd
00A8 =          wr2cmd          equ     0a8h            ; write sector cmd
00F4 =          wrtcmd          equ     0f4h            ; write track cmd
                ;
                ;
                ;       disk error status bits (erstat)
                ;
                ;       bit 7 - drive not ready
                ;       bit 6 - write protected
                ;       bit 5 - track seek error
                ;       bit 4 - sector not found
                ;       bit 3 - crc error
                ;       bit 2 - data lost
                ;       bit 1 - drq bit
                ;       bit 0 - write deleted sector read
                ;       feh - controller hang up
                ;       0fh - invalid track error
                ;
                ;
                ;               VERSAFLOPPY WINCHESTER III
                ;
                ;
0050 =          vfwbas          equ     50h             ; base of vfw ports
0050 =          buffer          equ     vfwbas+0        ; sector buffer
0051 =          flags equ       vfwbas+1        ; error flags
0051 =          wpcyl equ       vfwbas+1        ; wrt prec cyl
0052 =          scount          equ     vfwbas+2        ; sector count
```

```
0053 =          sector          equ     vfwbas+3        ; sector number
0054 =          cyll  equ       vfwbas+4        ; cylinder (lower byte)
0055 =          cylh  equ       vfwbas+5        ; cylinder (upper byte)
0056 =          sdh   equ       vfwbas+6        ; size/drive/head select
0057 =          status          equ     vfwbas+7        ; vfw status
0057 =          command         equ     vfwbas+7        ; vfw command port
0058 =          dmal  equ       vfwbas+8        ; dma address port (low)
0059 =          dmam  equ       vfwbas+9        ; dma address port (mid)
005A =          dmah  equ       vfwbas+10       ; dma address port (high)
005B =          control         equ     vfwbas+11       ; dma/floppy control
                ;
                ;
                ;       control port bit masks
                ;
0001 =          bigf  equ       1               ; 8" floppy
0002 =          hd3   equ       2               ; bit3 of winch head select
0004 =          sden  equ       4               ; single density
0008 =          mmode equ       8               ; motor mode : 1= 1 sec delay for motor start
0010 =          dread equ       10h             ; disk read
0020 =          dmaenb          equ     20h             ; dma enable
0040 =          pce   equ       40h             ; prom chip select enable - should be 0
0080 =          bank  equ       80h             ; select upper half of prom : n.u.
                ;
                ;
                ;       status port bit masks
                ;
0001 =          err   equ       1               ; error bit
0002 =          irq   equ       2               ; interrupt rqst bit
0004 =          cor   equ       4               ; corrected read bit
0008 =          drq   equ       8               ; data request bit
0010 =          scb   equ       10h             ; seek complete bit
0020 =          wf    equ       20h             ; write fault/prot bit
0040 =          rdy   equ       40h             ; drive ready bit
0080 =          bsy   equ       80h             ; vfw busy bit
                ;
                ;
                ;       vfw commands
                ;
0010 =          restore         equ     10h             ; heads to trk00
                ;
                        if dma
                ;
                rdcmd equ       28h             ; read sector
                ;
                        else    ; not dma
                ;
0020 =          rdcmd equ       20h             ; read sector
                ;
                        endif   ; not dma
                ;
0030 =          wrcmd equ       30h             ; write sector
0050 =          fmcmd equ       50h             ; format command
0051 =          sdfmcmd         equ     51h             ; format command for single density
0070 =          seekcmd         equ     70h             ; seek command
                ;
                        if dma
```

```
;
ram300		equ	01		; high byte, onboard ram address 010000
;
	endif	; dma
;
;
```

```
                  ;*************************************************
                  ;                                               *
                  ;       begin executable code                   *
                  ;                                               *
                  ;*************************************************
                  ;
                  ;
DF00                      org     runa
                  ;
                  ;
                  ;               all code must be page zero executable until init2  entry
                  ;       ------------------------------------------------------
                  ;
                  ;*********************************************
                  ;                                           *
                  ;       basic linkages into monitor         *
                  ;       ( cp/m compatable )                 *
                  ;                                           *
                  ;*********************************************
                  ;
                  mon$strt:
                          jmpr    init            ; 0 monitor cold boot
DF00+1862
                  ;
DF02 20           ver$byt:        db      ver$info        ; version number byte
                  ;
DF03 C380E1       m$strt:         jmp     rentry          ; 1 monitor re-entry
DF06 C32CE9       cse:    jmp     const           ; 2 console status
DF09 C322E9       cie:    jmp     conin           ; 3 read console
DF0C C327E9       coe:    jmp     conout          ; 4 write console
DF0F C327E9               jmp     conout          ; 5 list
DF12 C327E9               jmp     conout          ; 6 pun
DF15 C322E9               jmp     conin           ; 7
DF18 C37EEE       h2:     jmp     home            ; 8
DF1B C35AEC               jmp     seldsk          ; 9
DF1E C35FEC               jmp     settrk          ; 10
DF21 C364EC               jmp     setsec          ; 11
DF24 C369EC               jmp     setdma          ; 12
DF27 C392EE       r2:     jmp     read            ; 13
DF2A C385EE       w2:     jmp     write           ; 14
                  ;
DF2D C36EEC       l2:     jmp     loader          ; 15
DF30 C37DEC       s2:     jmp     saver           ; 16
DF33 C34EEB               jmp     ret$conf        ; 17
                  ;
                  ;
                  ;*********************************************
                  ;                                           *
                  ;       signon and version/date             *
                  ;                                           *
                  ;*********************************************
                  ;
                  ;
0020 =            ver$info        equ     20h
                                          ; monitor version 2.0
```

```
                    signon$msg:
DF36 0A0D534453         db      0ah,0dh,'SDSYSTEMS SBC-300 MONITOR  V2.0',0ah,0dh,00H
                    ;
DF5A 3039333038         db      '093083'
                    ;
                     IF MHZ4
                    ;
DF60 2D344D2D           db      '-4M-'
                    ;
                     ELSE       ; MHZ6
                    ;
                        db      '-6M-'
                    ;
                     ENDIF      ; MHZ6
                    ;
                    ;
```

```
                ;################################################
                ;                                               #
                ;       enter here after reset                  #
                ;                                               #
                ;################################################
                ;
                init:
DF64 F3                 di                      ; if not by reset
DF65 31C0FF             lxi     sp,mossp1       ; initialize stack pointer
                                                ; initialize mem mapper
DF68 216D01             lxi     h,map$init - runa
                        lxix    map$ret0
DF6B+DD21
DF6D+7100
                        jmpr    init$mpr
DF6F+186D
                ;
0071 =          map$ret0        equ     $ - runa
DF71 DB7B               in      cioctl          ; reset cio
DF73 AF                 xra     a
DF74 D37B               out     cioctl          ; wrt ptr or clr rst
DF76 DB7B               in      cioctl          ; state 0
                                                ; now do basic cio init
DF78 21C001             lxi     h,ciotab - runa
DF7B 0617               mvi     b,cionb         ; # of bytes
DF7D 0E7B               mvi     c,cioctl        ; cio control port
                        outir                   ; control bytes to cio
DF7F+EDB3
                ;
DF81 110080             lxi     d,initdlay      ; delay for system to stabilize after cio init
                initd0:
DF84 1B                 dcx     d
DF85 7A                 mov     a,d
DF86 B3                 ora     e
                        jrnz    initd0
DF87+20FB
                ;
                ;                                 now init scc
DF89 21D701             lxi     h,scctab - runa
DF8C 0610               mvi     b,sccnb         ; # of bytes
DF8E 0E7E               mvi     c,sccac         ; scc channel a control
                        outir                   ; control bytes to scc a
DF90+EDB3
DF92 21D701             lxi     h,scctab - runa
DF95 0610               mvi     b,sccnb
DF97 0E7C               mvi     c,sccbc         ; scc channel b control
                        outir                   ; control bytes to scc b
DF99+EDB3
                ;
                ;                                 print signon msg
DF9B 213500             lxi     h,signon$msg - 1 - runa
                        lxix    sm$ret
DF9E+DD21
DFA0+A200
00A2 =          sm$ret          equ     $ - runa
```

```
DFA2 23             inx     h
DFA3 7E             mov     a,m
DFA4 B7             ora     a
DFA5 C24B01         jnz     out$char - runa


             ...


             ;
             init$mpr:                      ; enter with hl => map table
DFDE 0E71           mvi     c,mapper
DFE0 56             mov     d,m
DFE1 23             inx     h
DFE2 7E      initml:        mov     a,m             ; lower data for mapper
DFE3 23             inx     h
DFE4 46             mov     b,m             ; upper data/address
DFE5 23             inx     h
                    outp    a               ; write to mapper
DFE6+ED79
DFE8 15             dcr     d               ; dec word count
                    jrnz    initml
DFE9+20F7
                    pcix
DFEB+DDE9
             ;


             ...


             ;
             out$char                       ; output accum to both possible consoles
E04B D301           out     vdbd
E04D D37D           out     sccbd
             ;
E04F 110004         lxi     d,codlay        ; delay - no status available yet
             dlay:
E052 1B             dcx     d
E053 7A             mov     a,d
E054 B3             ora     e
                    jrnz    dlay
E055+20FB
             ;
             .      pcix                    ; jump back by ix
E057+DDE9
             ;


             ...
```

```
                    ;#############################################
                    ;                                            #
                    ;     memory mapper - initialization data    #
                    ;             0000-3fff :  onboard prom      #
                    ;             4000-7fff :  onboard ram       #
                    ;             8000-ffff :  onboard ram low/hi #
                    ;                                            #
                    ;#############################################
                    ;
                    ;
                    map$init:
E06D 10                     db      16
E06E FF0FFE1FFD             dw      not 0f000h,not 0e001h,not 0d002h,not 0c003h
E076 EB4FEA5FE9             dw      not 0b014h,not 0a015h,not 09016h,not 08017h
E07E EF8FEE9FED             dw      not 07010h,not 06011h,not 05012h,not 04013h
E086 EBCFEADFE9             dw      not 03014h,not 02015h,not 01016h,not 00017h
                    ;
                    map$high:
E08E 08                     db      8
E08F E78FE69FE5             dw      not 07018h,not 06019h,not 0501ah,not 0401bh
E097 E3CFE2DFE1             dw      not 0301ch,not 0201dh,not 0101eh,not 0001fh
                    ;
                    ;
                    ;#############################################
                    ;                                            #
                    ;     memory mapper - map all ram            #
                    ;             0000-ffff :  onboard ram       #
                    ;                                            #
                    ;#############################################
                    ;
                    ;
                    map$ram:
E09F 10                     db      16              ; number of words of data
E0A0 EF0FEE1FED             dw      not 0f010h,not 0e011h,not 0d012h,not 0c013h
E0A8 EB4FEA5FE9             dw      not 0b014h,not 0a015h,not 09016h,not 08017h
E0B0 E78FE69FE5             dw      not 07018h,not 06019h,not 0501ah,not 0401bh
E0B8 E3CFE2DFE1             dw      not 0301ch,not 0201dh,not 0101eh,not 0001fh
                    ;
                    ;
                    ;#############################################
                    ;                                            #
                    ;     cio control byte tables                #
                    ;                                            #
                    ;#############################################
                    ;
                    ciotab:
E0C0 0001                   db      00h, 0000$0001b         ; reset cio chip
E0C2 00                     db           0000$0000b         ;  2 byte sequence
                    ;
                    ;     those commented out are in that state already by above reset
                    ;       db      05h, 0000$0000b         ; dppr-c: non inverting
E0C3 0602                   db      06h, 0000$0010b         ; ddr-c : all out except pberr
                    ;       db      07h, 0000$0000b         ; sior-c: normal i/o
                    ;       db      01h, 0001$0000b         ; must have port c enabled
                    ;
```

```
                ;               above is the bare minimum that the cio requires
                ;               the following inits port-b to handle vectored interrupts
                ;               off the s-100 bus.
                ;
E0C5 2806               db      28h, 0000$0110b         ; pmsr-b: bit port,or-pev
                ;       db      29h, 0000$0000b         ; phsr-b
                ;       db      2ah, 0000$0000b         ; dppr-b: non inverting
E0C7 2BFF               db      2bh, 1111$1111b         ; ddr-b : all inputs
E0C9 2CFF               db      2ch, 1111$1111b         ; sior-b: all 1's catchers
                ;                                       ;         2d thru 2f define ptrn msk
E0CB 2DFF               db      2dh, 1111$1111b         ; ppr-b : set for interrupt on any
                ;       db      2eh, 0000$0000b         ; ptr-b : 1 bit enabled by mask in 2f,
E0CD 2F40               db      2fh, active$vctrs       ; pmr-b : according to system equate
                ;
E0CF 0310               db      03h, ivtciob and 00ffh  ; ivr-b : lsb of vector table
E0D1 09C0               db      09h, 1100$0000b         ; csr-b : ei on port b
                ;
E0D3 0190               db      01h, 1001$0000b         ; mccr  : leave b,c enabled
E0D5 0088               db      00h, 1000$1000b         ; micr  : ei, vis-b
0017 =          cionb equ       $-ciotab
                ;
00B4 =          slvenb          equ     1011$0100b              ; clear slave reset
                ;
                ;#############################################
                ;                                           #
                ;       scc control byte table              #
                ;                                           #
                ;#############################################
                ;
                scctab:
                        if mhz4
E0D7 03C1               dw      0c103h
E0D9 0444               dw      4404h           ; #16 clock
E0DB 05EA               dw      0ea05h
E0DD 0B56               dw      560bh
E0DF 0C0B               dw      0b0ch
E0E1 0D00               dw      000dh           ; 9600 baud at 4mhz
E0E3 0E03               dw      030eh
E0E5 0F00               dw      000fh
                        else
                         if mhz6
                        dw      0c103h
                ;       dw      0404h           ; #1 clock
                        dw      4404h           ; #16 clock
                        dw      0ea05h
                        dw      560bh
                ;       dw      360ch
                ;       dw      010dh           ; 9600 baud at 6mhz
                        dw      120ch
                        dw      000dh           ; 9600 baud at 6mhz
                        dw      030eh
                        dw      000fh
                         endif
                        endif
0010 =          sccnb equ       $-scctab
                ;
```

```
                    ;#####################################################
                    ;                                                    #
                    ;       s-100 vectored interrupts (via cio port b)   #
                    ;       interrupt vector table                       #
                    ;                                                    #
                    ;#####################################################
                    ;
                    vtable:
E0E7 30E8                   dw      vi7
E0E9 37E8                   dw      vi6
E0EB 31E8                   dw      vi5
E0ED 2BE8                   dw      vi4
E0EF 25E8                   dw      vi3
E0F1 1FE8                   dw      vi2
E0F3 C5E7                   dw      pberr$sr        ; parity is on vi-1
E0F5 19E8                   dw      vi0
                    ;
                    ;
                    ;#############################################
                    ;                                            #
                    ;       system configuration info            #
                    ;       to be moved up to ff00               #
                    ;                                            #
                    ;#############################################
                    ;
                    sys$data:
E0F7 00DF                   dw      mon$strt        ; monitor start address
E0F9 0040                   dw      conf$init       ; 16 bit configuration vector
E0FB 20                     db      ver$info        ; monitor version #
E0FC 000000                 db      0,0,0           ; spare
                    ;
                    ;
```

```
                ;#################################################
                ;                                               #
                ;       begin high memory execution             #
                ;                                               #
                ;#################################################
                ;
                init2:
E0FF 219FE0             lxi     h,map$ram       ; map out prom
E102 CDDCDF             call    mapit
                ;
E105 21F7E0             lxi     h,sys$data      ; move up sysinfo
E108 1100FF             lxi     d,sysinfo
E10B 010800             lxi     b,8
                        ldir
E10E+EDB0
                ;
E110 21E7E0             lxi     h,vtable        ; cio port b interrupt vectors service routines
                                                ; ( i.e. s-100 vectored interrupts )
E113 1110FF             lxi     d,ivtciob       ; ram location port b iv table
                ;
E116 7A                 mov     a,d             ; im2 interrupts base page
                        stai                    ;  into i register
E117+ED47
E119 32FBFF             sta     ui              ; and into user ram register map
                ;
E11C 011000             lxi     b,16            ; eight vectors
                        ldir                    ; move them to ram
E11F+EDB0
                ;
E121 0E00               mvi     c,0             ; rst$ciob takes mask in reg c
E123 CD0CE9             call    rst$ciob        ; clear any interrupts from init
                ;
E126 3EC3               mvi     a,0c3h          ; fix nmi vector
E128 326600             sta     66h
E12B 210FE8             lxi     h,nmie          ; nmi entry
E12E 226700             shld    67h
                ;
E131 3EB4               mvi     a,slvenb        ; release slave reset line
E133 D378               out     cioc
                ;
                        im2
E135+ED5E
                ;
                ;
                ;       ###############################################
                ;       # CHECK FOR CONTROLLER TYPE AND SET ACCORDINGLY #
                ;       ###############################################
                ;
E137 AF                 xra     a
E138 3273E1             sta     cflag           ; initialize controller flag to zero
E13B DB53               in      sector          ; check for VFW-III
E13D 4F                 mov     c,a             ; save char in case it is
E13E 3E55               mvi     a,55h           ; send test byte
E140 D353               out     sector
E142 DB53               in      sector
```

```
E144 FE55           cpi     55h             ; see if they match
                    jrnz    set$vf2
E146+2008
E148 3EAA           mvi     a,0aah          ; send test byte
E14A D353           out     sector
E14C DB53           in      sector
E14E FEAA           cpi     0aah            ; see if they match
            set$vf2:
                    jrz     is3             ; yes- is VFW3 so restore char & continue
E150+2822
E152 2124F1         lxi     h,home2         ; reset jmp table to VF-II entries
E155 2219DF         shld    h2+1
E158 2133F1         lxi     h,read2
E15B 2228DF         shld    r2+1
E15E 214FF1         lxi     h,write2
E161 222BDF         shld    w2+1
E164 2167F3         lxi     h,loader2
E167 222EDF         shld    l2+1
E16A 2174F3         lxi     h,saver2
E16D 2231DF         shld    s2+1
E170 C380E1         jmp     rentry          ; entry warm boot point
E173 00     cflag:          db      0
            ;
E174 79     is3:    mov     a,c             ; restore char in vfw3
E175 D353           out     sector
E177 3EFF           mvi     a,0ffh          ; set cflag to non-zero
E179 3273E1         sta     cflag
            ;
E17C 3E13           mvi     a,13h           ; now do restore to clear controller
E17E D357           out     command         ;  after its self test
            ;
            ;#####################################################
            ;                                               #
            ;       warm boot entry point                   #
            ;                                               #
            ;#####################################################
            ;
            rentry:
E180 2100DF         lxi     h,usrsp         ; init user sp save loc
E183 22E6FF         shld    usp             ; load user sp loc
E186 2180E1         lxi     h,rentry        ; make .G default = exec0 entry
E189 22FEFF         shld    upc
E18C AF             xra     a
E18D 3200FF         sta     sbpk            ; clear software bp flag
E190 325DFF         sta     ivn             ; clear interrupt flag
E193 32FAFF         sta     uif             ; set user interrupts - disabled
E196 325EFF         sta     par$loc         ; clear parity known flag
E199 3C             inr     a
E19A 32C6FF         sta     long            ; set long register print
            ;
E19D FB             ei                      ; enable interrupts immedately before
            ;                               ; fall through to command interpreter loop
```

```
                ;*********************************************
                ;                                           *
                ;       command re-entry point              *
                ;                                           *
                ;*********************************************
                ;
E19E 31C0FF     exec0:          lxi     sp,mossp1
E1A1 CDCEE9             call    crlf
E1A4 0E2E               mvi     c,'.'
E1A6 CD0CDF             call    coe             ; write the character
E1A9 CDC7E9             call    echo            ; collect the command
E1AC FE2E               cpi     '.'
                        jrz     exec0
E1AE+28EE
E1B0 C5                 push    b
E1B1 CDD8E9             call    space
E1B4 CDFEEA             call    scan            ; scan for the operands to the command
E1B7 C1                 pop     b
E1B8 79                 mov     a,c
E1B9 219EE1             lxi     h,exec0
E1BC E5                 push    h               ; push return address
                ;
E1BD D641       cmdact:         sui     'A'
                        jrc     invcmd
E1BF+3851
E1C1 FE1A               cpi     'Z'-'A'+1
                        jrnc    invcmd
E1C3+304D
E1C5 21DEE1             lxi     h,cmdvct
E1C8 87                 add     a
E1C9 85                 add     l
E1CA 6F                 mov     l,a
                        jrnc    cmdac1
E1CB+3001
E1CD 24                 inr     h
E1CE 5E         cmdac1:         mov     e,m
E1CF 23                 inx     h
E1D0 56                 mov     d,m
E1D1 D5                 push    d               ; push destination address
E1D2 2AD2FF             lhld    opr2            ; hl,de set operands
                        lded    opr1
E1D5+ED5B
E1D7+D0FF
                        lxiy    spad            ; point iy to spad
E1D9+FD21
E1DB+C0FF
E1DD C9                 ret                     ; jump to it
                ;
                cmdvct:
E1DE 1AE236E275         dw      acmd,  bcmd,  ccmd,  dcmd
E1E6 EEE226E338         dw      ecmd,  fcmd,  gcmd,  hcmd
E1EE 6BE39EE312         dw      icmd,  jcmd,invcmd,  lcmd
E1F6 BBE412E2DE         dw      mcmd,invcmd,  ocmd,  pcmd
E1FE 12E240E54D         dw    invcmd,  rcmd,  scmd,  tcmd
E206 12E22EE766         dw    invcmd,  vcmd,  wcmd,  xcmd
```

```
E20E 9CE79FE7         dw      ycmd, zcmd
                ;
E212 0E3F       invcmd:       mvi     c,'?'
E214 CD0CDF           call    coe
E217 C39EE1           jmp     exec0
                ;
                ;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX K
SAMPLE I/O SUPPORT CODE

RESETTING VECTORED INTERRUPTS

```
;
;       reset ciob data catcher(s) and port b interrupt state.
;                       assumes interrupts have been disabled by caller
;                       on entry reg c must have mask for ciob port.
;                       i.e. to reset D6, c = 1011$1111b
;
;                       interrupts must be disabled through all code
;                        that accesses cio internal registers.
;
;
ciob            equ     79h             ; SBC-300 CIO port b data port
cioctl          equ     7bh             ; SBC-300 CIO control port
csr$b           equ     09h             ; cmd/status port b
rst$csr$b       equ     0010$0000b      ; reset ius,ip in cmd/status port
;
rst$ciob:
        in      ciob            ; first reset 1's catcher @ data port b
        ana     c               ;  according to flag passed in reg c
        out     ciob            ; 0's clear
;
        mvi     a,csr$b         ; cmd/status for port b
        out     cioctl          ; register select
        mvi     a,rst$csr$b     ; clear ius & ip
        out     cioctl          ; reset ip,ius
;                                 ignoring errors for now
        ret
;
```

MONITOR INITIALIZATION CODE

```
;
false equ 0
true  equ not false
;
sasi            equ     false
cntrs           equ     true
cntr3           equ     false
;
;    I/O Port addresses S D Systems SBC-300 cpu
;
        ; chip bases
;
;
p$300par        equ 70h         ; SBC-300 Parallel Data Port
p$300cio        equ 78h         ; Z8536 Counter/Timer & Parallel I/O
;
;
        ; Z8536 Counter/Timer & Parallel I/O Chip
;
p$cioc          equu    p$300cio+0
p$ciob          equ     p$300cio+1
p$cioa          equ     p$300cio+2
p$cioctl        equ     p$300cio+3
;
;
;
sp$sav  equ     0ff7eh          ; space for interrupts in page ff
int$stk equ     sp$sav
;
;
;
;       SBC-300  cio init
;
cioinit:
;
;                                 set up z-80 i-reg
        mvi     a,iv$page
        stai
;
;                                 set up im2 vectors in page.0ffh
        lxi     h,i$sr          ; just move the table up there
        lxi     d,i$sr$tab
        lxi     b,i$sr$nb
        ldir
;
        in      p$cioctl        ; force cio to state 0
;
        mvi     c,p$cioctl      ; send init info
        mvi     b,cionb
        lxi     h,ciotab
        outir
;
```

```
;
        im2
        ei
        ret
;
;
;       data for sbc$300 cio initialization
;
;
iv$page         equ     0ffh    ; all im2 vectors are in page 0ffh
i$sr$tab        equ     0ff08h  ; start of im2 vector table in page 0ffh
sr$cio$ct       equ     08h     ; 0ff08 - 0ff0f reserved for cio cntr/timers
sr$cio$b        equ     10h     ; 0ff10 - 0ff1f reserved for cio port b
sr$cio$a        equ     20h     ; 0ff20 - 0ff2f reserved for cio port a
sr$scc          equ     30h     ; 0ff30 - 0ff3f reserved for scc
;
;
i$sr:                           ; service routine addresses to move up
;                                 cio cntr/timers require 4 addresses
sr$cioct3       dw      sr$ct3  ; service routine address
sr$cioct2       dw      sr$ct2  ; service routine address
sr$cioct1       dw      sr$ct1  ; service routine address
sr$ciocte       dw      ct$err  ; service routine address
;                                 cio port b requires 8 addresses
sr$ciob0        dw      sr$vir7 ; service routine address
sr$ciob1        dw      sr$vir6 ; service routine address
sr$ciob2        dw      sr$vir5 ; service routine address
sr$ciob3        dw      sr$vir4 ; service routine address
sr$ciob4        dw      sr$vir3 ; service routine address
sr$ciob5        dw      sr$vir2 ; service routine address
sr$ciob6        dw      sr$vir1 ; service routine address
sr$ciob7        dw      sr$vir0 ; service routine address
actv$ints       equ     0000$0000b
                                ; any bit set will activate the VI
;                                 e.g. the following would activate
;                                 the parity error line:
;perr$enbl      equ     0100$0000b
;actv$ints      equ     perr$enbl
;
;
;                                 cio port a requires 8 addresses
; cio port a interrupts not currently used
;                                 scc requires 8 addresses
; scc interrupts not currently used
;
i$sr$nb         equ     $-i$sr  ; number of bytes in table
;
```

```
;
;***********************************************
;                                             *
;       cio control byte table                *
;                                             *
;***********************************************
;
;               cio port a data
;
        if sasi
;
d$pms$a equ     0000$0000b      ; bit port, no pattern match
d$phs$a equ     0000$0000b      ;-ignored if bit port
d$dpp$a equ     1111$1111b      ; sasi is negative logic
d$dd$a  equ     0011$1110b      ; sel,bsy-out,i/o,req,c/d,msg,bsy-in,rst
d$sio$a equ     0000$0000b      ; normal io bits
;                                         2d thru 2f define ptrn msk
d$pp$a  equ     0000$0000b      ;-ignored when no pattern match used
d$pt$a  equ     0000$0000b      ; leave all masked off
d$pm$a  equ     0000$0000b      ;
;
d$iv$a  equ     sr$cio$a        ; lsb of vector table
d$dr$a  equ     0000$0000b      ; init port to all 0's
;
        else    ; not sasi
;
d$pms$a equ     0000$0000b      ; bit port, no pattern match
d$phs$a equ     0000$0000b      ;-ignored if bit port
d$dpp$a equ     1010$0100b      ; invert strobe, ?, dev$rdy
d$dd$a  equ     0000$1100b      ; pa7-strobe, pa3-on$line, pa2-dev$rdy
d$sio$a equ     0000$0000b      ; normal io bits
;                                         2d thru 2f define ptrn msk
d$pp$a  equ     0000$0000b      ;-ignored when no pattern match used
d$pt$a  equ     0000$0000b      ; leave all masked off
d$pm$a  equ     0000$0000b      ;
;
d$iv$a  equ     sr$cio$a        ; lsb of vector table
d$dr$a  equ     0000$0000b      ; set strobe at data port
;
        endif   ; not sasi
;
;               cio port b data
;
d$pms$b equ     0000$0110b      ; bit port,or-pev
d$phs$b equ     0000$0000b      ;
d$dpp$b equ     0000$0000b      ; non inverting
d$dd$b  equ     1111$1111b      ; all inputs
d$sio$b equ     1111$1111b      ; all 1's catchers
;                                         2d thru 2f define ptrn msk
d$pp$b  equ     1111$1111b      ; set for interrupt on any
d$pt$b  equ     0000$0000b      ;   bit enabled by mask in 2f
d$pm$b  equ     actv$ints       ; whichever bits are set are enabled
```

```
;
d$iv$b  equ     sr$cio$b                ; lsb of vector table
;
;               cio cntr/timer data
;
d$pms$3 equ     1000$0011b      ; continuous, no external pins, no dcs
d$pms$1 equ     1000$0000b      ; continuous, no external pins, pulse output
d$pms$2 equ     1000$0011b      ; continuous, no external pins, no dcs
d$tcm$3 equ     244
d$tcl$3 equ     65              ; = 62500; = 1/32 second
d$tcm$1 equ     244
d$tcl$1 equ     65              ; = 62500; = 1/32 second
d$tcm$2 equ     0
d$tcl$2 equ     32              ; = 32; for 1 second
;
d$iv$ct equ     sr$cio$ct       ; note that cio chip provides bits d2,d1
;
;               cio c/s register commands
;
d$cisfs equ     0010$0000b      ; clear interrupt service flags (ip, ius)
d$sei   equ     1100$0000b      ; set ie
d$cei   equ     1110$0000b      ; clear ie
d$trgr  equ     0000$0110b      ; trigger a c/t without gating
;
;               begin ciotab
;
ciotab:
        db      00h, 0000$0000b         ; micr  : disable mie
        db      01h, 0001$0000b         ; mccr  : must leave port c enabled
;
;               port a first            parallel port status in port a
;
;
        db      20h, d$pms$a    ; pattern mode specification
;       db      21h, d$phs$a    ; pattern handhake specification
        db      22h, d$dpp$a    ; data path polarity
        db      23h, d$dd$a     ; data direction
        db      24h, d$sio$a    ; special io
;
;       db      25h, d$pp$a     ; port polarity
;       db      26h, d$pt$a     ; port transition
;       db      27h, d$pm$a     ; port mask
;
        db      02h, d$iv$a     ; interrupt vector
        db      08h, d$cei      ; command status
;
```

```
;
;           next, port b            init'd to handle vector'd interrupts
;
        db      28h, d$pms$b    ; pattern mode specification
;       db      29h, d$phs$b    ; pattern handhake specification
        db      2ah, d$dpp$b    ; data path polarity
        db      2bh, d$dd$b     ; data direction
        db      2ch, d$sio$b    ; special io
;
        db      2dh, d$pp$b     ; port polarity
        db      2eh, d$pt$b     ; port transition
        db      2fh, d$pm$b     ; port mask
;
        db      03h, d$iv$b     ; interrupt vector
        db      09h, d$sei      ; command status
;
;
;           finally, c/t's          ; use either 1&2 linked, or 3 alone
;
        db      1ch, d$pms$1    ; c/t-1 mode specification register
        db      1dh, d$pms$2    ; c/t-2 mode specification register
        db      1eh, d$pms$3    ; c/t-3 mode specification register
;
        db      16h, d$tcm$1    ; c/t-1 time constant register msb
        db      17h, d$tcl$1    ; c/t-1 time constant register lsb
        db      18h, d$tcm$2    ; c/t-2 time constant register msb
        db      19h, d$tcl$2    ; c/t-2 time constant register lsb
        db      1ah, d$tcm$3    ; c/t-3 time constant register msb
        db      1bh, d$tcl$3    ; c/t-3 time constant register lsb
;
        db      04h, d$iv$ct    ; c/t interrupt vector register
;
        db      0ah, d$cei      ; c/t-1 command/status register
        db      0ah, d$cisfs
;
        if not cntrs            ; disable interrupts on all c/t's
;
        db      0bh, d$cei      ; c/t-2 command/status register
        db      0bh, d$cisfs
        db      0ch, d$cei      ; c/t-3 command/status register
        db      0ch, d$cisfs
;
        else    ; cntrs           enable interrupts on either c/t-2 of c/t-3
;
        if cntr3                ; enable interrupts on c/t-3
;
        db      0bh, d$cei      ; c/t-2 command/status register
        db      0bh, d$cisfs
        db      0ch, d$sei      ; c/t-3 command/status register
;
        else    ; not cntr3       enable interrupts on c/t-2
;
        db      0bh, d$sei      ; c/t-2 command/status register
        db      0ch, d$cei      ; c/t-3 command/status register
        db      0ch, d$cisfs
```

```
;
        endif   ; not cntr3
;
        endif   ; cntrs
;
;
        db      01h, 1111$0111b ; master configuration control
;
;
        db      0bh, d$trgr     ; c/t-2 command/status register
        db      0ah, d$trgr     ; c/t-1 command/status register
        db      0ch, d$trgr     ; c/t-3 command/status register
;
;
        db      00h, 1000$1100b ; master interrupt control
;
cionb   equ     $-ciotab
;
;
;
;
;
;       routine to service counter/timer err and transfer control
;         to routine of highest priority counter/timer needing
;         service
;
;
csr$ct3         equ     0ch             ; command/status register, c/t-3
err$msk         equ     0001$0000b      ; ERR bit in csr's
;
ct$err:                 ; z80 has disabled interrupts
        sspd    sp$sav          ; save state
        lxi     sp,int$stk
        push    psw
        push    h
        push    b
;
        lxi     h,i$sr$tab - 2  ; c/t isr's are 1st in table
        mvi     c,p$cioctl
        mvi     b,csr$ct3 + 1   ; c/t-3 is highest priority
ct$err0:
        inx     h               ; 1 word per address
        inx     h
        dcr     b               ; csr's are sequential
        outp    b               ; read the csr
        inp     a
        ani     err$msk         ; check ERR bit
        jrz     ct$err0         ; if not set then go check next c/t
```

```
;
        mov     c,m             ; else set up address of service routine
        inx     h
        mov     b,m
        lhld    sp$sav          ; put it on user's stack (no choice)
        dcx     h
        mov     m,b
        dcx     h
        mov     m,c
        shld    sp$sav          ; update user's stack ptr
;
        pop     b               ; restore state
        pop     h
        pop     psw
        lspd    sp$sav
        ret                     ; and jump to service routine
;                                 with interrupts still disabled
;
        end
```

APPENDIX M
DISCLAIMER

SDSystems, INC. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, SDSystems, INC. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of SDSystems, INC. to notify any person of such revision or changes.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX N
LIMITED WARRANTY

All SDSystems printed circuit board assemblies are warranted for a period of one (1) year from date of invoice to be free from defects of material and workmanship.

Should an SDSystems board fail to perform to specifications, obtain a Return Material Authorization (RMA) number from your distributor or from SDSystems. Include this number in all correspondence and with the returned product. Ship the item prepaid to SDSystems and it will, at our option, be repaired or replaced free of charge provided the unit is received during the warranty period.

In order to validate this warranty, the enclosed warranty card must be returned to SDSystems. If no warranty card is on file at the time of product return, dated proof of purchase will be required.

This warranty is invalid if product has been misused or improperly modified. Modifications documented in the SDSystems unit publications may be performed without invalidating the warranty. All other modifications will invalidate the warranty. Warranty is limited to replacement of defective parts and no responsibility is assumed for damage to other equipment.

SDSYSTEMS MAKES NO WARRANTIES, GUARANTEES, OR REPRESENTATIONS, EXPRESSED OR IMPLIED, WITH RESPECT TO THE PRODUCTS COVERED HEREBY, EXCEPT AS EXPRESSED HEREIN, AND BUYER EXPRESSLY WAIVES ANY OTHER WARRANTIES, GUARANTEES, OR REPRESENTATIONS INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR USE. SDSYSTEMS NEITHER ASSUMES NOR AUTHORIZES ANY OTHER PERSON TO ASSUME FOR SELLER ANY OTHER LIABILITIES IN CONNECTION WITH THE SALE OF THE PRODUCTS. IN NO EVENT WILL SDSYSTEMS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX O
PARTS LIST FOR SBC-300

QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
	SBC-300 4 MHZ		
0	SBC-300 BOARD SCHEMATIC	0300723	
1	SBC-300 PC BOARD	7000066	
1	IC, 7406	7010007	U45
1	IC, 7438	7010030	U5
4	IC, 74LS00	7010160	U8, U16, U24, U30
5	IC, 74LS02	7010162	U4, U26, U27, U51, U68
2	IC, 74LS04	7010164	U32, U37
2	IC, 74LS08	7010166	U34, U75
1	IC, 74LS10	7010168	U17
5	IC, 74LS14	7010172	U6, U19, U52, U53, U60
1	IC, 74LS20	7010174	U56
2	IC, 74LS32	7010181	U18, U20
7	IC, 74LS74	7010195	U1, U2, U7, U9, U33, U43, U74
1	IC, 74LS93	7010205	U42
1	IC, 74LS112	7010210	U15
1	IC, 74LS139	7010220	U57
2	IC, 74LS158	7010229	U64, U72
1	IC, 74LS163	7010233	U25
1	IC, 74LS164	7010234	U67
3	IC, 74LS240	7010260	U13, U21, U35
2	IC, 74LS244	7010264	U55, U71
5	IC, 74LS245	7010265	U61, U70, U76-U78
1	IC, 74LS280	7010279	U39
1	IC, 74LS368	7010303	U36
2	IC, 74LS373	7010304	U29, U40
1	IC, 74LS374	7010305	U38
2	IC, 75188 OR MC1488	7010332	U10, U11
2	IC, 75189 OR MC1489	7010333	U3, U12
1	IC, 74LS682	7010518	U69
1	8531 4 MHZ ASCC	7010536	U47
2	74LS125	7010526	U14, U44
3	74LS189	7010527	U54, U62, U63
1	IC, MK3880-4 Z80A (4 MHZ)	7010334	U48
1	8536 4 MHZ CIO	7010538	U46
9	IC, PIN 1 REFR 200 NS DRAM	7010532	U23, U31, U41, U49, U50, U58, U66, U73, U79
1	DELAY LINE, 5 TAP, 200 NS	7120010	U22
1	PAL 16L8 - DECODE PROG.	7250006	U28
1	PAL 16L8 SLAVE PROG.	7250007	U59

SBC-300 4 MHZ (Continued)

QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
1	SIP, 4.7K OHM	7010434	RN4
1	SIP, 1K OHM, 6 PIN 5 RES	7010556	RN3
2	SIP, 220/330, 10 PIN	7010530	RN1, RN2
1	RES 22 OHM 1/4W 5% CC	7020033	R8
1	RES 100 OHM 1/4W 5% CC	7020049	R14
2	RES 220 OHM 1/4W 5% CC	7020057	R4, R9
5	RES 1K OHM 1/4W 5% CC	7020073	R5-R7, R12, R16
3	RES 4.7K OHM 1/4W 5%	7020089	R3, R13, R15
2	RES 10K OHM 1/4W 5% CC	7020097	R10, R11
2	RES 150 OHM 1/2W 5% CC	7020171	R1, R2
3	10 PIN 1K RES NTWK	7020223	RN5-RN7
4	CAP 10MF 16V TANT	7030009	C1, C2, C6, C9
2	CAP 150PF	7030042	C15, C39
1	CAP 300PF 20% 50V .2	7030044	C14
33	CAP .1UF 50V	7030045	C3-C5, C7, C8, C10, C11, C16-C38, C40-C42
1	CAP .01MF 16V CRAMIC	7030046	C13
1	CAP 33PF 50V	7030047	C12
1	DIODE, 1N4001	7040002	CR3
2	ZENER DIODE 1N4742A	7040004	CR1, CR2
1	TRANS PNP 2N3906	7040006	Q1
1	8 POSITION DIP SWITCH	7050002	SW1
2	SOCKET 16 PIN 300 ML	7060003	J5, J6
1	SOCKET 28 PIN GOLD PL	7060022	XU65
1	CRYSTAL, 8 MHZ	7080007	Y1
2	6-32 X 3/8 PPH SCREW	7130006	
2	6-32 NUT	7130007	
2	LOCKWASHER #6	7130009	
1	78H05C +5V 5 AMP VLT RGLR	7160012	VR1
1	2X25 CONN, STRT HDR	7090197	J4
2	CONN, 26 PIN STRT HDR	7090196	J2, J3
2	HDR ASSY	0100736	J5, J6
1	HEATSINK TMH 6103-B	7130004	
37	65474 BERG PV JUMPER	7170004	
7	BERG 1X2 STR .230 PIN TIN	7170018	W4, W5, W9, W11, W12, W15-1, W21
5	BERG 1X3 STR .230 PIN TIN	7170021	W1, W3, W6, W10, W19
2	BERG 2X5	7170105	W13, W17
2	HDR ASSY	0100736	J5, J6
1	BERG STIK ST 2X9 TIN PL	7170097	W14
1	SIP, 1K OHM, 8 PIN RES	7010413	RN8
1	SOCKET, 40 PIN GOLD PL	7060025	XU48
1	BERG 2X4	7170020	W16
1	BERG 2X15	7170106	W15
2	PCB EJECTORS	7130228	
0	SBC-300 TEST PROCEDURE ASSY	0900723	

QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
	SBC-300 6 MHZ		
0	SBC-300 BOARD SCHEMATIC	0300723	
1	SBC-300 PC BOARD	7000066	
1	IC, 7406	7010007	U45
1	IC, 7438	7010030	U5
4	IC, 74LS00	7010160	U8, U16, U24, U30
5	IC, 74LS02	7010162	U4, U26, U27, U51, U68
2	IC, 74LS04	7010164	U32, U37
2	IC, 74LS08	7010166	U34, U75
1	IC, 74LS10	7010168	U17
5	IC, 74LS14	7010172	U6, U19, U52, U53, U60
1	IC, 74LS20	7010174	U56
2	IC, 74LS32	7010181	U18, U20
7	IC, 74LS74	7010195	U1, U2, U7, U9, U33, U43, U74
1	IC, 74LS92	7010204	U42
1	IC, 74LS112	7010210	U15
1	IC, 74LS139	7010220	U57
2	IC, 74LS158	7010229	U64, U72
1	IC, 74LS163	7010233	U25
1	IC, 74LS164	7010234	U67
3	IC, 74LS240	7010260	U13, U21, U35
2	IC, 74LS244	7010264	U55, U71
5	IC, 74LS245	7010265	U61, U70, U76-U78
1	IC, 74LS280	7010279	U39
1	IC, 74LS368	7010303	U36
2	IC, 74LS373	7010304	U29, U40
1	IC, 74LS374	7010305	U38
2	IC, 75188 OR MC1488	7010332	U10, U11
2	IC, 75189 OR MC1489	7010333	U3, U12
1	IC, 74LS682	7010518	U69
1	8531A 6 MHZ ASCC	7010520	U47
2	74LS125	7010526	U14, U44
3	74LS189	7010527	U54, U62, U63
1	Z8400B- 6 MHZ CPU (Z80B)	7010528	U48
1	8536A 6 MHZ CIO	7010529	U46
9	150 NS DRAM	7010534	U23, U31, U41, U49, U50, U58, U66, U73, U79
1	150 NS DELAY LINE	7010535	U22
1	PAL 16L8 - DECODE PROG.	7250006	U28
1	PAL 16L8 SLAVE PROG.	7250007	U59
1	SIP, 4.7K OHM	7010434	RN4

SBC-300 6 MHZ (Continued)

QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
1	SIP, 1K OHM 6 PIN 5 RES	7010556	RN3
2	SIP, 220/330, 10 PIN	7010530	RN1, RN2
1	RES 22 OHM 1/4W 5% CC	7020033	R8
1	RES 100 OHM 1/4W 5% CC	7020049	R14
2	RES 220 OHM 1/4W 5% CC	7020057	R4, R9
5	RES 1K OHM 1/4W 5% CC	7020073	R5-R7, R12, R16
3	RES 4.7K OHM 1/4W 5%	7020089	R3, R13, R15
2	RES 10K OHM 1/4W 5% CC	7020097	R10, R11
2	RES 150 OHM 1/2W 5% CC	7020171	R1, R2
3	10 PIN 1K RES NTWK	7020223	RN5-RN7
4	CAP 10MF 16V TANT	7030009	C1, C2, C6, C9
2	CAP 150PF	7030042	C15, C39
1	CAP 300PF 20% 50V .2	7030044	C14
33	CAP .1UF 50V	7030045	C3-C5, C7, C8, C10, C11, C16-C38, C40-C42
1	CAP .01MF 16V CRAMIC	7030046	C13
1	CAP 33PF 50V	7030047	C12
1	DIODE, 1N4001	7040002	CR3
2	ZENER DIODE 1N4742A	7040004	CR1, CR2
1	TRANS PNP 2N3906	7040006	Q1
1	8 POSITION DIP SWITCH	7050002	SW1
2	SOCKET 16 PIN 300 ML	7060003	J5, J6
1	SOCKET 28 PIN GOLD PL	7060022	XU65
1	XTAL, 12 MHZ	7080019	Y1
2	6-32 X 3/8 PPH SCREW	7130006	
2	6-32 NUT	7130007	
2	LOCKWASHER #6	7130009	
1	78H05C +5V 5A VLT RGLR	7160012	VR1
1	HEATSINK TMH 6103-B	7130004	
37	65474 BERG PV JUMPER	7170004	
1	SIP, 1K OHM 8 PIN RES	7010413	RN8
1	2X25 CONN, STRT HDR	7090197	J4
2	CONN, 26 PIN STRT HDR	7090196	J2, J3
2	HDR ASSY	0100736	J5, J6
1	SOCKET, 40 PIN GOLD PL	7060025	XU48
7	BERG 1X2 STR .230 PIN TIN	7170018	W4, W5, W9, W11, W12, W15-1, W21
5	BERG 1X3 STR .230 PIN TIN	7170021	W1, W3, W6, W10, W19
1	BERG STIK ST 2X9 TIN PL	7170097	W14
2	BERG 2X5	7170105	W13, W17
1	BERG 2X4	7170020	W16
1	BERG 2X15	7170106	W15
2	PCB EJECTORS	7130228	
0	SBC-300 TEST PROCEDURE ASSY	0900723	

REPLACEMENT CHIPS (Or Equivalent)

	RAMs		ROMs	Delay Lines	
	150 ns	200 ns		150 ns	200 ns
	(P)				
MITSUBISHI	M5K4164S-15	-20	2716	DDU-4-5150	DDU-4-5200
FUJITSU	MB8265-15	-20	2732		
MOSTEK	MK4164-15	-20	2764		
MOTOROLA	MCM6664-15	-20			

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX P
PARTS PLACEMENT DIAGRAM

THIS PAGE INTENTIONALLY LEFT BLANK

JUMPER NUMBERING NOTATION

Throughout the manual, jumper locations are referenced by 'W' number and pin number. The following number convention is adhered to for all jumpers on the board.

For horizontal jumpers that are single row (1x3, 1x4, etc.), pins are numbered consecutively from left to right.

For vertical jumpers that are single row (1x3, 1x4, etc.), pins are numbered consecutively from top to bottom.

For horizontal jumpers that are dual row (2x3, 2x4, etc.), pins are numbered consecutively from bottom left to right, continuing on the top right to left.

EXAMPLE: W7(1), W7(6), and W7(8) are to be jumpered.

```
          16     15     14     13     12     11     10     9

          ---                                ---           ---
         | o |    o      o      o      o    | o |    o    | o |
W7       |   |                              |   |         |   |
         | o |    o      o      o      o    | o |    o    | o |
          ---                                ---           ---

          1      2      3      4      5      6      7      8
```

For vertical jumpers that are dual row (2x3, 2x4, etc.), pins are numbered consecutively from left top to bottom, continuing on the right bottom to top.

For horizontal jumpers that are triple row (3x3, 3x4, etc.), pins are numbered consecutively from left bottom to top, continuing from bottom to top for each column from left to right.

EXAMPLE: W3(4-5) is to be jumpered.

```
                        6      9

               3   o    o      o      o   12
                       ---
W3             2   o   | o |   o      o   11
                       |   |
               1   o   | o |   o      o   10
                       ---
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L
JUMPER CONNECTION QUICK REFERENCE TABLES

STANDARD JUMPER CONNECTIONS

(Example: "1-2" means jumper pins 1 and 2 together)

	Master	Slave
W1	OUT	OUT
W3	1-2	1-2
W4	OUT	IN
W5	IN	OUT
W6	1-2	1-2
W9	IN	OUT
W10	1-2	1-2
W11	IN	OUT
W12	IN	IN
W13	POSITION 1	POSITION 1
W14	POSITION 5	POSITION 5
W15-W18	ALL IN	CUT ALL
W19	OUT	2-3
W21	IN	OUT

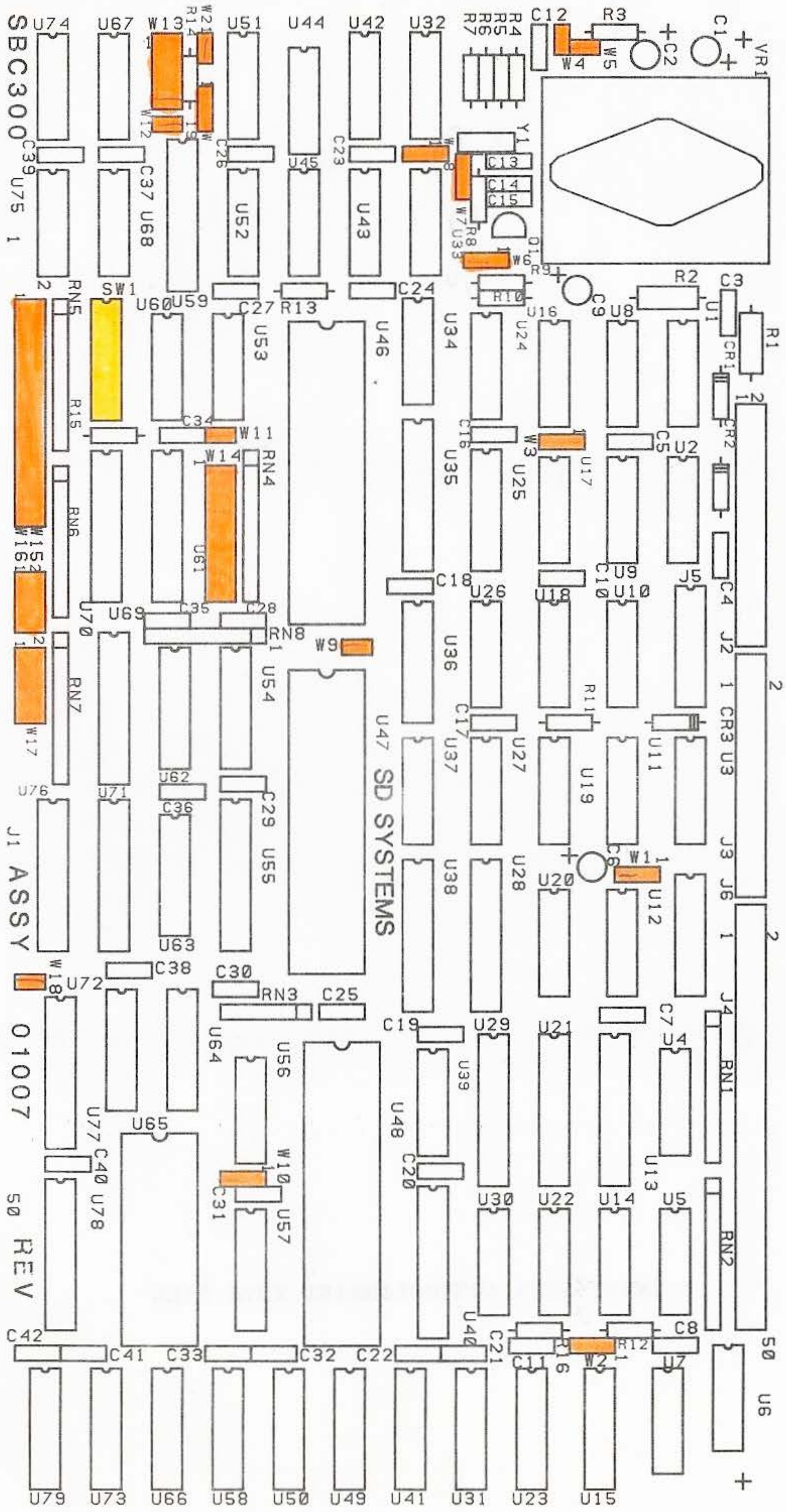
JUMPER DEFINITIONS

NOTE: "(etch)" means an etch jumper. If a change is desired, the user must cut the etch and insert a jumper.

Jumper	Purpose	Setup
W1	SCC Port A Baud CLK select	1-2 External (modem) clock 2-3 2 mHz system clock
W2 (etch)	RAM size select	1-2 64K DRAM 2-3 Future expansion
W3	Wait state generator	1-2 Memory cycle 2-3 OP CODE fetch only (M1)
W4	Master/Slave select	IN Slave OUT Master
W5	Bus refresh signal pin 66 (for SD EXPANDORAM II & III)	IN Outputs RFSH* signal OUT No connection to bus
W6	pSTVAL* source select	1-2 IEEE-696 timing 2-3 Ø1 clock
W7 (etch)	Bus clock select	1-2 Future expansion 2-3 4/6 mHz board
W8 (etch)	CPU clock select	1-2 4/6 mHz board 2-3 Future expansion
W9	WAIT* for EXPANDORAM III compat.	IN EXPANDORAM III OUT All others
W10	PROM select	1-2 64K and smaller 2-3 128K

JUMPER DEFINITIONS--Continued

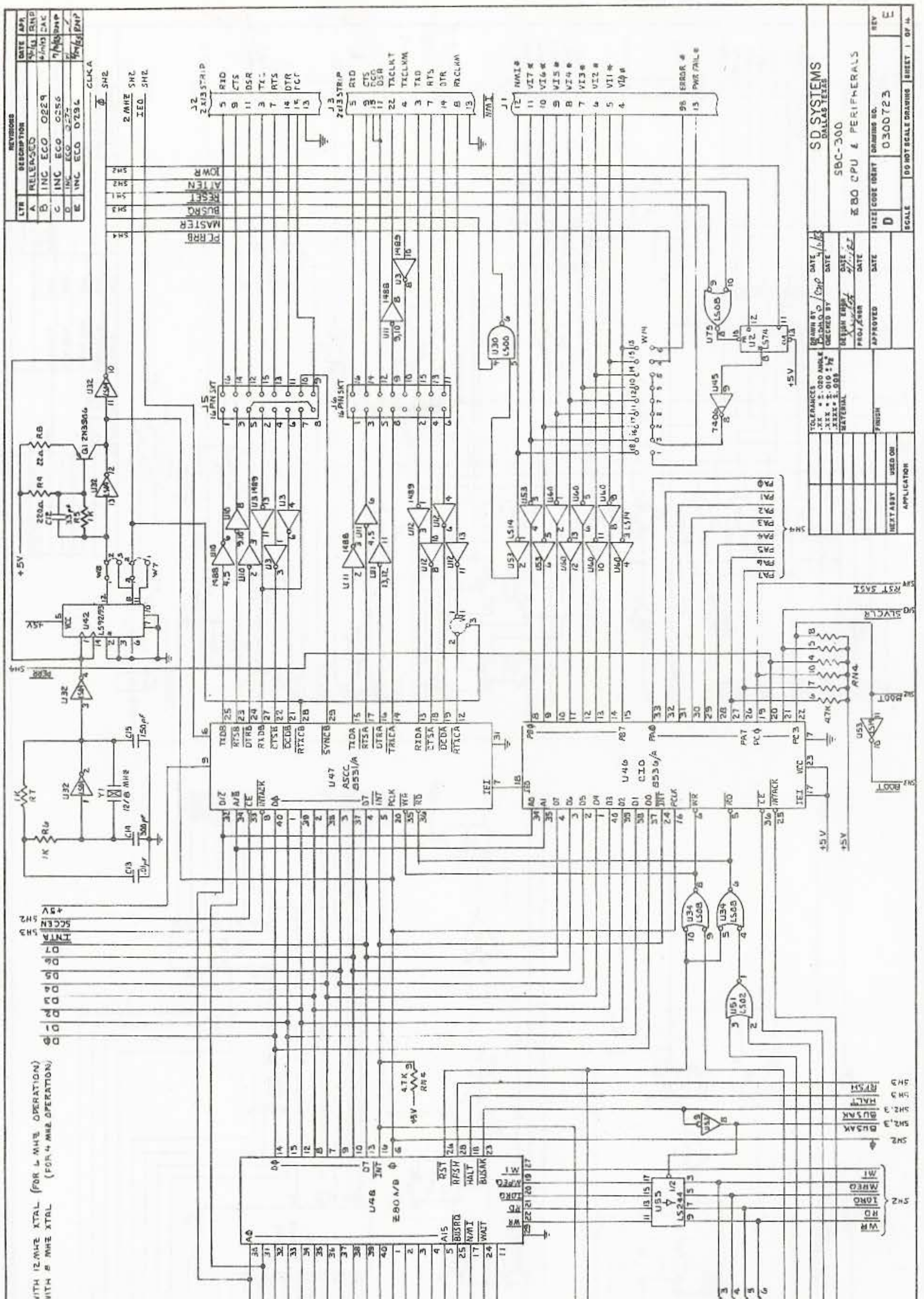
Jumper	Purpose	Jumper #	Setup	
W11	MWRT select		IN	SBC-300 creates MWRT
			OUT	MWRT created elsewhere
W12	BRFSH jumper		IN	Refresh during TMA control
			OUT	Ignore refresh
W13	TMA wait generator		1-5	Wait states inserted
W14	Interrupt select	1		PWR FAIL to NMI
		2		Attention to VI6
		3		Attention to VI7
		4		Parity error to VI1
		5		Attention to VI2
		6		ERROR* to VI0
		7		Attention to VI4
		8		Attention to VI5
		9		Attention to VI3
W15-W18	Bus pull up resistors for master; remove for slave			
W19	SLVCLR select		1-2	Manual reset for slave
			2-3	SLVCLR from bus
W21	RDY/XRDY enable		IN	RDY/XRDY gated to WAIT*
			OUT	RDY/XRDY not used

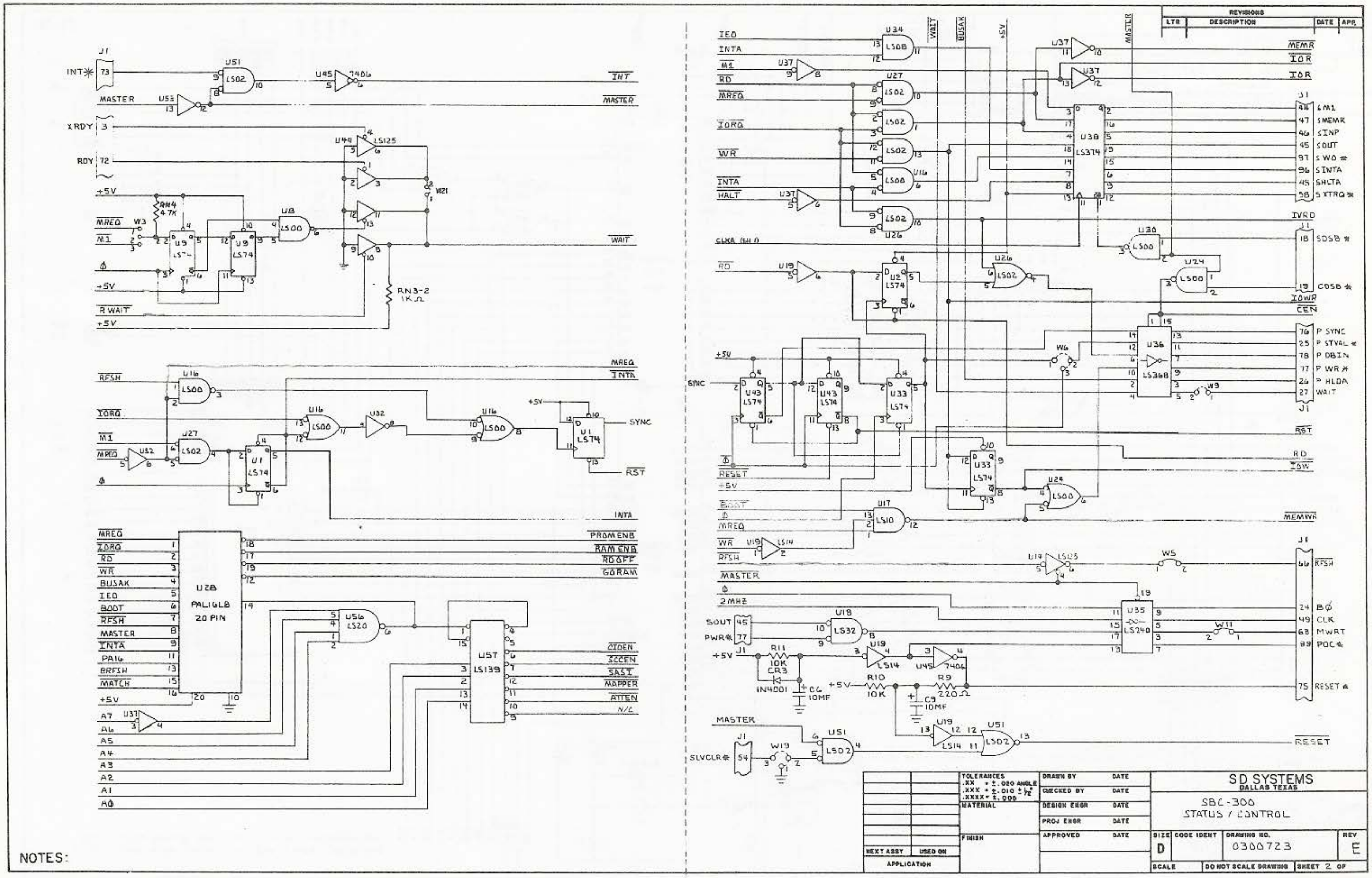

SBC300 J1 ASSY 01007 50 REV
SD SYSTEMS
VR1
SW1
J1
J2
J3
J4
J5
J6
RN1
RN2
RN3
RN4
RN5
RN6
RN7
RN8
W1
W2
W3
W4
W5
W6
W7
W8
W9
W10
W11
W12
W13
W14
W15
W16
W17
W18
W21
Y1
Q1
CR1
CR2
CR3

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX Q
SCHEMATIC

THIS PAGE INTENTIONALLY LEFT BLANK

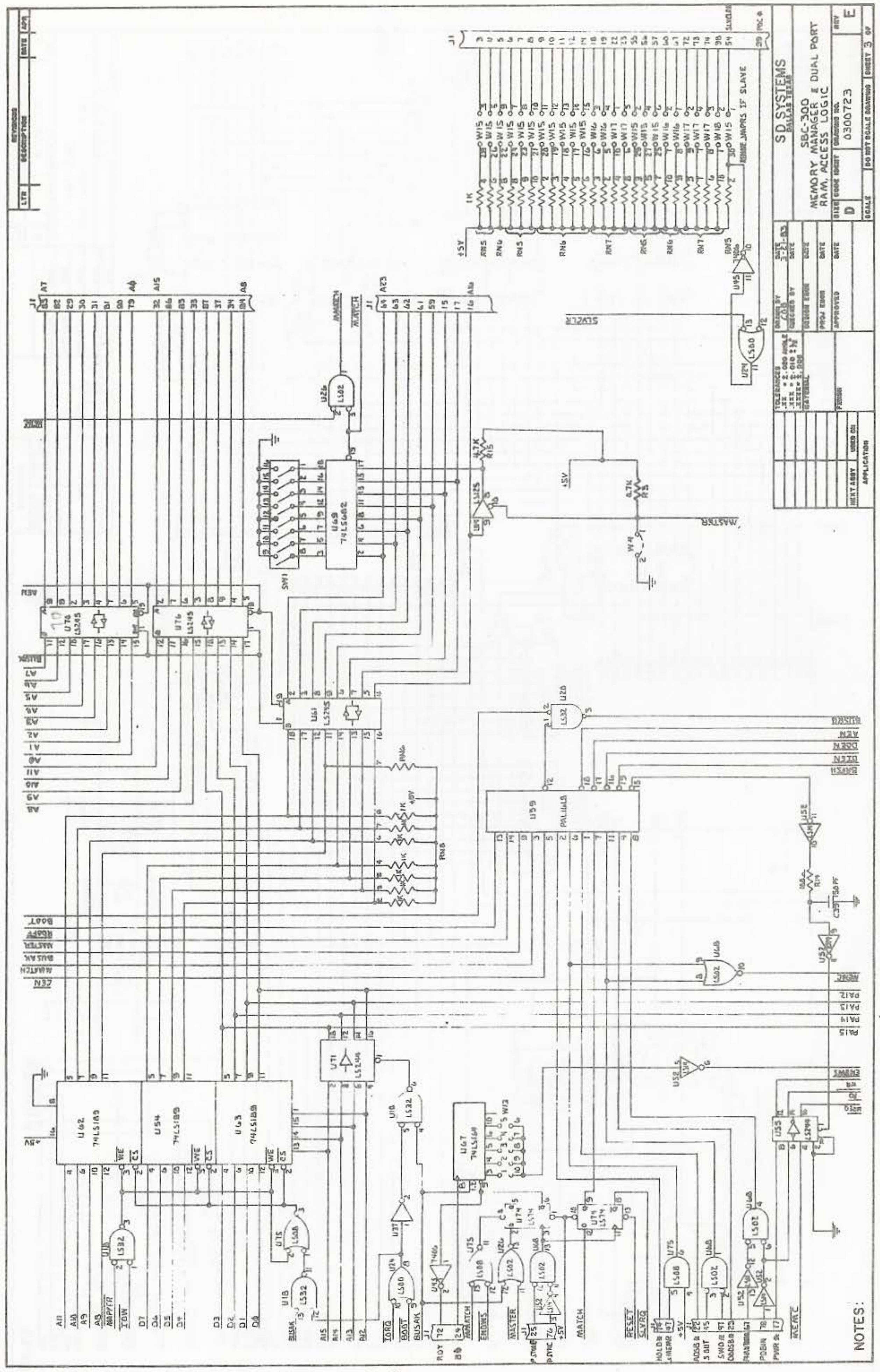

NOTES:
1. USE 74LS92 WITH 12 MHZ XTAL (FOR 6 MHZ OPERATION)
AND 74LS93 WITH 8 MHZ XTAL (FOR 4 MHZ OPERATION)
REVISIONS
A RELEASED
B INC ECO 0229
C INC ECO 0256
D INC ECO 0271
E INC ECO 0294
U48 Z80 A/B
U47 ASCC 8531/A
U46 CIO 8536/A
U55 LS244
SD SYSTEMS DALLAS TEXAS
SBC-300
Z80 CPU & PERIPHERALS
D
0300723
E
SHEET 1 OF 4
NOTES:

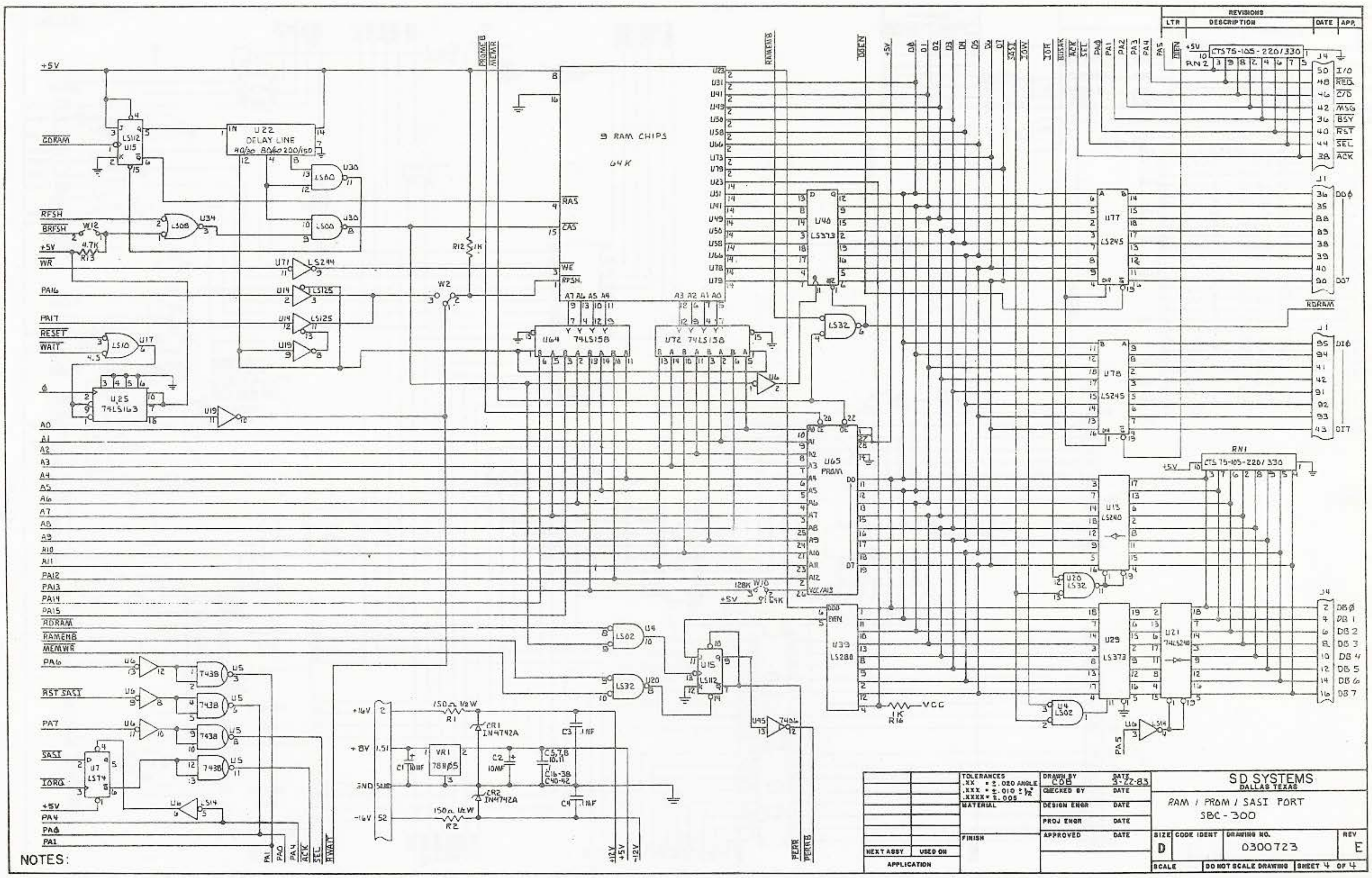


TOLERANCES .XX = ±.020 ANGLE .XXX = ±.010 ±½° .XXXX = ±.005	DRAWN BY	DATE	SD SYSTEMS DALLAS TEXAS
MATERIAL	CHECKED BY	DATE	SBC-300 STATUS / CONTROL
	DESIGN ENGR	DATE	
	PROJ ENGR	DATE	
FINISH	APPROVED	DATE	SIZE D · CODE IDENT · DRAWING NO. 0300723 · REV E
NEXT ASSY · USED ON · APPLICATION			SCALE · DO NOT SCALE DRAWING · SHEET 2 OF

REVISIONS: LTR · DESCRIPTION · DATE · APP.

NOTES:


SD SYSTEMS
DALLAS TEXAS
SBC-300
MEMORY MANAGER & DUAL PORT
RAM. ACCESS LOGIC
0300723
D
E
NOTES:


REVISIONS
LTR
DESCRIPTION
DATE
APP.
9 RAM CHIPS
64K
U22 DELAY LINE
U65 PROM
U64 74LS158
U72 74LS158
U40 LS373
U77 LS245
U78 LS245
U13 LS240
U29 LS373
U21 74LS240
U39 LS280
U25 74LS163
U7 LS74
U15 LS112
CTS 75-105-220/330
RN1
RN2
J1
J4
+16V
+8V
GND
-16V
VR1 7805
CR1 1N4742A
CR2 1N4742A
TOLERANCES .XX ±.020 ANGLE .XXX ±.010 ±½° .XXXX ±.005
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DRAWN BY COB
DATE 3-22-83
CHECKED BY
DATE
DESIGN ENGR
DATE
PROJ ENGR
DATE
APPROVED
DATE
SD SYSTEMS
DALLAS TEXAS
RAM / PROM / SASI PORT
SBC-300
SIZE D
CODE IDENT
DRAWING NO. 0300723
REV E
SCALE
DO NOT SCALE DRAWING
SHEET 4 OF 4

NOTES: